TelePACE Studio

Ladder Logic Training Manual

CONTROL MICROSYSTEMS
SCADA products.... for the distance

TelePACE Studio Ladder Editor Training Manual

Printed in Canada.

CONTROL
MICROSYSTEMS
SCADA products ... for the distance

48 Steacie Drive Telephone: (613) 591-1943
Kanata, Ontario   Fascimile: (613) 591-1022
K2K 2A9   Technical Support: 1-(888)-226-6876
Canada   1-888-2CONTROL

Table of Contents

Relay Ladder Logic

Relay Ladder Logic has been used in the control of industrial processes for many years. Early control circuits were created by wiring input devices such as pushbuttons and limit switches to output devices such as motors. As the processes became increasingly complicated mechanical relays were added to the control circuits. The use of control relays enabled one area of a process to be controlled by another area.

Control Relays

A typical industrial relay is shown in *Figure 1: Basic Relay*. A relay is a coil of wire wrapped around a core. When the switch is closed current flows through the coil and an electromagnet is created. The armatures, 1 and 4, are drawn from the Normally Closed (NC) connection to the Normally Open (NO) connection.

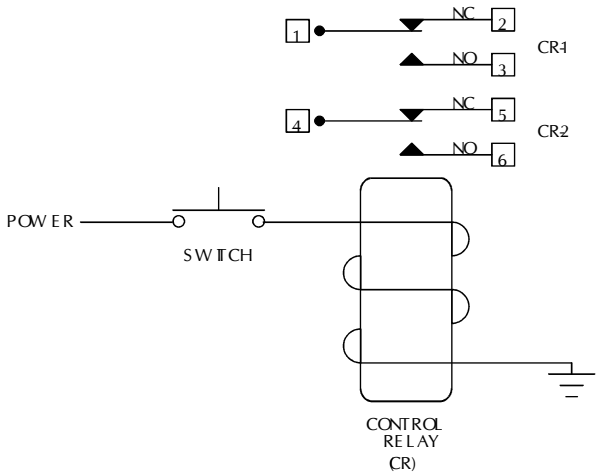


Figure 1: Basic Relay

The relay shown in figure 1 has two sets of contacts, CR-1 and CR-2. When power is applied to the coil, the connection between contacts 1 and 2 of CR-1 is broken and the connection between contacts 1 and 3 is made. The status of the contacts is directly controlled by the current flow through the coil.

Industrial control relays can, and many times do, have large numbers of contact sets. All the contacts of a relay are controlled by the presence or absence of current through the coil of the relay. The use of mechanical relays, timers, delay timers and counters enabled the creation of very sophisticated hardwired control circuits.

Simplified Relay Ladder Logic Circuit

An example of a simplified relay ladder logic circuit is shown in *Figure 2: Relay Ladder Logic Circuit*. Each element in the circuit is described as follows:

- SW 1 through SW 4 are switches.
- CR 1 is a control relay with a NO contact.
- START is a normally open pushbutton switch.
- STOP is a normally closed pushbutton switch.
- M1 is a motor with NO and NC contacts.

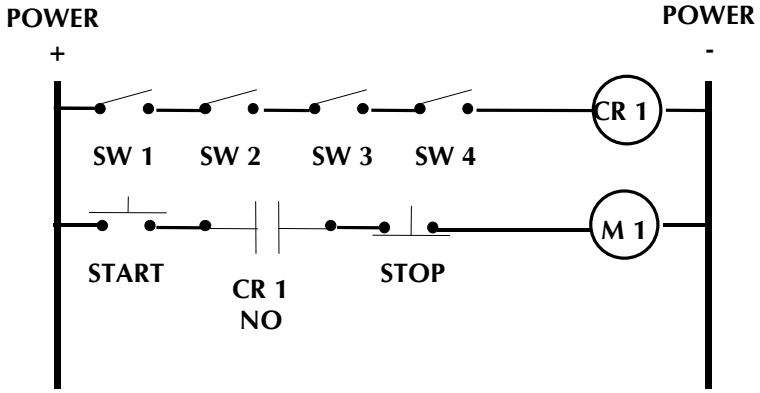


Figure 2: Relay Ladder Logic Circuit

Early Relay Ladder Logic systems had some important limitations:

- The physical size of the control and wiring panels became larger and larger as the complexity of the control systems increased.
- Changes to the control system would involve large scale rewiring of the control and wiring panels.

Programmable Relay Ladder Logic

The TelePACE Relay Ladder Logic, used with Control Microsystems controllers, overcomes the limitations of early relay ladder logic through programmable logic functions. The simplified controller diagram in
Figure 3 shows how the Relay Ladder Logic circuit from exercise 1 would be wired. The physical inputs are wired to digital inputs on the controller. The TelePACE Relay Ladder Logic program operating in the controller will determine whether the MOTOR is started or stopped.

Changing the control system for this example would involve changing the Ladder Logic program. Any combination of switches and pushbuttons can be programmed to turn on or stop the motor. Logic functions such as delay start, motor runtime, number of starts etc. can all be easily added to the control system, without additional wiring.

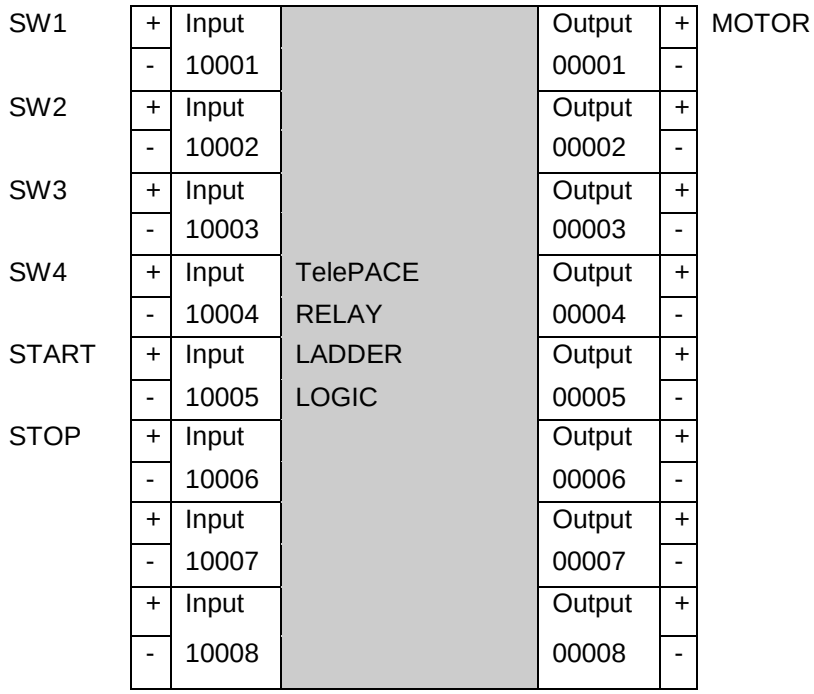


Figure 3: TelePACE Relay Ladder Logic

TelePACE Studio Ladder Logic Program

The major components of a ladder program consist of ladder networks, ladder function elements, and associated comments. It is important to fully understand each of these major elements and how they are executed as a program within the controller.

Networks

A network is a diagrammatic representation of control logic similar to a wiring schematic, showing the interconnection of relays, timers, contacts and other control elements. The number of networks in a program is only limited by the memory available.

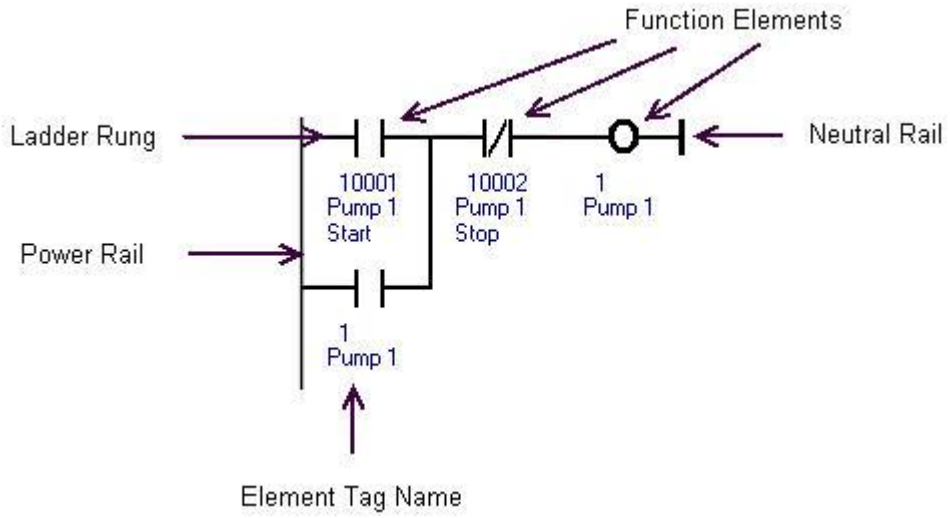


Figure 4: A Ladder Network

Network Elements

Network elements are contacts, coils, and function blocks. Shunts are used to interconnect elements. Coils are always found connected to the neutral rail and represent either physical outputs in the controller or internal (memory only) outputs in the I/O database. Contacts represent either physical status inputs from the controller or internal (memory only) status inputs in the I/O database. Function blocks are used to perform specific functions, such as moving data, manipulating data or communicating data.

Network Comments

Each network in the TelePACE project can have a comment attached that describes the functionality of the network logic. The Network Comments area of the Network Canvas is above the network logic. The comments window may be expanded using the Windows sizing controls.

TelePACE Ladder Editor Environment

TelePACE Studio is a powerful programming and monitoring environment, which enables the user to:

- Create ladder logic programs.
- Edit ladder logic programs On Line or Off Line with a Controller.
- Read and write programs to a Controller.
- Configure the Controller serial communication ports.

TelePACE Studio Display

The TelePACE Studio display provides the interface between the programmer and the TeleSAFE controller. The major sections of the TelePACE Studio display are shown in the following diagram. TelePACE Studio user interface is based on the Microsoft Fluent Interface. This type of interface allows users to focus on creating, editing and monitoring TPS projects rather than trying to remember where the menu commands are located on the menu bar.

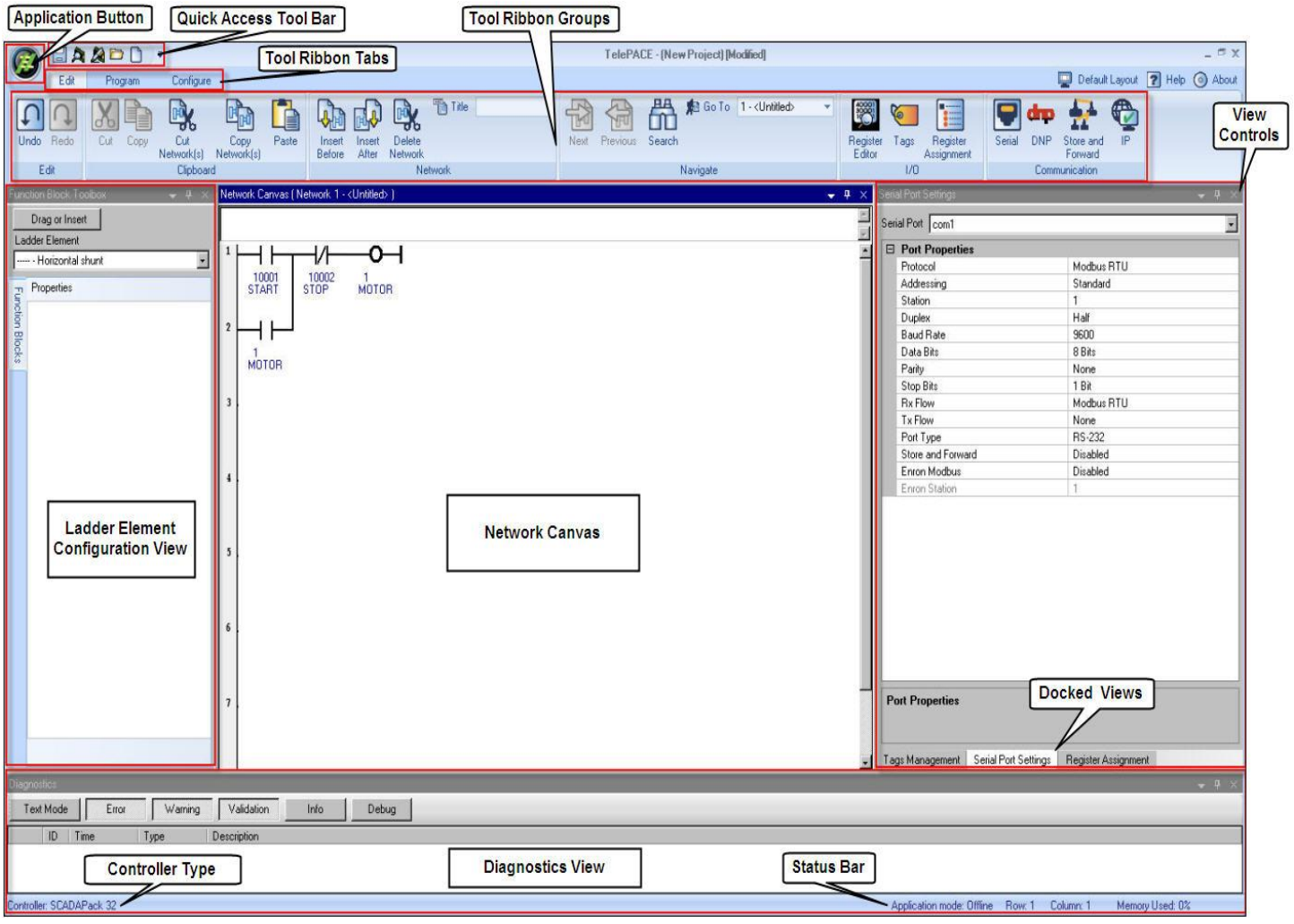


Figure 5: TelePACE Studio Layout

Program Network Layout

The Ladder Logic program consists of the main program followed by a number of subroutines. The main program is defined as all the logic networks up to the start of the first subroutine, or until the end of the program if no subroutines exist. A subroutine is defined as all the logic networks from a subroutine element until the next subroutine element or the end of the program if there are no more subroutines.

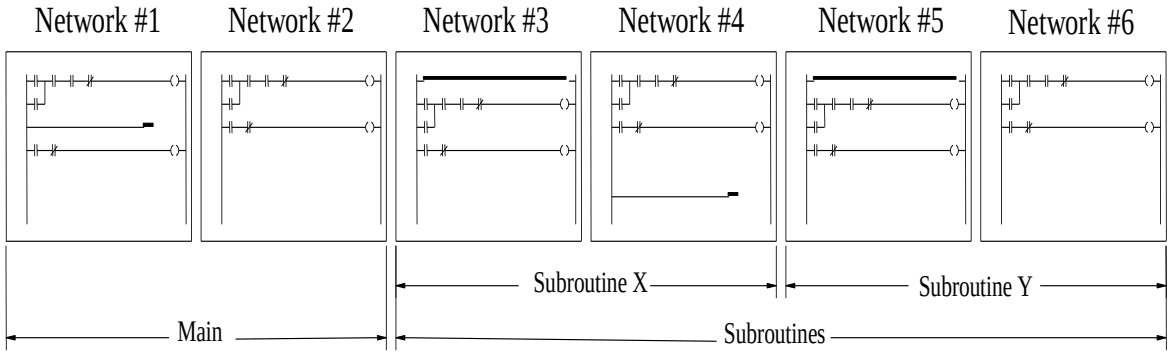


Figure 6: Program Network Layout

In *Figure 6: Program Network Layout*, the main program consists of networks 1 and 2. The next four networks make up the subroutines, with networks 3 and 4 comprising subroutine X and networks 5 and 6 comprising subroutine Y. If subroutines were not needed, the entire program would consist of a main section only.

Program Execution Order

Networks in the TelePACE Studio Ladder Editor may contain up to 8 rungs. Each rung can contain a maximum of 10 logic elements. The highlighted cursor in the Ladder Editor pane occupies one logic element position.

The controller evaluates each element, or function block, in the network in a sequence that starts at the top left hand corner; or ROW 1, COLUMN 1. The ladder evaluation moves down column 1 until it reaches row 8. The evaluation then continues at the top of column 2 and moves down to row 8 again.
This process continues until the entire network has been evaluated. If there is more than one network in the main program, evaluation continues to the next sequential network in the program until the entire main program has been evaluated. Then the evaluation returns to the first network in the main program.
If a subroutine call function block is encountered, execution transfers to the start of the called subroutine and continues until the end of the subroutine. Execution then returns to the element after the subroutine call element.

Subroutine execution can be nested. This allows subroutines to call other subroutines. Subroutine execution cannot be recursive. This prevents potential infinite loops in the ladder logic program.

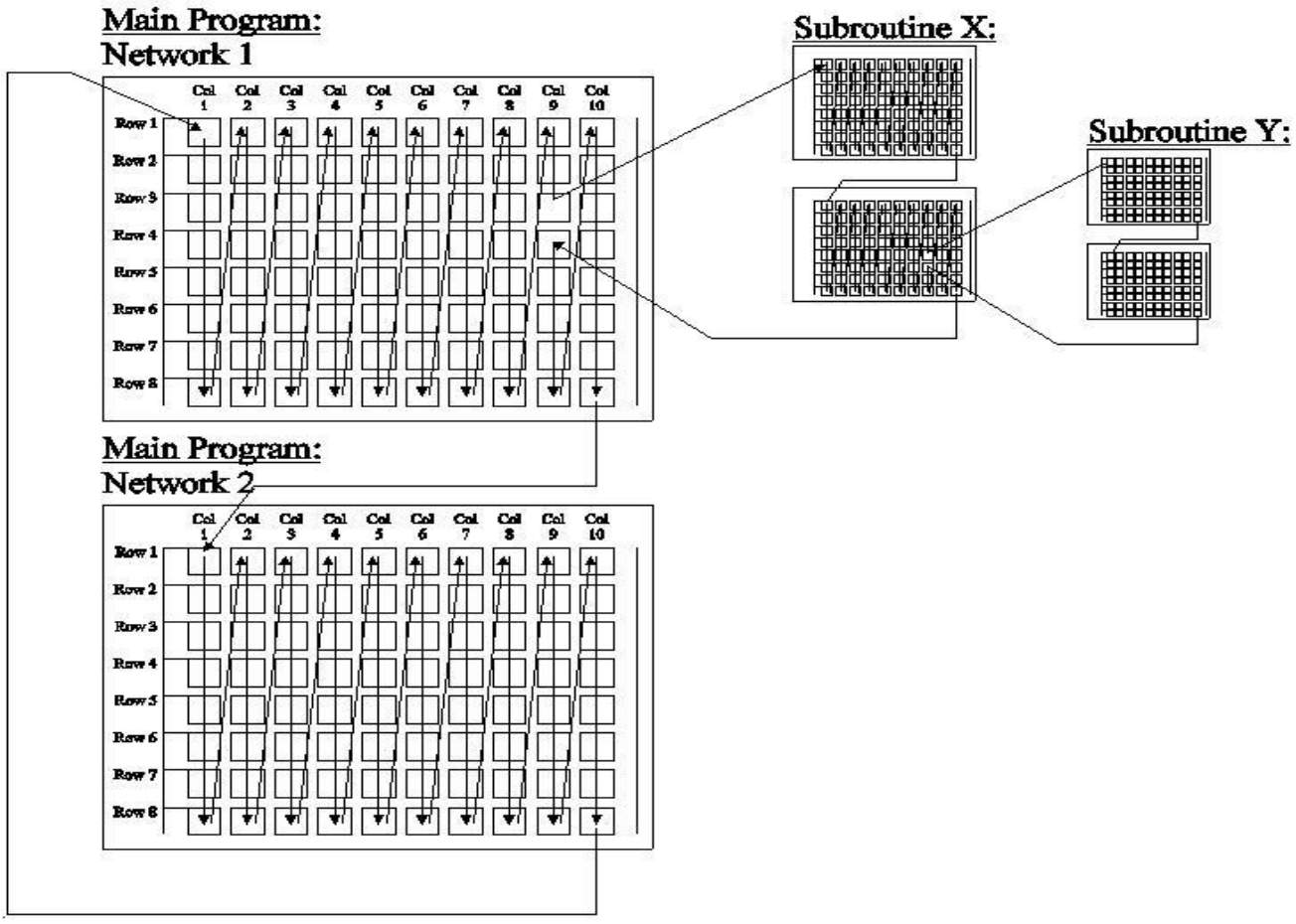


Figure 7: TelePACE Ladder Editor Network Execution

Ladder Logic Memory Usage

Memory usage in a Ladder Logic application program is based on the number of networks and number of elements used by a program. There are 12k words of program space available in all SCADAPack models.
Limiting the number of unnecessary horizontal shunts can reduce the program size. Optimizing other logic such as math calculations, can also ensure that efficient use is made of the available memory.

Networks:

- Each network created in an application requires one word of memory, whether the network is used or not.

Columns:

- Each column that is occupied in a network requires one word of memory.

Single Elements:

- Each single element requires one word of memory.

Double Elements:

- Each double element requires two words of memory.

Triple Elements:

- Each triple element requires three words of memory.

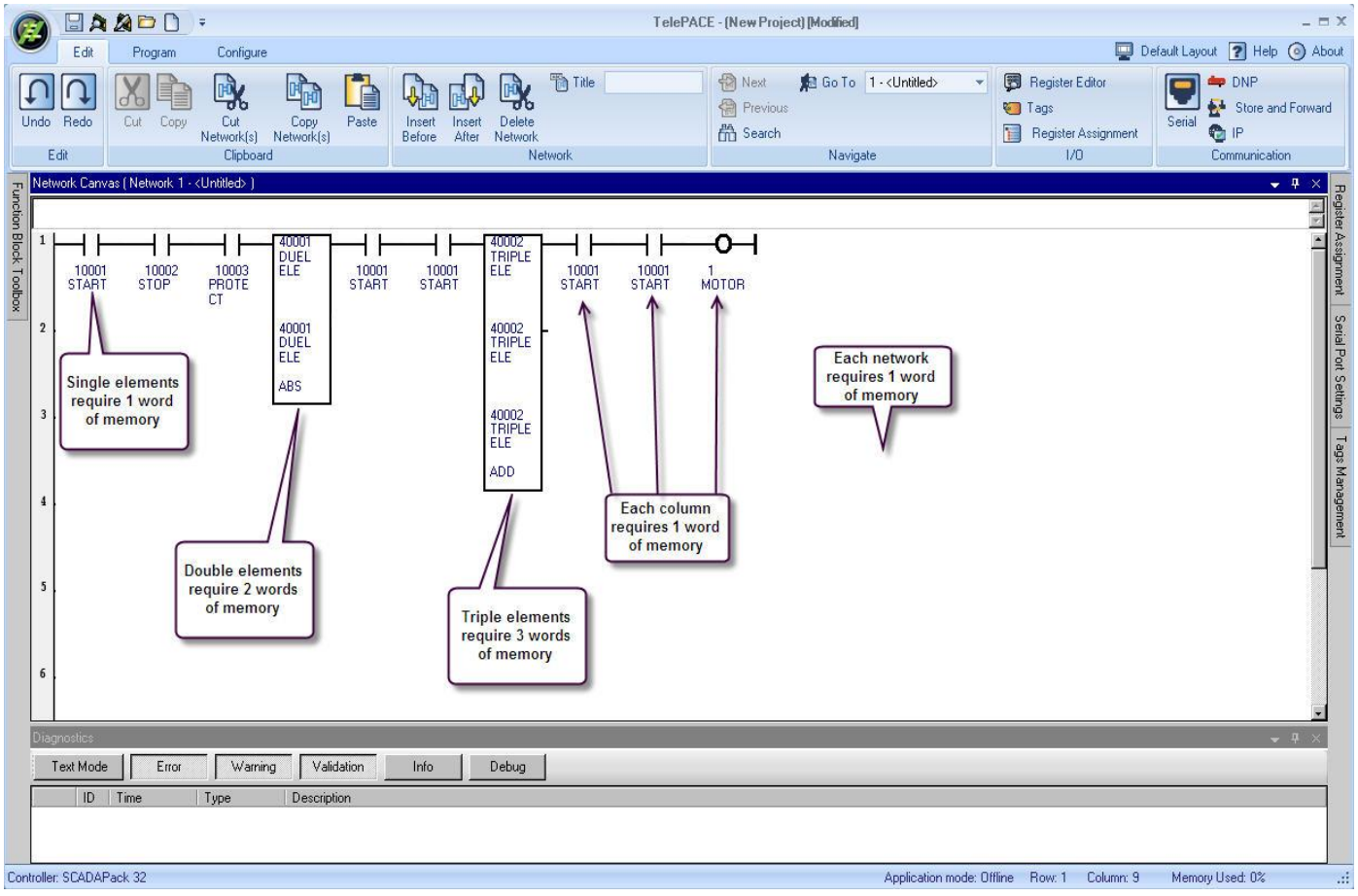


Figure 8: Ladder Logic Memory Usage

I/O Database

The I/O database allows data to be shared between C programs, Ladder Logic programs and communication protocols. A simplified diagram of the I/O Database is shown in *Figure 9: I/O Database Block* Diagram.

The I/O database contains general purpose and user-assigned registers. General-purpose registers may be used by Ladder Logic and C application programs to store processed information and to receive information from a remote device. Initially all registers in the I/O Database are general-purpose registers.

User-assigned registers are mapped directly from the I/O database to physical I/O hardware, or to controller system configuration and diagnostic parameters. The mapping of registers from the I/O database to physical I/O hardware and system parameters is performed by the Register Assignment.

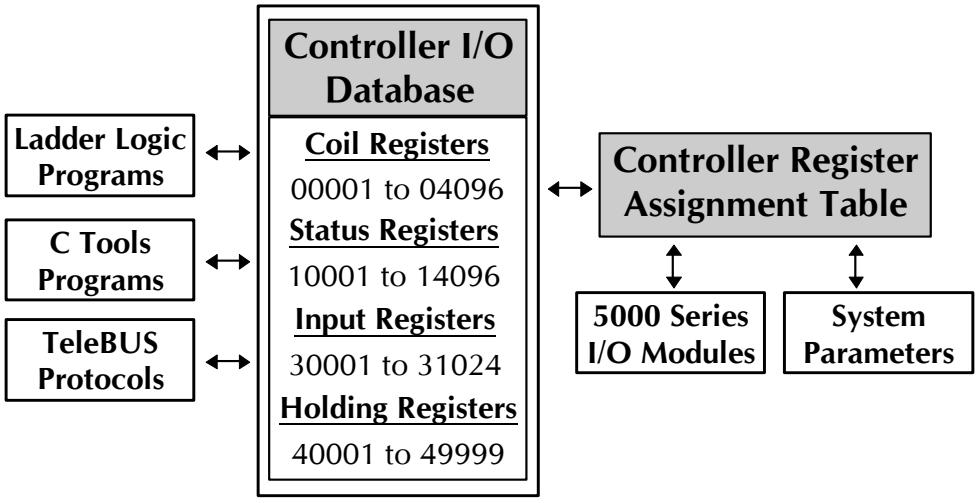


Figure 9: I/O Database Block Diagram

User-assigned registers are initialized to the default hardware state or system parameter when the controller is reset. Assigned output registers do not maintain their values during power failures. Assigned output registers do retain their values during application program loading
.
General-purpose registers retain their values during power failures and application program loading. The values change only when written by an application program or a communication protocol.

The TeleBUS communication protocols provide a standard communication interface to the controller. The TeleBUS protocols are compatible with the widely used Modbus RTU and ASCII protocols and provide full access to the I/O database in the controller.

I/O Database register types

The I/O database is divided into four types of I/O registers. Each of these types is initially configured as general purpose registers by the controller.

Coil Registers

Coil registers are single bit registers located in the digital output section of the I/O database. Coil, or digital output, database registers may be assigned to 5000 Series digital output modules or SCADAPack I/O modules through the Register Assignment. Coil registers may be also be assigned to controller on board digital outputs and to system configuration modules.

- There are 4096 coil registers numbered 00001 to 04096. Ladder logic programs, C language programs, and the TeleBUS protocols can read from and write to these registers.

Status Registers

Status registers are single bit registers located in the digital input section of the I/O database. Status, or digital input, database registers may be assigned to 5000 Series digital input modules or SCADAPack I/O modules through the Register Assignment. Status registers may be also be assigned to controller on board digital inputs and to system diagnostic modules.

- There are 4096 status registers are numbered 10001 to 14096. Ladder logic programs and the TeleBUS protocols can only read from these registers. C language application programs can read data from and write data to these registers.

Input Registers

Input registers are 16 bit registers located in the analog input section of the I/O database. Input, or analog input, database registers may be assigned to 5000 Series analog input modules or SCADAPack I/O modules through the Register Assignment. Input registers may be also be assigned to controller internal analog inputs and to system diagnostic modules.

- There are 1024 input registers numbered 30001 to 31024. Ladder logic programs and the TeleBUS protocols can only read from these registers. C language application programs can read data from and write data to these registers.

Holding Registers

Holding registers are 16 bit registers located in the analog output section of the I/O database. Holding, or analog output, database registers may be assigned to 5000 Series analog output modules or SCADAPack analog output modules through the Register Assignment. Holding registers may be also be assigned to system diagnostic and configuration modules.

- There are 9999 input registers numbered 40001 to 49999. Ladder logic programs, C language programs, and the TeleBUS protocols can read from and write to these registers.

Summary of Modbus Register Types

Register Type	Typical Usage *	Address Range	No. of Registers	Register Size	Value Range	Access
Coil Registers	Digital Outputs	00001 to 04096	4096	1 bit	1 or 0 (ON or OFF)	read and write
Status Registers	Digital Inputs	10001 to 14096	4096	1 bit	1 or 0 (ON or OFF)	read only
Input Registers	Analog Inputs	30001 to 39999 **	9999	16 bits	0 to 65535 in Unsigned format *** -32768 to 32767 in Signed format	read only
Holding Registers	Analog Outputs	40001 to 49999	9999	16 bits	0 to 65535 in Unsigned format *** -32768 to 32767 in Signed format	read and write

* All registers may also be used as general purpose registers to simply save data in memory (e.g. memory for program data, internal variables).

30001 to 31024 for SCADAPack controllers

** 30001 to 39999 for SCADAPack 32

*** When 2 sequential 16-bit registers are used together, the following 32-bit registers are possible:

32 bit Register Format	Value Range
Unsigned Double	0 to 4,294,967,295
Signed Double	-2,147,483,648 to 2,147,483,647
Floating Point	-3.402×10^{38} to 3.402×10^{38} Six digits of precision

Register Assignment

The Register Assignment is used to assign I/O database registers to user-assigned registers using I/O modules. An I/O Module can refer to an actual I/O hardware module such as a *5401 Digital Input Module* or it may refer to a set of controller parameters, such as serial port settings.

5000 Series I/O modules used by the controller must be assigned to I/O database registers in order for the I/O points to be used by the ladder program.

The configuration and diagnostic I/O modules used by the controller must be assigned to I/O database registers in order for the I/O points to be used by the ladder program.

Ladder logic programs may read data from, or write data to the physical I/O only through user-assigned registers in the I/O database.

I/O Module Types

AIN - *Analog Input* modules are used to assign data from physical analog inputs to input registers in the I/O Database. Physical analog inputs are specific 5000 Series analog input modules, generic analog input modules, and internal controller data such as RAM battery voltage and board temperature.

AOUT - *Analog Output* modules are used to assign data from the I/O Database to physical analog outputs. The physical analog outputs are specific 5000 Series analog output modules or generic analog output modules.

DIN - *Digital Input* modules are used to assign data from physical digital inputs to input registers in the I/O Database. Physical digital inputs are specific 5000 Series digital input modules, generic digital input modules, and controller digital inputs.

DOUT - *Digital Output* modules are used to assign data from the I/O Database to physical digital outputs. Physical digital outputs are specific 5000 Series digital output modules or generic digital output modules.

CNTR - *Counter Input* modules are used to assign data from physical counter inputs to input registers in the I/O Database. Physical counter inputs are specific 5000 Series counter input modules and controller counter inputs.

DIAG - *Controller Diagnostic* modules are used to assign diagnostic data from the controller to input or status registers in the I/O Database. The diagnostic data is used to monitor internal controller data such as controller status code, the force LED, serial port communication status and serial port protocol status.

CNFG - *Controller Configuration* modules are used to assign data from I/O Database coil and holding registers to controller configuration registers. The configuration data is used to configure internal controller settings such as clearing protocol and serial counters, real time clock settings and PID control blocks.

SCADAPack modules are used to assign data to registers in the I/O Database, from physical SCADAPack digital and analog I/O. Physical inputs and outputs are specific SCADAPack I/O modules.

Register Assignment View

The Register Assignment View provides the interface between the I/O Database and the physical I/O devices used by the SCADAPack controller. The major sections of the Register Assignment display are shown in the following diagram.

Only the I/O Modules that are defined in the Register Assignment have a connection to Database I/O registers.

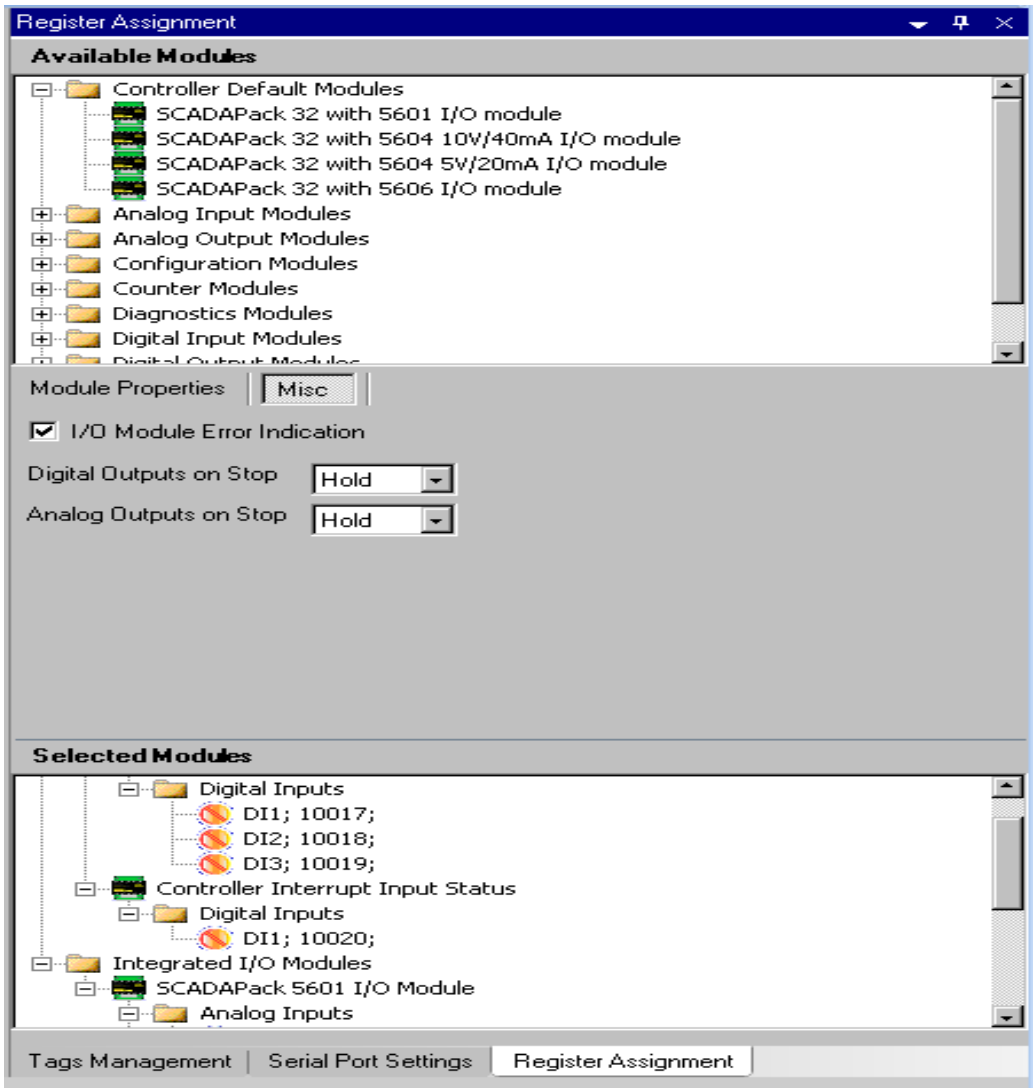


Figure 10: Register Assignment View

I/O Module Error Indication

The Register Assignment view under Misc. contains a check box selection for *I/O Module Error Indication*. The *I/O Module Error Indication* check box determines if the controller displays I/O module communication errors. If enabled, the controller will blink the Status LED if there is an I/O error. If disabled, the controller will not display the module communication status.

The module communication status is always checked. This option controls only the indication on the Status LED.

Assigning Registers to I/O Modules

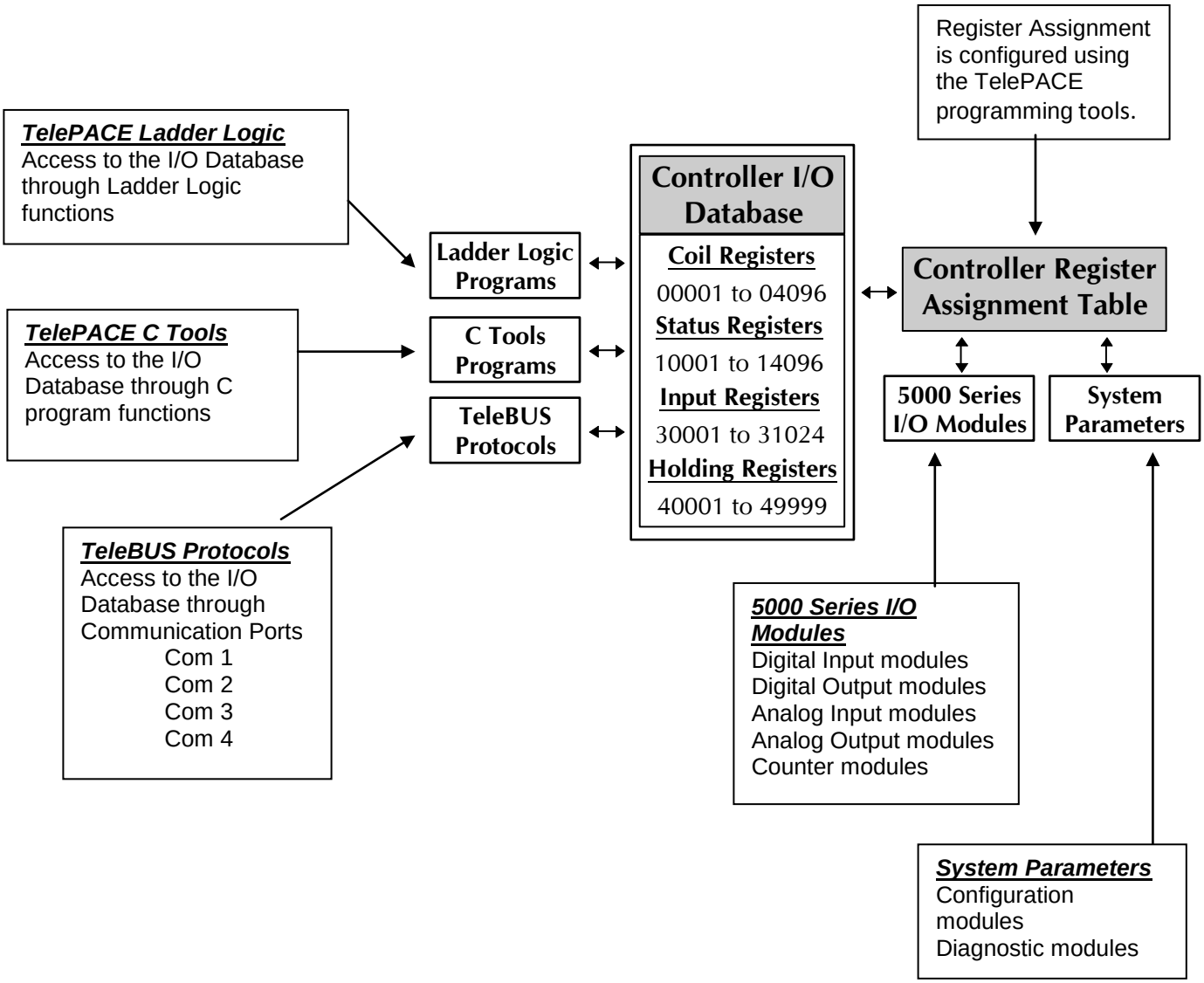
The I/O module reference is found in the TelePACE Studio Help menu of the TelePACE I/O Database Registers. The register assignment table in the I/O module reference is used to describe the information required in the Add Register Assignment dialog.

The information required for each section of the Add Register Assignment dialog is explained in the following table. The shaded column at the left of the table indicates the information that is required for the I/O module in the Add Register Assignment dialog. The requirements vary depending on the I/O module.

Each part of the register assignment table is explained in the following table.

Module	I/O module highlighted in the Module window of the Add Register Assignment dialog.
Address	For hardware I/O modules this is the physical address of the 5000 Series I/O module. This address is set to be equal to the address of 5000 Series I/O module. For Configuration and Diagnostic I/O modules this is the I/O module address or the communication port, depending on the I/O module selected. For I/O modules that do not require an address this selection is grayed out and no entry is allowed.
Type	The I/O databases register type that the I/O module will assign data to. The I/O database register types are: Coil Register 0xxxx Status Register 1xxxx Input Register 3xxxx Holding Register 4xxxx
Start	The I/O database registers address where the I/O module begins the register assignment.
End	The I/O database registers where the I/O module ends the register assignment. This value is automatically set to Start + number of registers required by the I/O module.
Registers	Number of I/O database registers that the I/O module assigns.
Description	The register type description for I/O modules that assign multiple types of registers.

Register Assignment Overview



Ladder Logic Program Development

To demonstrate the features of the TelePACE Studio Ladder Editor, a simple Ladder Logic program will be created. This program development will lead you through the steps required to create a TelePACE Studio Ladder Logic application program. Each of the following steps will be explained in the following chapters.

1. **Open TelePACE Studio and configure your PC settings for type of communications that will be used to communicate with the controller. (Serial)**
2. **Install serial connection between PC and Controller.**
3. **Configure controller for Service Mode.**
4. **Select Controller type.**
5. **Configure Controller serial port settings.**
6. **Configure Controller IP settings if applicable.**
7. **Configure Controller Register Assignment.**
8. **Configure 5000 Series I/O modules used in the application.**
9. **Add Configuration and Diagnostic I/O modules.**
10. **Initialize the Controller.**
11. **Set the controller clock.**
12. **Configure Outputs-On-Stop settings if applicable.**
13. **Title the network.**
14. **Create and comment the Ladder Logic program.**
15. **Save the TelePACE Ladder Logic program.**
16. **Write the Ladder Logic program to the controller.**
17. **Run and test the Ladder Logic program.**

Motor Control Circuit Application

To demonstrate the steps required developing a ladder logic program using TelePACE Studio we will develop a simple ladder logic application.

Figure shows a simple motor control circuit for this application.

Closing the START contact applies power to the motor control coil. A contact on the motor coil holds on the power when the START contact is released. Pressing the STOP switch opens the normally closed STOP contact, and removes power from the motor control coil.

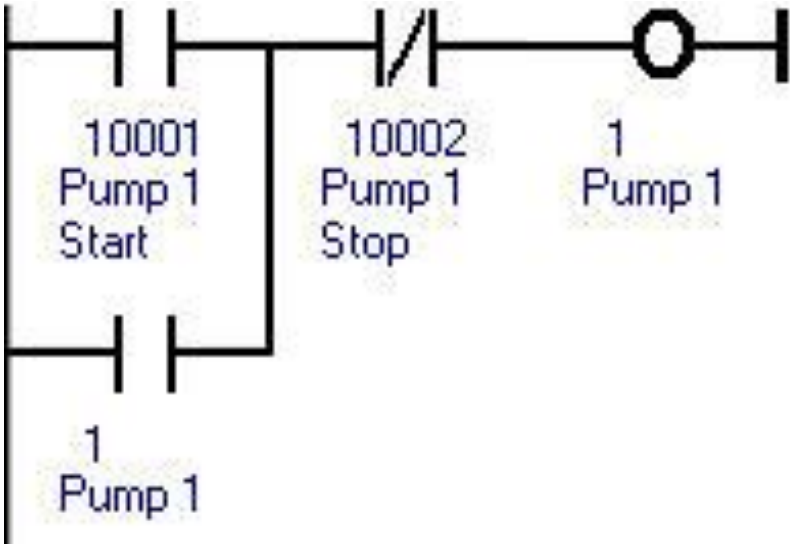


Figure 11: Motor Control Circuit

Configure TelePACE

The configuration of TelePACE Studio involves setting the serial port parameters for the target controller and selecting options to customize the Ladder Editor environment.

PC Communication settings

Configure PC communications to satisfy the requirements of the communication media between the PC running the TelePACE Studio software and the target controller. Enter the parameters used by your personal computer (PC) to communicate with the controller using the PC Communication Settings command under the Program tab on the Tool Ribbon then PC Communications group.

This is for Serial RS232 protocol.

- Select Modbus RTU for the communication Protocol.
- Select Standard Addressing.
- Enter address 1 in the Station address selection box.
- Enter 3 seconds in the TeleSAFE response Time Out selection box.
- Enter 3 attempts in the number of Attempts selection box.
- Select the serial Port your PC will use for communication to the TeleSAFE controller.
- Select 9600 for the Baud rate.
- Select none for Parity type.
- Select 8 bits for the number of Data Bits.
- Select 1 Bit for the number of Stop Bits.
- Select Direct Connection.

The default settings of Modbus RTU protocol, 9600 baud rate, no parity, 8 data bits, 1 stop bit and a station address of 1 are the same settings the Controller will have when it is started in Service Mode.

Install serial connection between PC and Controller

An RS-232 serial communication cable is required to connect the PC serial port to any of the controller serial ports. All of our controllers support a serial link but controllers like the 350 and 330 also have a USB port that can be used for communications and programming.

Configure Controller for Service Mode

When a new controller is first used it should be started in Service mode. Service mode is selected by performing a *Service Boot* using the following procedure:

- Remove power from the controller.
- Press down on the LED POWER button.
- Apply power to the controller.
- Continue holding the LED POWER button until the STAT LED starts turns on.
- Release the LED POWER button.
- If the LED POWER button is released before the STAT LED turns on, the controller will start in RUN mode.

When the controller starts in SERVICE mode:

default serial communication parameters of Modbus RTU, 9600 baud rate, no parity, 8 data bits, 1 stop bit and station address 1 are used for all communication ports:

- the Ladder Logic program is stopped;
- the C program is stopped;
- all programs are retained in non-volatile memory;

? What is the reason for performing a service boot?

Select Controller Type

Click on the Configure tab on the Tool Ribbon and click on the Change Controller in the Controller Group and select the controller needed.

4202 DR Extended/4203 DR I/O Register Assignment
SCADASense 4202/4203 DS I/O Default Register Assignment
Micro16 Default Register Assignment (Controller I/O Only)
SCADAPack (5601 I/O Module) Default Register Assignment
SCADAPack (5604 I/O Module) Default Register Assignment
SCADAPack LIGHT Default Register Assignment
SCADAPack PLUS (5601 I/O Module) Default Register Assignment
SCADAPack PLUS (5604 I/O Module) Default Register Assignment
SCADAPack LP Default Register Assignment
SCADAPack 330 Default Register Assignment
SCADAPack 314 Default Register Assignment
SCADAPack 334 Default Register Assignment SCADAPack 350 Default Register Assignment
SCADAPack 32 (5601 I/O Module) Default Register Assignment
SCADAPack 32 (5604 I/O Module) Default Register Assignment
SCADAPack 32P Default Register Assignment
SCADAPack 100

Configure Controller Serial Port Settings

The communication settings for each serial port on the controller must be configured. There is a set of port settings for each serial port on the controller. The port settings selected will depend on the type of application the controller is being used.

- The controller serial port that is used by TelePACE Studio for programming must be configured with the same settings as the PC Serial Port Settings.

To configure the serial port settings:

- Click on the Serial Port Settings tab in the Docked Views area.
- Enter the required serial port settings for each serial port in the Controller Serial Ports Settings dialog.
- If the application does not require the serial port it is recommended that the default settings for the serial port be used.

Configure Controller Register Assignment

Register assignments are stored in the user configured Register Assignment and are downloaded with the ladder logic application program. Using register assignment is simply a process of adding and configuring I/O modules.

The steps required to configure the register assignment differ depending on the controller used.

During this course, we will use the default settings for the controller supplied by the instructor.

To configure the Register Assignment:

- Click on the Register Assignment tab in the Docked Views area.
- Expand the Controller Default Modules and choose the module installed with your controller by double clicking on the module.

Note: That the module properties and modules selected are now viewable. Explore the selected modules environment.

Configure 5000 Series I/O modules used in the application.

5000 Series I/O modules used in an application must be assigned an individual hardware address for each type of module. This addressing is selected using DIP switch settings on the 5000 Series I/O module. Depending on the type of I/O module the module address will be 0 through 8 or 0 through 16.

The 5000 Series I/O bus will support a maximum of forty I/O (input/output) modules. 5000 Series I/O module types may be combined in any manner to the maximum supported by the controller used. The types of input and output modules available are:

- Digital Input modules
- Digital Output modules
- Analog Input modules
- Analog Output modules
- Counter Input modules

Each type of I/O module, connected to the I/O bus, must have a unique I/O module address. Different types of I/O modules may have the same module address.

For this training session the 5699 I/O simulator is used with the SCADAPack controller. The SCADAPack I/O is configured using the Register Assignment and does not require physical addressing.

Add Configuration and Diagnostic I/O Modules

The controller contains a number of internal configuration and diagnostic I/O modules that may be required by the application.

Controller configuration I/O modules are used to assign data from I/O Database coil and holding registers to controller configuration registers. The configuration data is used to configure controller settings such as clearing protocol and serial counters, real time clock settings and PID control blocks.

Controller diagnostic I/O modules are used to assign diagnostic data from the controller to input or status registers in the I/O Database. The diagnostic data is used to monitor internal controller data such as controller status code, the force LED, serial port communication status and serial port protocol status.

Initialize the controller

The initialize command is used to restore a controller to default settings. This is typically done when starting a new project with a controller.

- From the Controller group select Initialize and then click on the Docked View – Initialize Controller.
- Select the items to initialize in the Initialize Controller dialog.
- Check Erase Ladder Logic Program to clear the ladder logic program memory in the controller.
- Check Erase C Program to clear C application program memory in the controller.
- Check Initialize Controller to reset all values in the I/O database to default settings.
- Check Erase Register Assignment to remove all register assignments.

The Initialize Controller selection:

- Removes all register forces in the controller.
- Sets controller serial port configurations to the default settings.
- Sets all I/O database registers used for Element Configuration to zero.

Configure Outputs on Stop

The controller analog and digital outputs are configured to maintain their value or go to the default state when the ladder logic program execution is stopped.

To configure the outputs on stop setting:

- In the Register Assignment tab in the Docked View and click on the Misc button.
- This dialog controls the state of the controller analog and digital outputs when the ladder logic program is stopped.
- The state of the digital outputs may be set to hold their last value or to turn off when the ladder program is stopped.
- The state of the analog outputs may be set to hold their last value or to go to zero when the ladder program is stopped.

Create and Comment Ladder Logic Program

Develop the TelePACE Studio Ladder Logic program to meet the requirements of the application. It is strongly recommended that program comments be added as the program is developed.

See *Figure 12*.

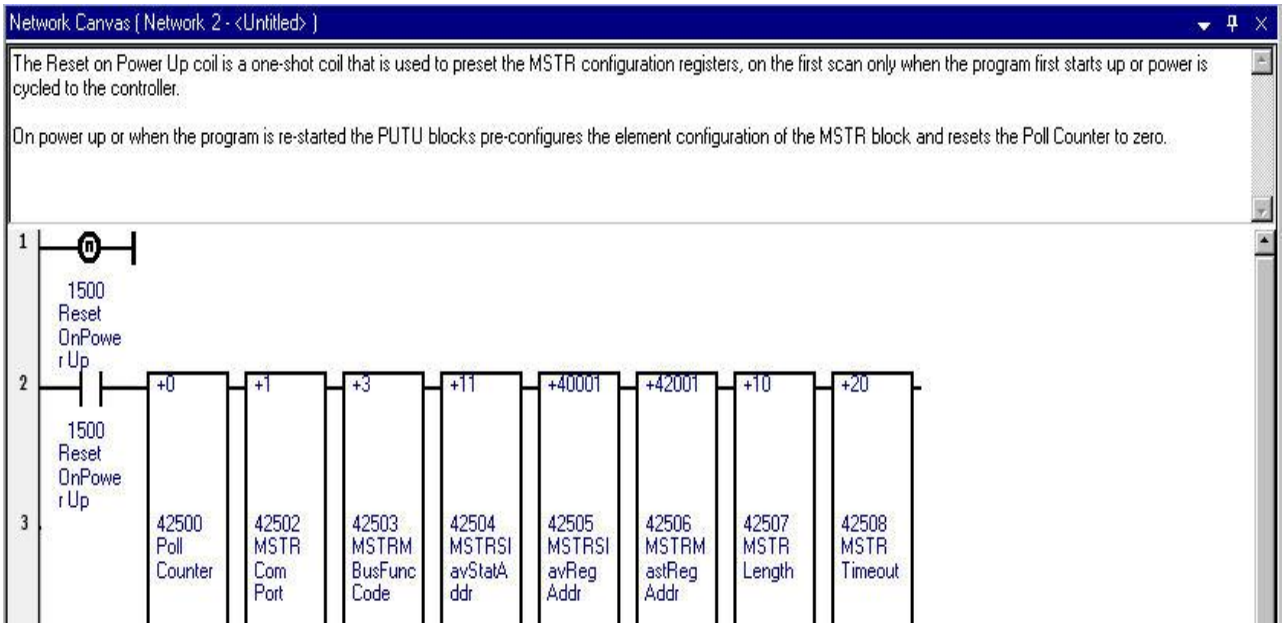


Figure 12: Adding Comments to each Network

Creating Tag Names

Tag names describe I/O points by names that are familiar to you. They make a ladder logic program much easier to read.

The first tag for the motor control example is START. The tag names are a maximum of 32 alphanumeric characters.

Select Tags Management Tab from the Docked View. The Tag Management dialog appears (see *Figure*).

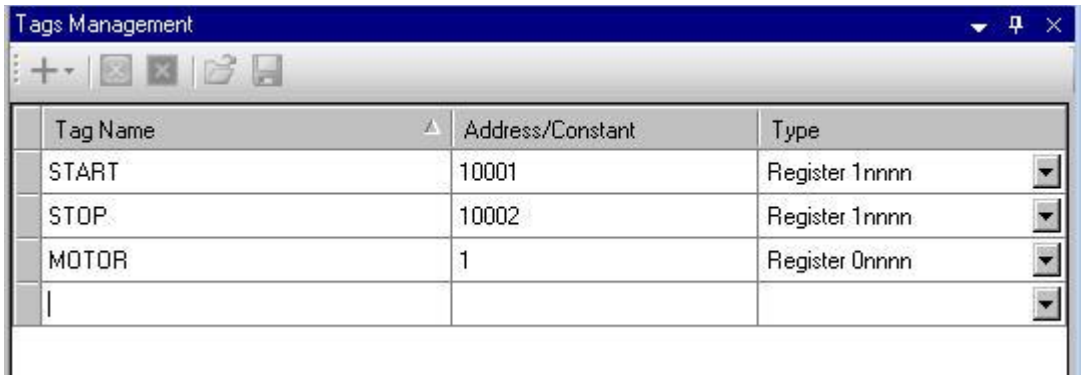


Figure 13: Edit Tag Names Dialog Box

Creating Tag Names:

- Type a tag name into the Tag Name edit box.
- The address of START can be any of the input addresses.
- To assign START to the first Digital Input, enter 10001 in the Address/Constant box.
- The tag name START is to be assigned an I/O address of 10001.
- The Type selection box assigns the value in the Address/Constant box as an Address or constant.
- To assign STOP to the second Digital Input, enter 10002 in the Address/Constant box.
- The tag name STOP is to be assigned an I/O address of 10002.
- The Type selection box assigns the value in the Address/Constant box as an Address or constant.
- To assign MOTOR to the second Digital Input, enter 1 in the Address/Constant box.
- The tag name STOP is to be assigned an I/O address of 1.
- The Type selection box assigns the value in the Address/Constant box as an Address or constant.

Entering Ladder Logic Networks

Ladder logic networks define the control actions of the controller. Networks are entered using the mouse and the keyboard.

- Move the mouse pointer to the Ladder Editor pane.
- Click the left button on the mouse and the Ladder cursor is displayed. Move the cursor to the position of Column 1 and Row 1. Right click on the position with the mouse and move your mouse to the Add Function Block and choose the desired element.

As shown in (*Figure*).

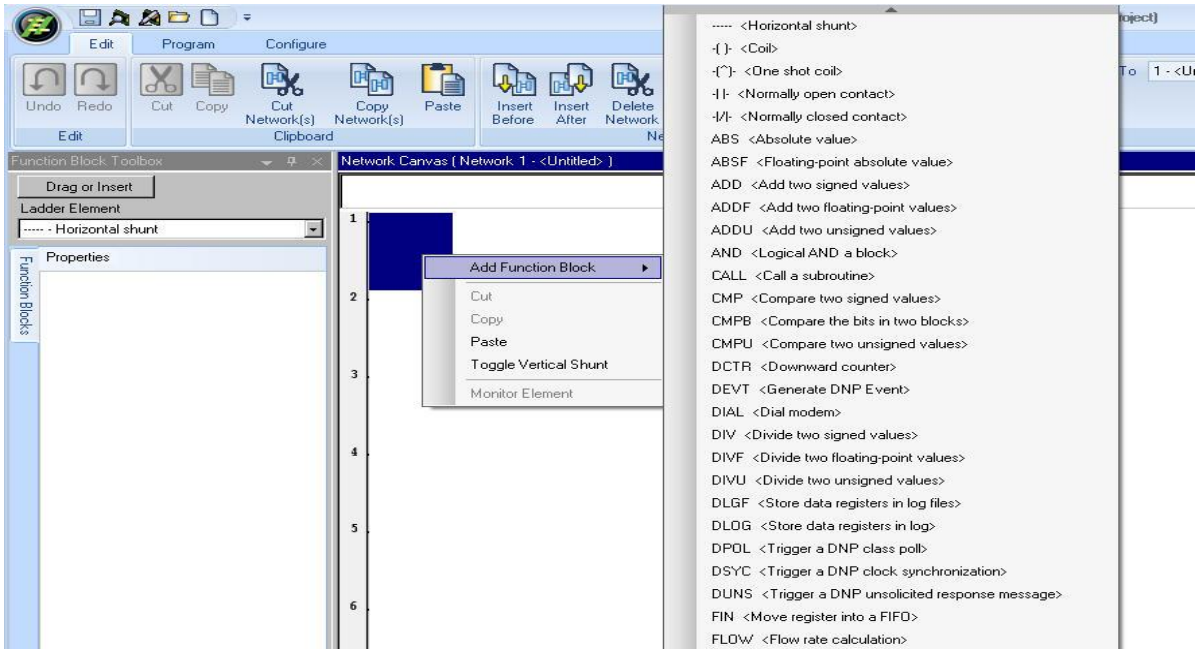


Figure 14: Insert/Edit Network Element

Click on normally open contact in the Function list.

Click on the drop down arrow on the Input Tag Name list box to display tags that apply to the current input type. Select a tag from the list. The first element in the example is the START input. Select START from the list box.

Click on the OK button to insert the element. Repeat these steps for each element of the logic network.

Position the mouse pointer over the position to the right of the first element. Follow the steps above to insert a normally closed contact with the address STOP.

Elements may also be entered from the keyboard. Press the right arrow key. The edit cursor will move one element to the right. Press the Insert key. Follow the steps above to insert a coil with the address MOTOR. Position the mouse pointer over the position under the first element. Follow the steps above to insert a normally open contact with the address MOTOR. To insert vertical shunts, position the mouse pointer over the element where a shunt is required, and click on the right mouse button. For the motor control example, insert a vertical shunt on the START contact element.

You can also enter function blocks into the network through the Ladder Element Configuration View. Using this method will give you a bit more functionality. Your instructor will demonstrate this in class.

Adding Comments

Text can be inserted into the Comment Editor pane that describes the logic in the network. This is a valuable documentation tool that helps make programs easier to understand in the future. The comments entered are for each network. The comment editor is blank when a new network is created.

To enter network comments:

- Position the mouse pointer in the comment editor pane and click the mouse button.
- Size the comment editor pane to view the amount of text you require.
- Enter the text you wish. The comment editor uses the Windows system font for the text.

To enter network titles:

- Double click on the Network Title field in the Edit Tab of the Tool Ribbon and in the Network group click in the Title area.
- A 30-character name for the network can be entered.

Save the Ladder Logic Program

You can save your ladder logic program to disk at any time. Saving your programs as you work protects you against lost work. We recommend making backup copies of your completed programs and storing them in a safe place.

- Position the mouse pointer over the Application button and click the left mouse button and select Save from the menu. The standard Windows Save As File dialog

- Figure) appears because the file has no name. If the file has a name, it is saved, and no dialog appears or you can just click the Save Icon that looks like a 3.5” floppy.

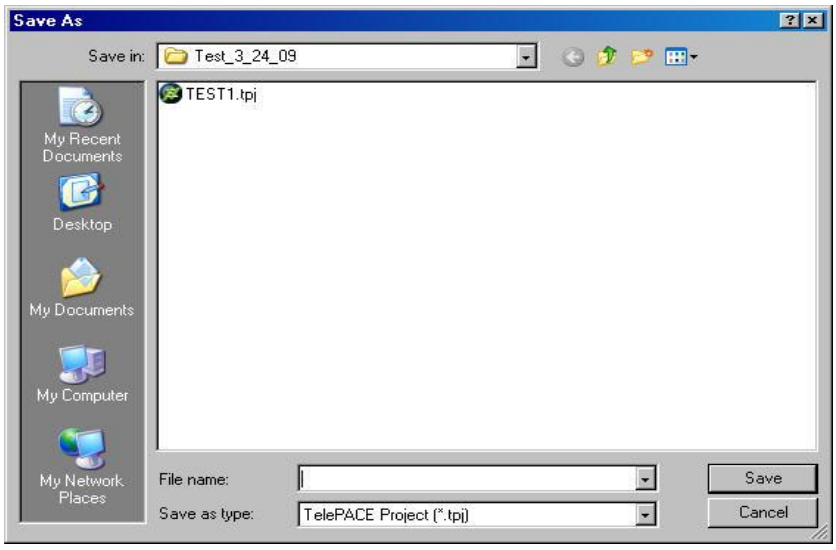


Figure 15: Save As File Dialog

Enter the file name. For this example enter MOTOR. It is not necessary to enter the extension. It is supplied automatically, if you do not type it.

Click on the Save button. The file is saved a MOTOR.tpj.

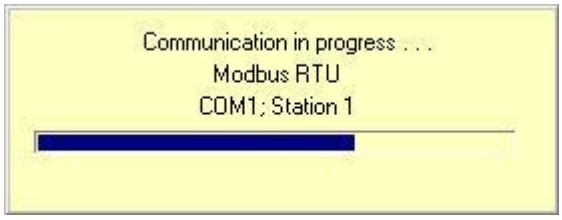
Write Ladder Logic Program to the Controller

The ladder logic program created in the TelePACE Studio Ladder Canvas must be written to the target controller before program execution can occur. When a ladder logic program is written to a controller it replaces any ladder logic program in the controller. If the program in the controller is executing a dialog box will request whether to stop execution of the new program when the write is complete or to continue execution of the new program after the write is complete.

The controller executes the ladder logic program. To write a program to the controller, Select the Program tab on the Tool Ribbon and in the Program group click Write. The Communication Progress dialog *(*
Figure*)* appears. Make your choice, and the program will be written to the controller.

Figure 16: Write to TeleSAFE Communication Progress Dialog

The download progress is shown in the Serial Communication dialog.



Run the Ladder Logic Program

Once the program is written to the controller the controller must be put in the Run mode.

Click on the Program tab on the Tool Ribbon and in the Execution Group select Run.

The RUN LED on the controller will indicate that the controller is in the Run mode.

The ladder logic program in a SCADAPack controller is not executing if the RUN LED is not on.

Monitoring Execution

The last step in completing a ladder logic application program is to test the program execution to ensure it meets the requirements of the application. The Monitor On line mode enables the real time monitoring of a program executing in a controller.

The TelePACE Studio Network Canvas displays the power-flow through the network on the screen. No changes can be made to the program while in monitor mode. The Register Editor is displayed to the side of the Network Canvas.

To monitor a program on line:

Click on the Program tab on the Tool Ribbon and in the Mode Group, click on the Monitor button.

A Communication Progress dialog similar to the figure below appears.

Figure 17: Communication Progress Dialog

When the Monitor button is clicked the controller checks the program that is loaded into memory with the program in TelePACE. If the programs are not exactly similar an error message is displayed.

To test the motor control program:

Apply power to the START input. Observe that the MOTOR output is on. Observe the power-flow through the contacts to the MOTOR coil.

Remove power from the START input. The MOTOR hold in contact maintains the power-flow to the MOTOR coil.

Observe that the output of the START contact is not powered, indicating that power is not flowing through the contact.

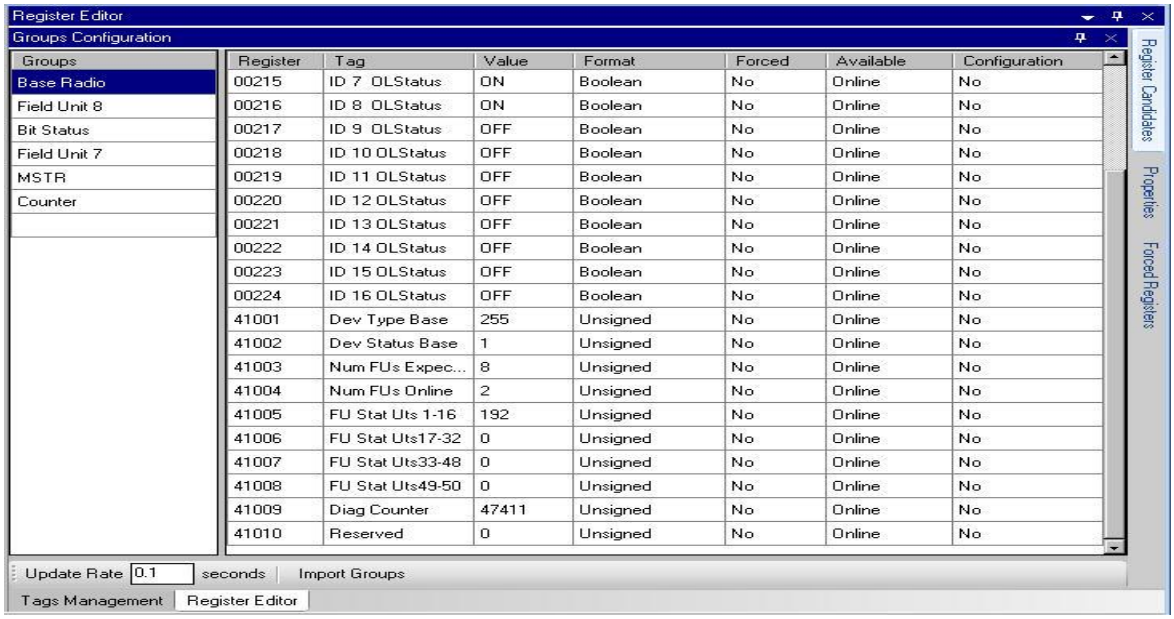
Apply power to the STOP input. Observe that the MOTOR output is off. Observe there is no power-flow through the contacts to the MOTOR coil.

Remove power to the STOP input.

The motor control circuit is functioning properly.

Monitoring Registers On line

The Register Editor enables you to create a list of discrete points or registers that will be displayed in real time on top of the Ladder Editor pane when the Ladder Editor is in the Debug operation and Monitor On line activity.



Register	Tag	Value	Format	Forced	Available	Configuration
00215	ID 7 OLStatus	ON	Boolean	No	Online	No
00216	ID 8 OLStatus	ON	Boolean	No	Online	No
00217	ID 9 OLStatus	OFF	Boolean	No	Online	No
00218	ID 10 OLStatus	OFF	Boolean	No	Online	No
00219	ID 11 OLStatus	OFF	Boolean	No	Online	No
00220	ID 12 OLStatus	OFF	Boolean	No	Online	No
00221	ID 13 OLStatus	OFF	Boolean	No	Online	No
00222	ID 14 OLStatus	OFF	Boolean	No	Online	No
00223	ID 15 OLStatus	OFF	Boolean	No	Online	No
00224	ID 16 OLStatus	OFF	Boolean	No	Online	No
41001	Dev Type Base	255	Unsigned	No	Online	No
41002	Dev Status Base	1	Unsigned	No	Online	No
41003	Num FUs Expec...	8	Unsigned	No	Online	No
41004	Num FUs Online	2	Unsigned	No	Online	No
41005	FU Stat Uts 1-16	192	Unsigned	No	Online	No
41006	FU Stat Uts17-32	0	Unsigned	No	Online	No
41007	FU Stat Uts33-48	0	Unsigned	No	Online	No
41008	FU Stat Uts49-50	0	Unsigned	No	Online	No
41009	Diag Counter	47411	Unsigned	No	Online	No
41010	Reserved	0	Unsigned	No	Online	No

Figure 18: Ladder Logic Memory Usage

The Register Viewer pane displays the following information:

- The Groups area displays the groups that have been created for this TelePACE Studio project to display groups of registers of interest and display the registers contained in the each group.
- The Register grid section displays the registers, and register attributes, for each register in a group.
- The Register Candidates tab is used to add registers to the selected group.
- The Properties tab is used to define the properties of each register within a group.
- The Forced Registers tab displays registers in the project that are forced this tab also allows the un-forcing of selected registers.

There are two addition functions at the bottom of the Register Viewer view:

- The Update Rate field determines how often the register viewer updates values when on-line. Valid values are 0.1 to 1000 seconds. The default value is 1 second. This is a direct entry field and the value entered will be used when the enter key is clicked or the mouse is navigated to another area. An error message is displayed if the value is outside the valid range.
- The Import Groups command allows the user to import register groups from other existing TelePACE projects. See the Register Groups section for information on how to import Groups.

When the TelePACE project is in the online mode, including edit online or monitor online, all of the register settings of the current selected group are read from the controller. If communication fails with the target controller in the online mode, the display is not be updated and a message is displayed in the Diagnostics View.

Adding Registers to the Register Editor:

In the Register Editor click on the Register Candidates and add the register you want to monitor or range of registers to monitor.

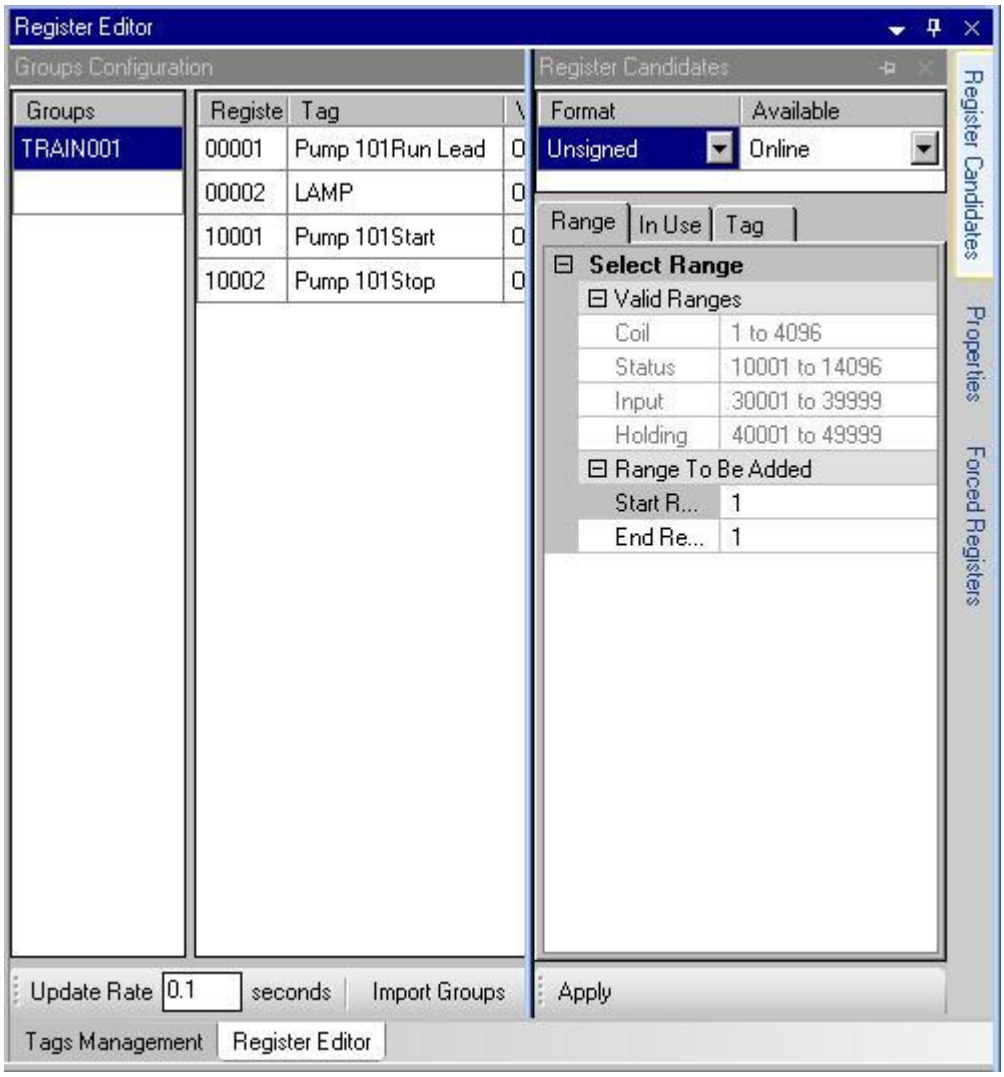


Figure 19: Ladder Logic Memory Usage

Contact Power-flow Monitoring:

It is often necessary, while in Monitor On line mode, to determine whether a contact would pass power if power were supplied to it. This feature is extremely useful when testing the operation of ladder logic programs.

In Monitor On line mode, TelePACE Studio shows the power-flow through the network. It also colors contacts that are not powered to show how power would flow if they were.

Figure 20: Contact Power-flow Monitoring shows how power-flow information is displayed in Monitor On line mode.

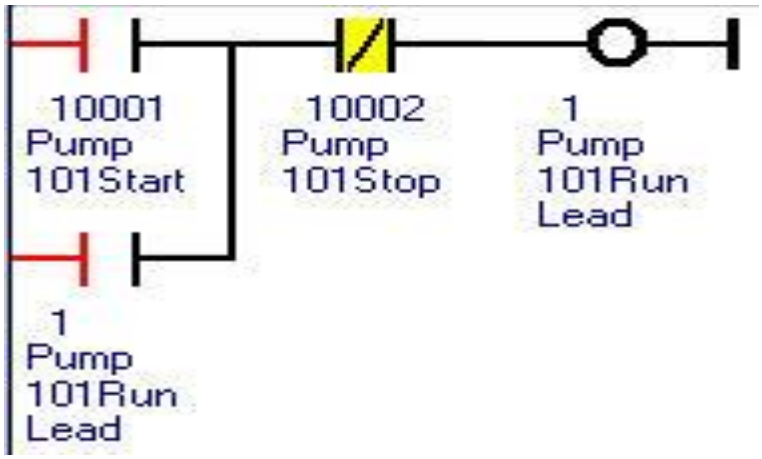


Figure 20: Contact Power-flow Monitoring

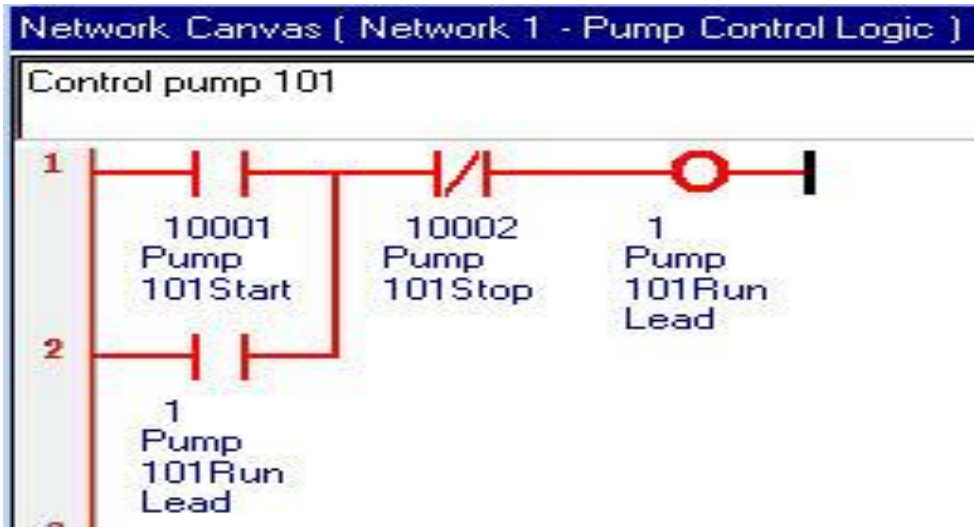
In figure 20**,** normally open contact 10001 and normally closed contact 10002, are not energized. The shaded background, in contact 10002, indicates power would flow through the contact if the input side were powered.

The background of a normally open contact is colored when it is energized and its input is not powered.

The background of a normally closed contact is colored when it is not energized and its input is not powered.

The background is only displayed on contacts if the input side of the contact is not powered.

This is what it would look like powered:



Editing a Program On line

Changes are often required to a program already executing in a controller. You could make the changes off line, and then write the entire program into the controller. However, it is often more convenient to make changes on line.

On line editing changes the program in the controller and the editor at the same time. The program can continue to execute in the controller while changes are made, or you can stop the program.

For example, assume a lamp is added to the motor control circuit, as shown in Figure . The lamp contact is on when the motor is running.

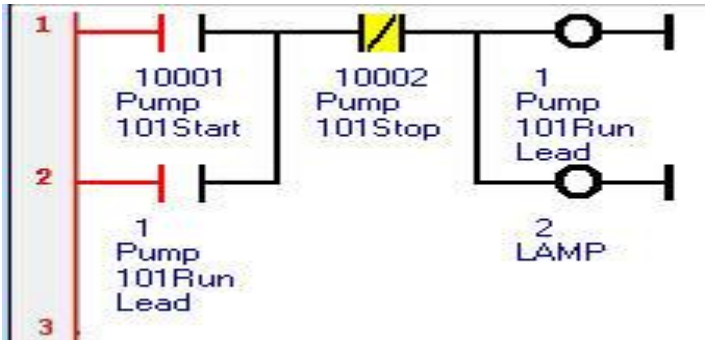


Figure 21: Revised Motor Control Circuit

To edit the program on line:

Click on the Program tab on the Tool Ribbon and in the Mode Group click on Edit On Line.

To enter the new tag name:
Select Tags Management Tab from the Docked View.
Enter the tag name LAMP. The output address is 00002, and the tag type is address.

Insert the element by positioning the mouse pointer in row 2, column 3 and insert a coil. The Insert / Edit Network Element dialog *(Figure)* appears.
Select coil from the Function list.

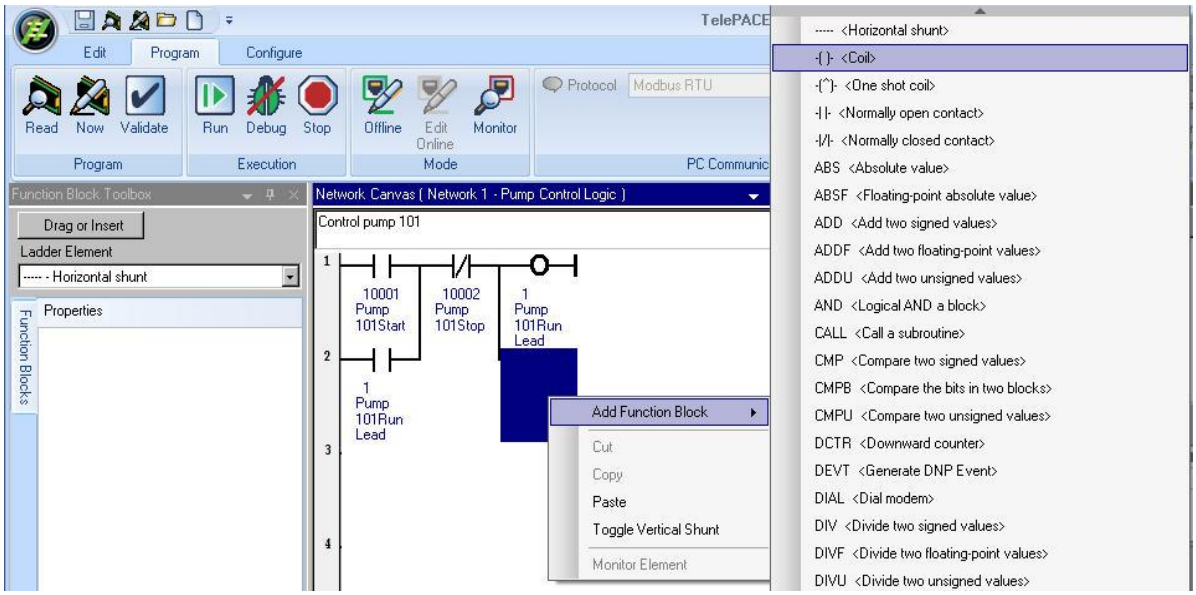


Figure 22: Insert / Edit Network Element

Click on the Input Tag List drop down box. Select the tag LAMP from the list.
Add the Vertical Shunt and then click the Monitor button. A question box will appear:

Click OK then another message box opens:

Be careful when selecting the Yes option. The program will execute with the changes you make, even if the changes are not complete. This may cause undesired operation. Select No if you are making multiple changes.

The change to the program is now complete. Test it as described in the *Monitoring* Execution section.

Editing Off line

Once the program is completed and tested, all or parts of the program can be used to control a similar process. To demonstrate this procedure we will use motor control to control a motor in another network. This will also demonstrate the copy and paste features of the editor.

To edit the program off line:
Click on the Off Line button in the Mode group.
Click on the Edit tab on the Tool Ribbon then click on the Network After in the Network Group. (see *Figure*)

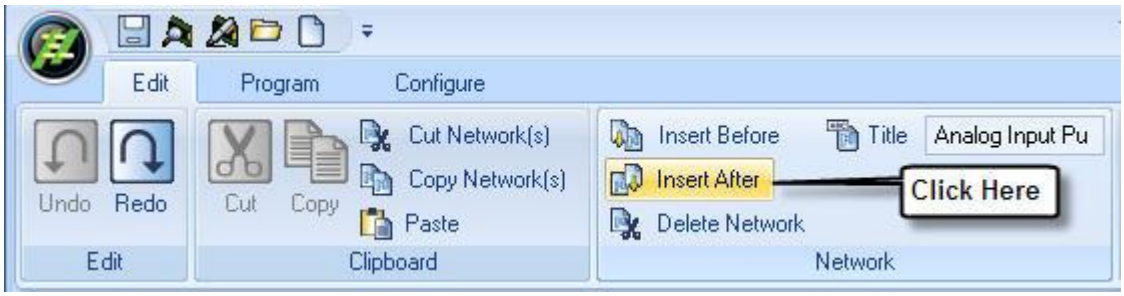


Figure 23: Insert Dialog Box

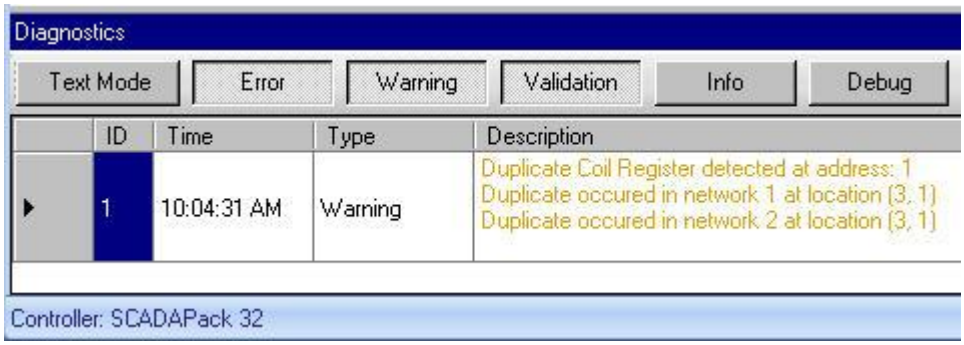
We want to copy this entire network except the LAMP.
To do this position the cursor at the top left position of the motor control network, over the START contact. Click on the first element in row 1 column 1 and then hold your Shift button down on your key board and left click on the bottom right of all the ladder. Next hold your CTRL button down and left click on the LAMP coil to de-select it.

Use the Ctrl+C to copy the selected elements to the clipboard or put your mouse on the highlighted elements and right click and select copy.

From the Edit tool ribbon in the Navigate group click on the Next button to move to network 2. Position the cursor in the upper left most position in the network and right click and select paste. The elements selected from Network 1 are now pasted into the position of the cursor in Network 2. The START contact, MOTOR contact and STOP contact and the vertical shunt to the right of the STOP contact are now in Network 2.

Delete the vertical shunt by positioning the cursor over the STOP contact and hit your F5 key.

The warning alerts you to the fact that there are two circuits controlling the same output coil



We will change the address of the coil later. We want the control in Network 2 to control a different motor so the Tag Names and Addresses for the START, and STOP contacts and the MOTOR coil and contacts must be changed. To change the tag name and address of the START contact in Network 2. Add the Tag name START2 with the Number 10003. Click on the Tags Management tab in the Docked Views area and click on the empty Tag Name box and type START2 with an address of 10003. Repeat this procedure for each element in Network 2 changing the Tag Names to STOP2 and MOTOR2. (Tag Name STOP2 will be Number 10004 and Tag name MOTOR2 will be Number 00003).

Save your work and write the new Ladder to the controller. Test the program.

Forcing Registers

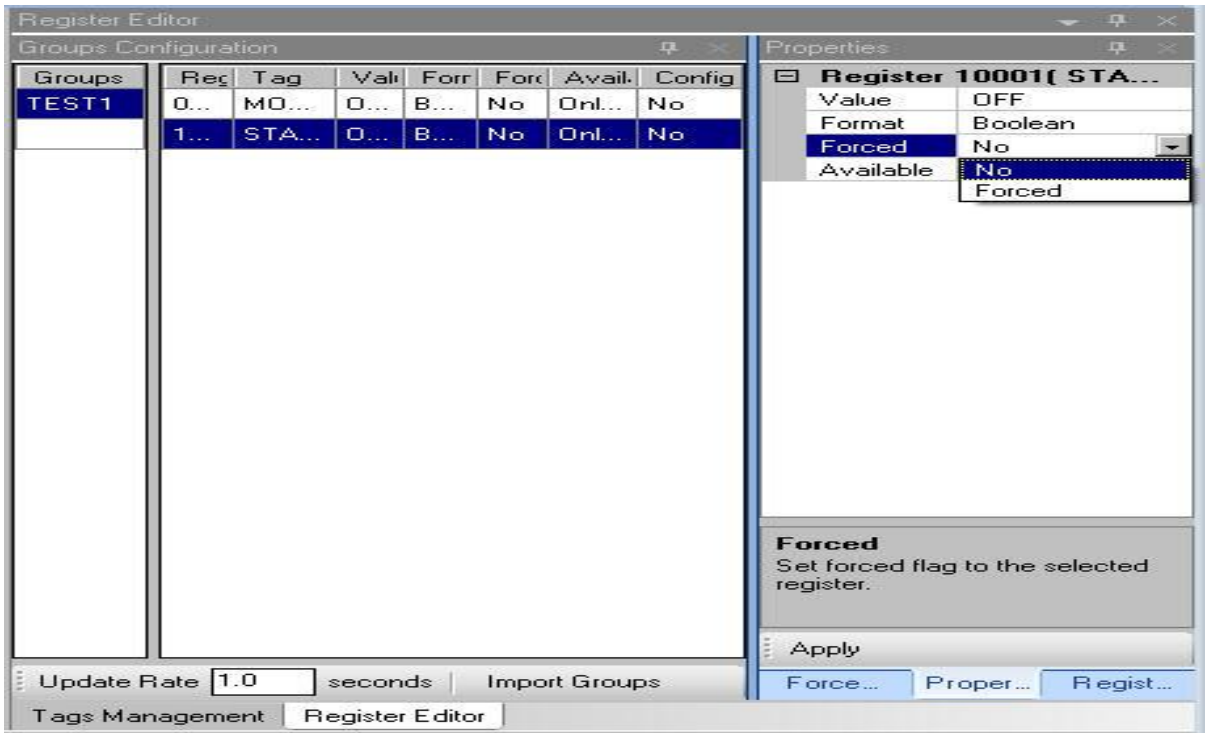
It is often convenient and necessary to be able to change the value of a register while the program is in operation. Toggling a digital output or changing the value contained in an analog holding register, for example, enables the programmer to simulate a particular condition and then test the operation of the ladder program.

If you are not in Network 1, go there now.

To force a register:

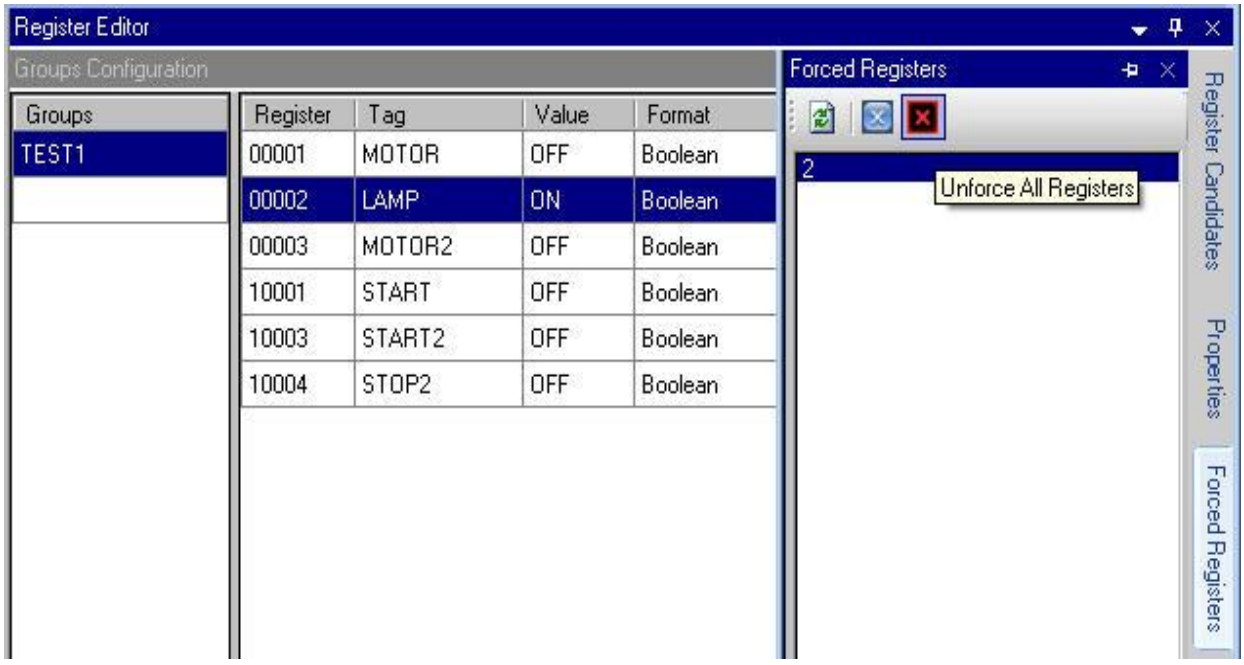
Click on the Monitor button in the Mode group.
In the Register Editor click on the register you need to apply the force to and then click on the properties tab and turn on the force and put your value in the Value area and hit Apply button.
Click off of the properties window and monitor the register you just forced.

Applying a new Value without forcing will work only if the ladder logic program is not writing to the register. Forcing will override any ladder logic programming and the register will remain in forced condition until it is removed. It is a good habit to check for forced registers before you disconnect from the controller.



To remove the force on single or multiple registers:

Edit the register again and change the Force list box to *No* or Select List Forced tab from the Docked view. The Forced Registers dialog is displayed. Highlight the forced register from the list and select Un-Force Selected Registers or Un-Force All registers if you want to remove multiple forced registers. This command is only available in the *Edit On Line* or *Monitor On Line* modes.



Controller Lock

Locking a controller prevents unauthorized access. The controller will reject commands sent to the unit when it is locked. A controller that is unlocked operates without restriction.

The Lock Controller Dialog, see
Figure , specifies a password to be used to lock the controller and the commands that are to be locked.

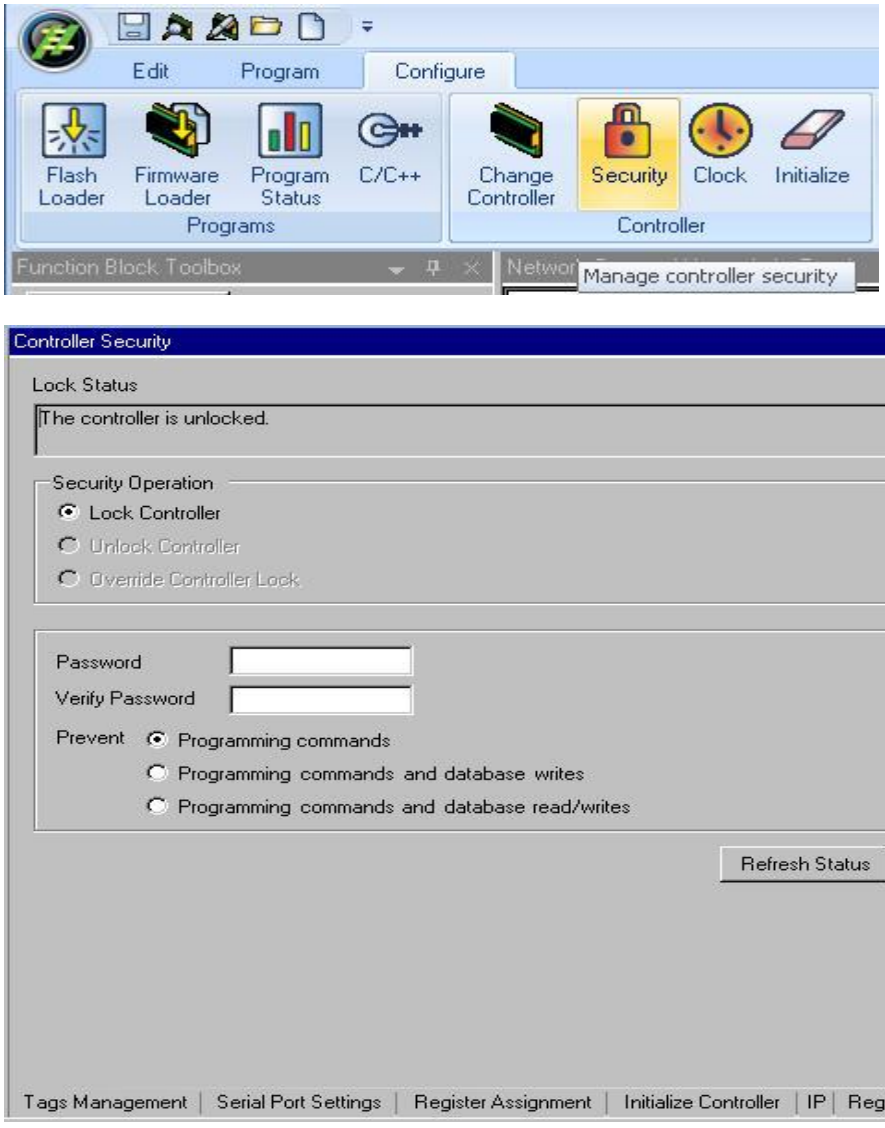


Figure 24: Lock Controller Dialog

Enter a password in the Password edit box. Re-enter the password in the Verify Password edit box. Any character string up to eight characters in length may be entered. Typing in these edit boxes is masked. An asterisk is shown for each character typed. The *Prevent* radio buttons select the commands that are locked.

- Locking the programming commands prevents modifying or viewing the program in the controller. Communication protocols can read and write the I/O database.
- Locking programming and database write commands prevents modifying or viewing the program and prevents writing to the I/O database. Communication protocols can read data from the I/O database, but cannot modify any data.
- Locking programming and database commands prevents modifying or viewing the program and prevents reading and writing the I/O database. Communication protocols cannot read or write the I/O database.

The *Apply* button verifies the passwords are the same and sends the lock controller command to the controller. The dialog is closed. If the passwords are not the same an error message is displayed.

If the controller is already locked, a message indicating this is shown instead of the dialog.

Unlock Controller:

The *Unlock Controller Dialog,* see *Figure* , prompts the user for a password to be used to unlock the controller. If the controller is locked, the following dialog is displayed.

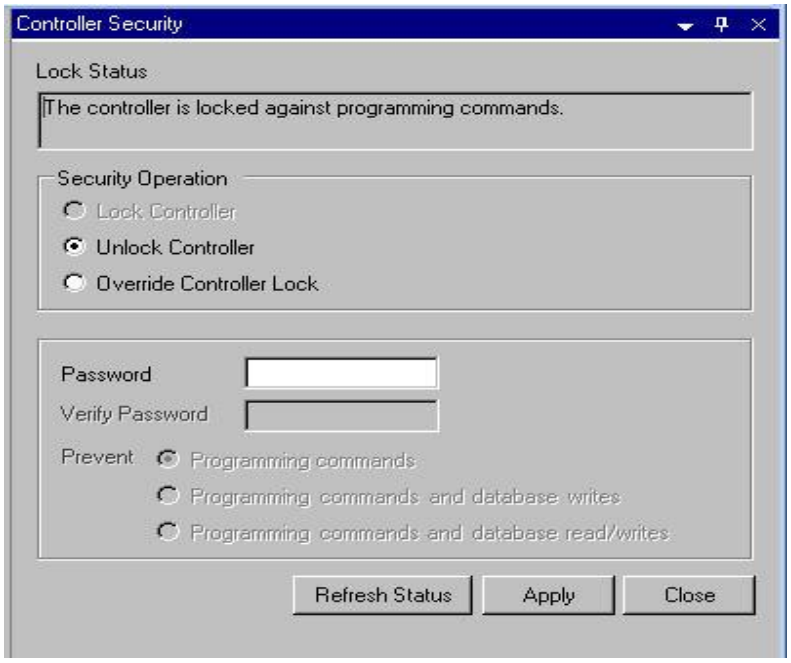


Figure 25: Unlock Controller Dialog

Enter the password that was used to lock the controller in the *Password* edit box. Any character string up to eight characters in length may be entered. . Typing in this edit box is masked. An asterisk is shown for each character typed.

The *Cancel* button closes the dialog without any action.

The *OK* button sends the *Unlock Controller* command to the controller. If the password is correct the controller will be unlocked. If the password is not correct, the controller will remain locked.

Override Controller Lock

The *Override Controller Lock Dialog,* see **Figure 26**, allows the user to unlock a controller without knowing the password. This can be used in the event that that the password is forgotten.

To prevent unauthorized access to the information in the controller, the C and Ladder Logic programs are erased. Use this command with caution, as you will lose the programs in the controller. Selecting the *Override Controller Lock* command displays this dialog.

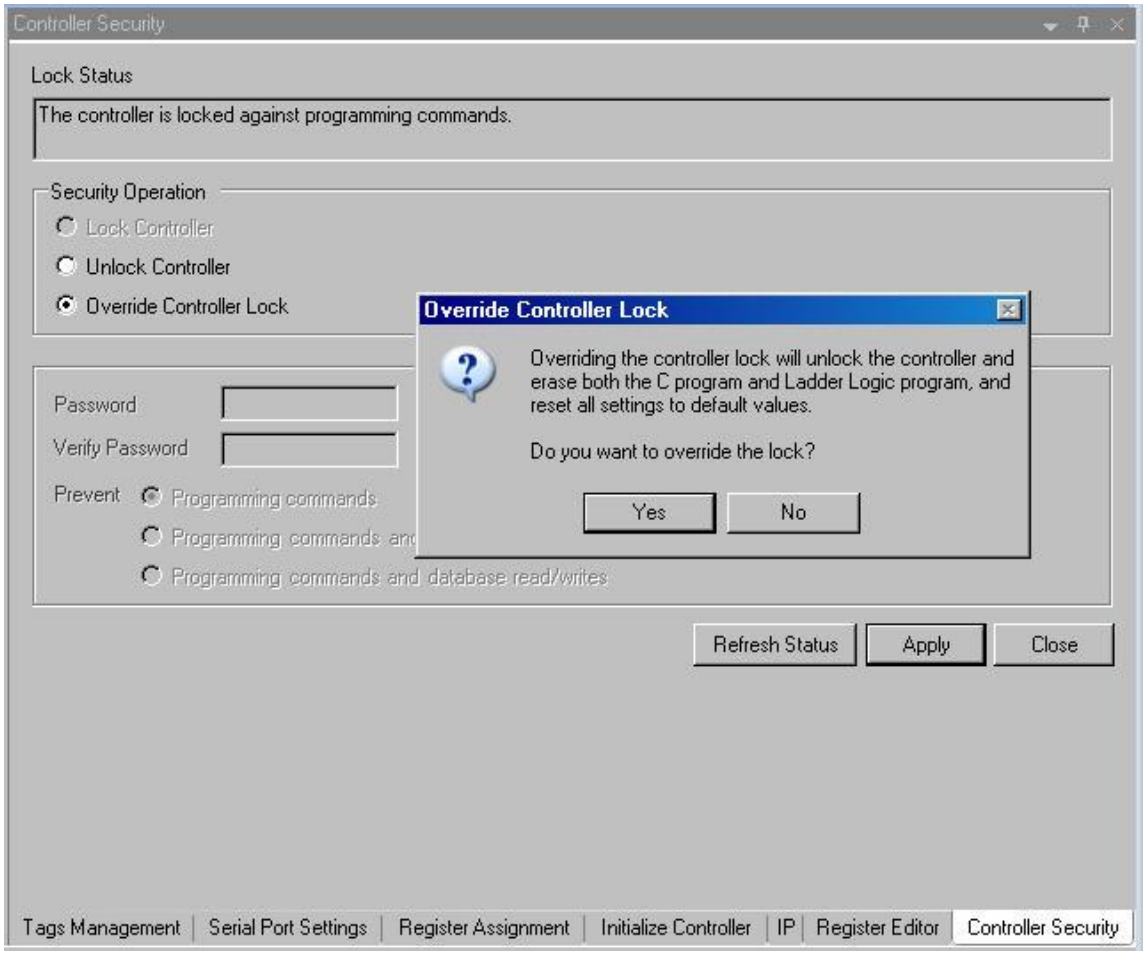


Figure 26: Override Controller Lock Dialog

The *Yes* button unlocks the controller and erases all programs.
The *No* button closes the dialog without any action.

Ladder Logic Functions

The Ladder Logic Functions used in TelePACE programs are summarized below. For details and examples using each function see the *Function Block Reference* of the *TelePACE User Manual* on your PC under the TelePACE Program group of your Start Menu.

Math Functions using Unsigned Registers

ADDU – Add Unsigned Values
CMPU – Compare Unsigned Values
DIVU** – Divide Unsigned Values
MODU – Modulus of Unsigned Values
MULU* – Multiply Unsigned Values
PUTU – Put Unsigned Value into Registers
SUBU – Subtract Unsigned Values

*MULU – result is an Unsigned Double (32 bit register); a 32 bit register = two consecutive 16 bit registers.
**DIVU – input is an Unsigned Double (32 bit register); result is an Unsigned (16 bit register)

Math Functions using Signed Registers

ABS – Absolute Value
ADD – Add Signed Values
CMP – Compare Signed Values
DIV** – Divide Signed Values
MUL* – Multiply Signed Values
PUT – Put Signed Value into Registers
SUB – Subtract Signed Values

*MUL – result is a Signed Double (32 bit register)
**DIV – input is a Signed Double (32 bit register); result is a Signed (16 bit register)

Math Functions using Floating Point Registers

ABSF – Floating-Point Absolute Value
ADDF – Add Floating-Point Values
DIVF – Divide Floating-Point Values
GTEF – Floating-Point Greater Than or Equal
LTEF – Floating-Point Less Than or Equal
MULF – Multiply Floating-Point Values
POWR – Raise Floating-Point Value to a Power
PUTF – Put Floating-Point Value
SUBF – Subtract Floating-Point Values
SQRF – Square Root of Floating-Point Value

Convert Register Type

FTOS – Floating-Point to Signed Integer
FTOU – Floating-Point to Unsigned Integer
STOF – Signed Integer to Floating-Point
UTOF – Unsigned Integer to Floating-Point

Boolean Operations

AND – And Block
Coil
CMPB – Compare Bit
GETB – Get Bit from Block
Normally Closed Contact
Normally Open Contact
NOT – Not Block
One Shot Coil
OR – Or Block
PUTB – Put Bit into Block
ROTB – Rotate Bits in Block
XOR – Exclusive Or Block

Counters and Timers

DCTR – Down Counter
PULM – Pulse Minutes
PULS – Pulse Seconds
UCTR – Up Counter
Timers – 0.01, 0.1, and 1 second Timers

Controlling Block of Registers

L–>L – List to List Transfer
L–>R – List to Register Transfer
MOVE – Move Block
OVER – Override Block of Registers
R–>L – Register to List Transfer

SCADA Application Functions

FLOW – Flow Accumulator
PID – PID Controller
PIDA – Analog Output PID
PIDD – Digital Output PID
SCAL – Scale Analog Value
SLP – Put Controller into Sleep Mode
TOTL – Analog Totalizer

Data Logging

DLOG – Data Logger
FIN – FIFO Queue Insert
FOUT – FIFO Queue Remove
GETL – Data Logger Extract

Communications

DIAL – Control Dial-Up Modem
HART – Send HART Command
INIM – Initialize Dial-Up Modem
MSTR – Master Message
MSIP – Master TCP/IP Message

Program Execution

CALL – Execute Subroutine
SUBR – Start of Subroutine

Using Keyboard Shortcuts

The following function keys or control key shortcuts are available:

Ctrl-A	**Save As File**
Ctrl-S	**Save File**
Ctrl-O	**Open File**
Ctrl-Q	**Exit**
Ctrl-Z	**Undo**
Ctrl-Y	**Redo**
Ctrl-X	**Cut Selected**
Ctrl-C	**Copy Selected**
Ctrl-V	**Paste**
F3	**Repeat Last Find**
F5	**Toggle Vertical Shunt**
F6	**Insert Horizontal Shunt**
F7	**Previous Network**
F8	**Next Network**
Delete	**Delete Menu**
Insert	**Insert/Edit Element**
PgDn	**Scroll Down one page**
PgUp	**Scroll Up one page**

Pump Control Example

In this section of the TelePACE Studio Ladder Logic training course a practical example of a simple lead / lag pump control program will be developed. A program will be developed from a very simple single pump start / stop switch control into a more complex two pump automatic lead / lag control system.

The development of the lead / lag pump control is divided into five parts. Each part of this section emphasizes one or more function blocks that are available for use when programming the Controller using the TelePACE Ladder Logic.

In Part One of this section the Controller is programmed to use the Series 5000 I/O as a means of controlling the turning on and off of a digital output. Digital inputs and then analog inputs are used to turn the output on and off. This process simulates the starting and stopping of a pump connected to the digital output.

In Part Two math functions are used to scale the analog input value used to control the pump operation. The types of numbers used in the TelePACE Studio Ladder Logic are explained and the use of an initialization network is introduced.

In Part Three counter and timer functions are added to the program to store the accumulated pump run time and to store the number of pump starts.

In Part Four of this section adds a lag pump to the program and a lead / lag pump control program is developed.

The last section involves using an automatic switching sequence to control the lead / lag pump operation in the program.

Part One: Digital Output Control

The first step in the program development will be to create a Ladder Logic program that is similar to the Motor.lad program.

Step 1 Create tag Names

Open a new file in the TelePACE Studio and enter the Tag names and registers in the Tags Management View.

Tag Name	Address/Constant	Type
AIN 0 0_100%	42202	Register 4nnnn
Analog Input 0	30001	Register 3nnnn
Lead Start	1059	Register 0nnnn
Lead Start Setpoint	43000	Register 4nnnn
Lead Stop	1058	Register 0nnnn
Lead Stop Setpoint	43001	Register 4nnnn
Multiply Low Word	42200	Register 4nnnn
Power Up Reset	1057	Register 0nnnn
Pump 1	1	Register 0nnnn
Pump 1 Hi Minutes	42207	Register 4nnnn
Pump 1 Low Minutes	42206	Register 4nnnn
Pump 1 Run Hours	42210	Register 4nnnn
Pump 1 Start	10001	Register 1nnnn
Pump 1 Starts	41224	Register 4nnnn
Pump 1 Stop	10002	Register 1nnnn
Pump 1 Timer Reset	1071	Register 0nnnn

Step 2 Create Pump Start Program

The Pump Start program is very similar to the program created in the Using the TelePACE Ladder Editor section of the course.

Insert the elements as shown in
Figure 27: Pump Start Program.

Save the Ladder Logic program as TRAIN001.tpj

Load the program into the Controller and monitor the program execution.

The output coil 00001 can be turned on and off using the input switches 10001 and 10002. Using digital inputs to control digital outputs is a very common occurrence when using relay ladder logic. Pumps are usually started and stopped based on the level of the tank or reservoir they are being used to fill.

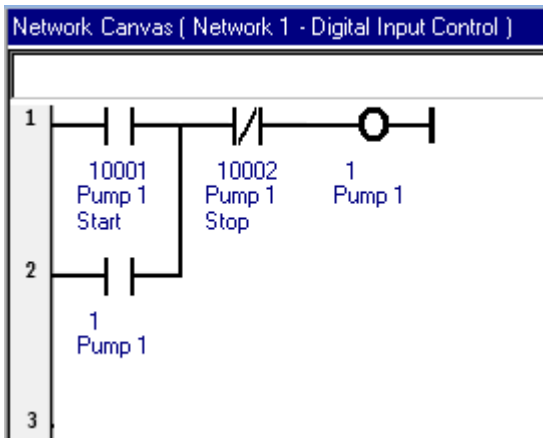


Figure 27: Pump Start Program

Step 3 Analog Input to Start and Stop the Pump

In order to use an analog input to control the starting and stopping of the digital output the value of the analog input must be compared to some set point value. The TelePACE Studio Ladder Logic functions used for comparing values are the CMP and CMPU. The CMP function is used to compare signed values and the CMPU is used to compare unsigned values.

Delete the elements that were inserted in step 2.

Select the Help menu and then select the CMPU function from the function block reference section. The Compare function allows two registers or constants to be compared and an output from the function that is the result of the comparison.

This function will be used to compare the raw value of analog input 0 with a constant representing one half of the maximum raw value of analog input 0. This constant value is referred to as the set point.

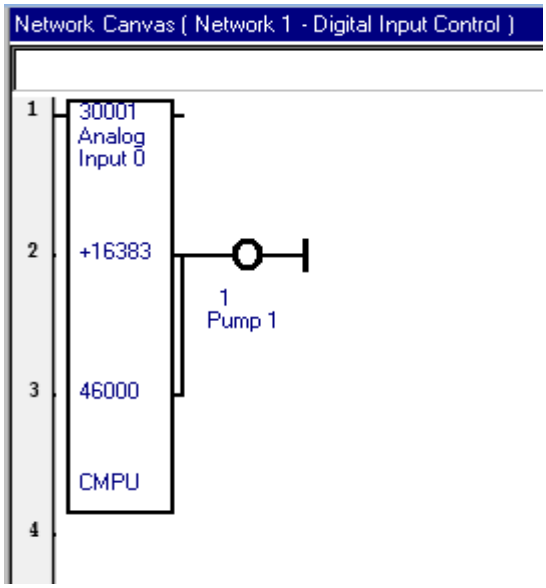


Figure 28: CMPU Function Block

? What is the range of a raw analog input on the SCADAPack 5601 I/O module?

Select the CMPU function from the function list and configure the function as shown in Figure 28.

? What is the function of the vertical shunt used on the CMPU output?

? What is the limitation of using this type of control?

Save the Ladder Logic program as TRAIN002.tpj and load the program into the Controller and monitor the program execution.

Step 4 Adding Hysteresis to the Pump Control

Generally a pump will not be started and stopped at one analog input level. The pump will continually be starting and stopping as the level rises above and falls below the set point. Hysteresis is simply having one set point to start the pump and another set point to stop the pump.

Examine the network shown in
Figure 29: Hysteresis Example and answer the following questions.

? What do the two numbers 29490 and 22937 represent?

? What type of coils are 1058 and 1059?

? At what analog input level will the coil 01059 be energized?

? At what analog input level will the coil 01058 be de-energized?

Modify the program as shown in
Figure 29: Hysteresis Example.

Load the program into the Controller and monitor the program execution.

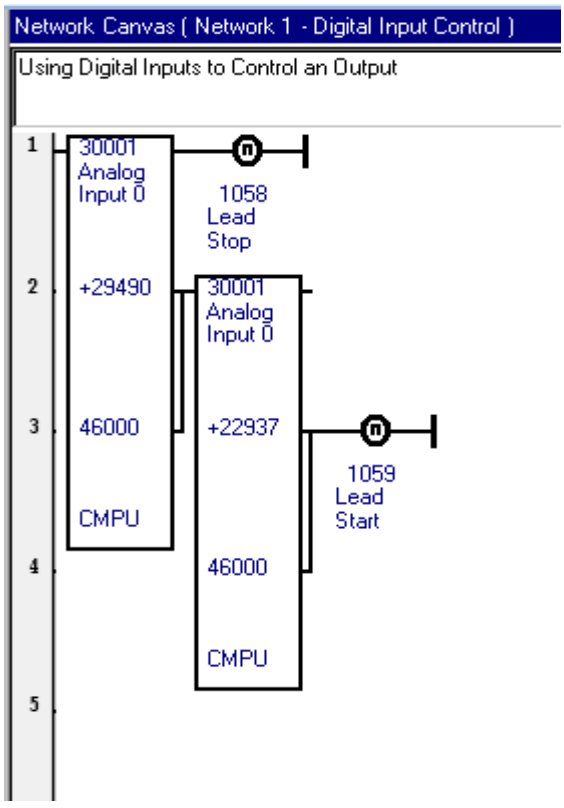


Figure 29: Hysteresis Example

Save this program as TRAIN003.tpj

Step 5 Pump Control Using Hysteresis

The limitations of the above example are quickly seen when the program is monitored on line. To overcome these limitations and to control the pump start and stop the program needs to be modified to that shown in Figure 30

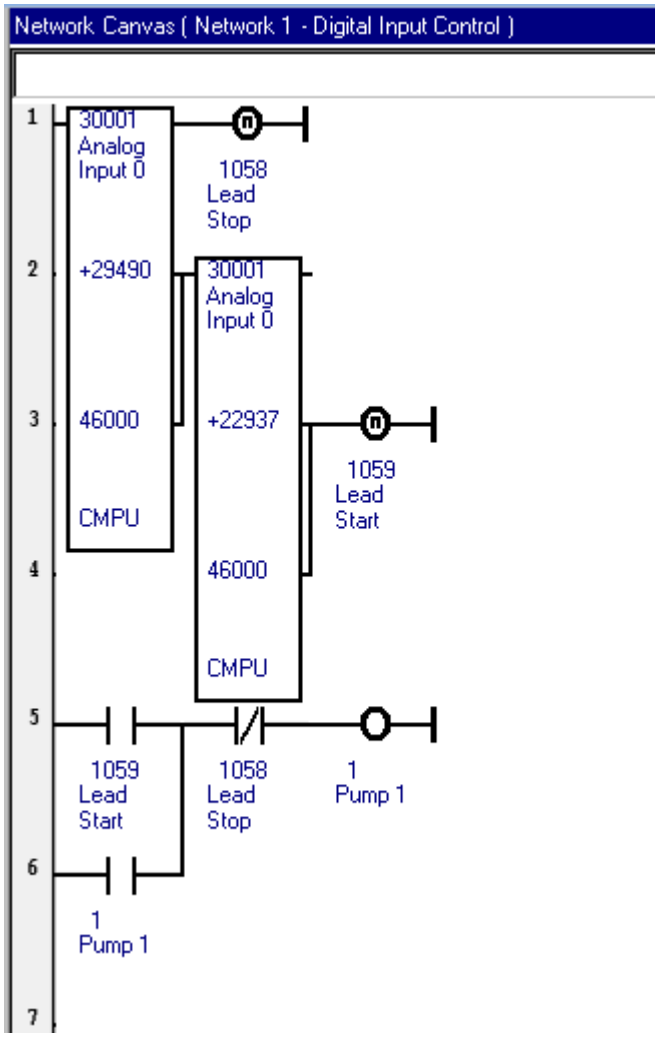


Figure 30: Complete Hysteresis

Save this program as TRAIN004.tpj.

Part Two: Scaling Analog Inputs

In this part of the course logic will be added to the program to scale the analog input, register 30001. A field Transmitter is commonly used to convert a process value, temperature, pressure, level etc., into voltage or current level. This level is input to a 5000 Series analog input module. The current or voltage value at the analog input is analog to digital (A/D) converted into a raw I/O count of between 0 and 32767

.

The Ladder Logic program running in the controller often must convert the raw I/O count back into the units of measurement of the process value, or to a percentage of the maximum input. This process is referred to as scaling the analog input.

Before the program can be modified to scale the analog input, the way in which decimal numbers are represented in the TelePACE Ladder Logic needs to be understood.

Representing Numbers in the Controller:

The types of numbers used in TelePACE Studio are called integers or fixed point numbers. Integers are numbers that do not have decimal places and are commonly called whole numbers. Integer numbers can be positive or negative.

The controller stores integer numbers in 16 bit words. Each bit in the word is either a 1 or a 0 and the combination of 1's and 0's represent the binary equivalent of the integer number stored in the word. The bits in a word are numbered from 0 to 15 starting at the right side of the word. Bit 0 is referred to as the least significant bit and bit 15 is the most significant bit. Each bit in a word has a binary and a decimal value as shown in the following table:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
327	163	81	40	20	10	51	25	12	64	32	16	8	4	2	1
68	84	92	96	48	24	2	6	8							

Using the above table it can be seen that if all 16 bit positions are 1's the word will have a value of 65535. This is the largest number that a single word can contain.

Each 16-bit word can contain an unsigned integer or a signed integer. Unsigned integers are positive and have the range of 0 to 65535. Signed integers may be positive or negative. When a word contains a signed integer the most significant bit, bit 15, is used for the sign extension, positive or negative. This leaves 15 bits to represent signed integers. Signed integers have a range of -32768 to +32767.

The 16 bit words in the controller are used to store constants, input registers, output registers and blocks of 16 status and coil registers.

? What would the value be for a register with the following bit combination?

0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1

? What would the bit structure be for the integer value 65535?

? What is a limitation of using 16 bit words?

Double Words

Often the result of a multiplication function is much larger than the maximum value of 65535 available using a single word. The same is true when a divide function is used, the number to be divided is often larger than 65535.

The TelePACE Ladder Logic uses double words to store the result of a multiplication function, and as the value to be divided in a divide function. A double word is simply two sequential 16 bit words. The largest value that can be stored using a double word is 4,294,967,295.

Scaling Analog Input 30001

The training.lad program will be modified to scale the raw analog input, register 30001, into a percentage of the maximum analog input value, 32767. The calculation used for the scaling will be:

$$\frac{\text{Value of Input Register 30001} \times 100\%}{\text{Maximum Value of Input Register 30001}}$$

Open the Ladder Logic program saved in Part One.

Insert a network before network 1.

In this exercise we are going to use three additional function blocks the SUBU, MULU, and DIVU.

Step 1 Subtract Input Register by 6553 – 4mA = 0

There are three subtraction functions that are available in the TelePACE Ladder Editor. Since the value of input register 30001 will always be between 0 and 32767 the SUBU function can be used.

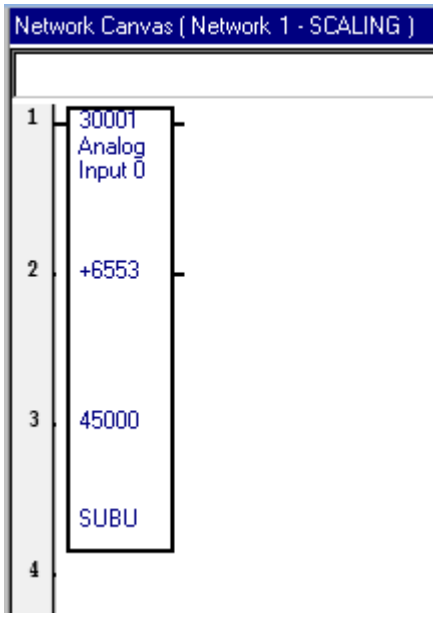


Figure 31: Insert / Edit MULU Function

Step 2 Multiply Input Register by 100

There are three multiplication functions that are available in the TelePACE Ladder Editor. Since the value of input register 30001 will always be between 0 and 32767 the MULU function can be used.

Complete the Dialog box as shown in Figure 32: Insert / Edit MULU Function.

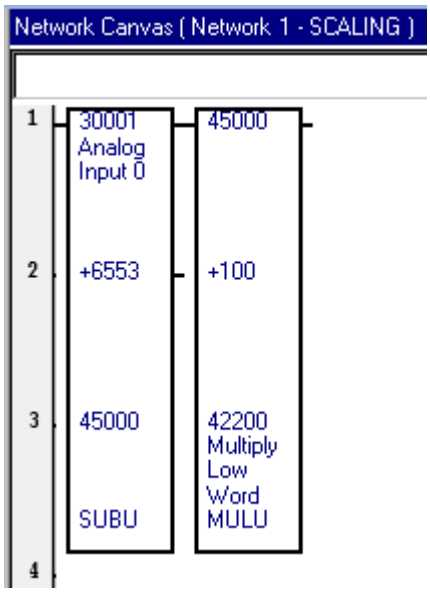


Figure 32: Insert / Edit MULU Function

Step 3 Divide the result by 32767

There are three division functions that are available in the TelePACE Ladder Editor, the DIV function for dividing signed numbers and DIVU function for dividing unsigned numbers and the DIVF for floating points. Since the result of the multiplication in step 1 will always be positive the DIVU function can be used.

The completed network should now look like the network in
Figure 33: Completed Scaling Network.

Load the modified program into the controller and run it.

Using the Monitor On line function monitor registers 30001 and 42202.
Adjust the input pot for different values in register 30001.

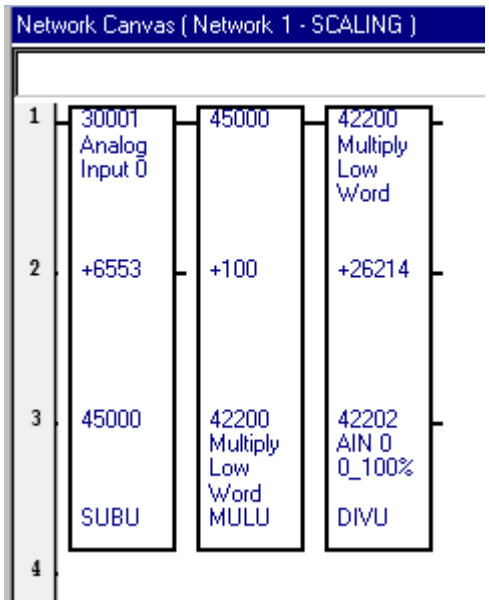


Figure 33: Completed Scaling Network

To increase the resolution of the scaling calculation, change the constant +100 in the second value to multiply in the MULU function to +1000. This change will cause the value in register 42202 to have a range of 0 to 1000. This range will represent a scaled value of 0.0 to 100.0%.

Modify the MULU function for a constant of +1000 instead of +100.

Modify the CMPU function in network 2 to use the scaled input values as shown in figure 33A.

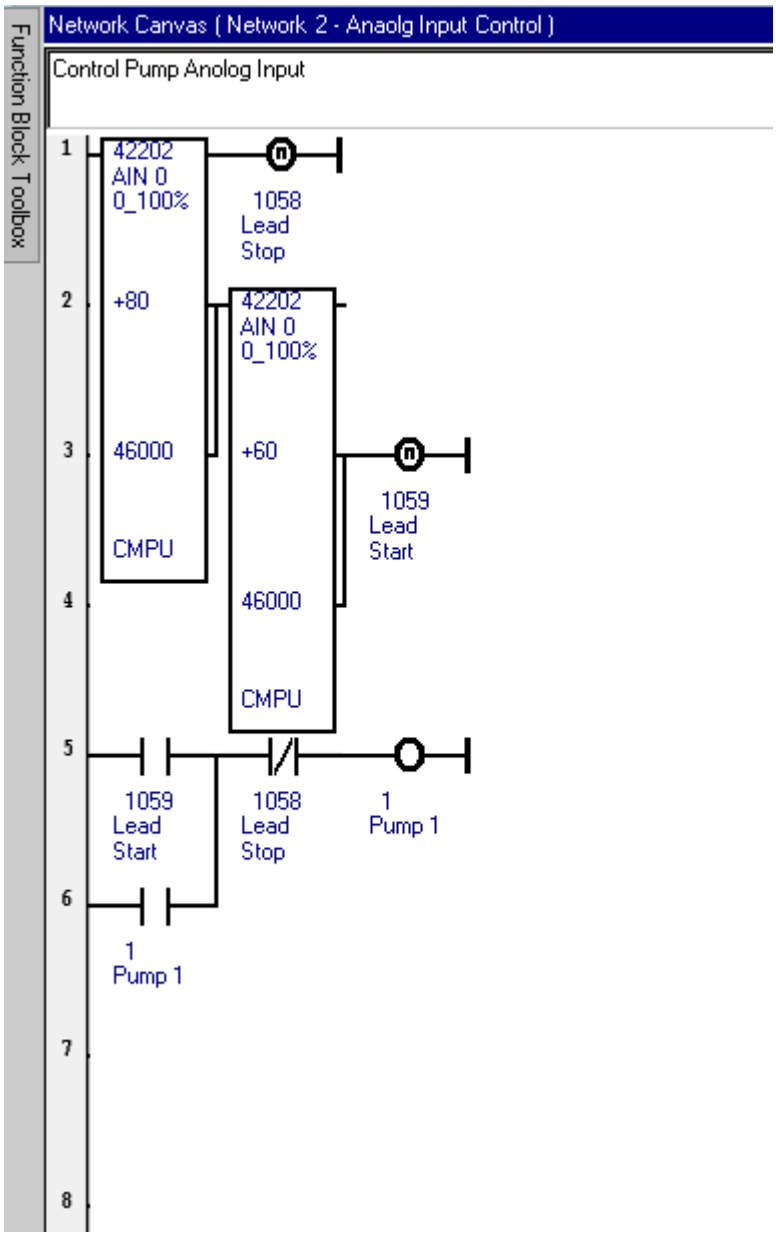


Figure 33A: Modified Set Point Network

Save the Ladder Logic program as TRAIN005.tpj

Step 4 Change Set Points to Registers from Constants

In this step the start and stop set point values in network 2 will be changed from constants to analog output registers. When a constant is used in a Ladder Logic function, it can only be changed by editing the function using the TelePACE Ladder Editor. To enable the start and stop Set points to be changed through the I/O Database an analog output register must be used.

? What is the advantage of changing these set points through the I/O Database?

Modify the network 2 as shown in
Figure 34: Modified Set Point **Network**.

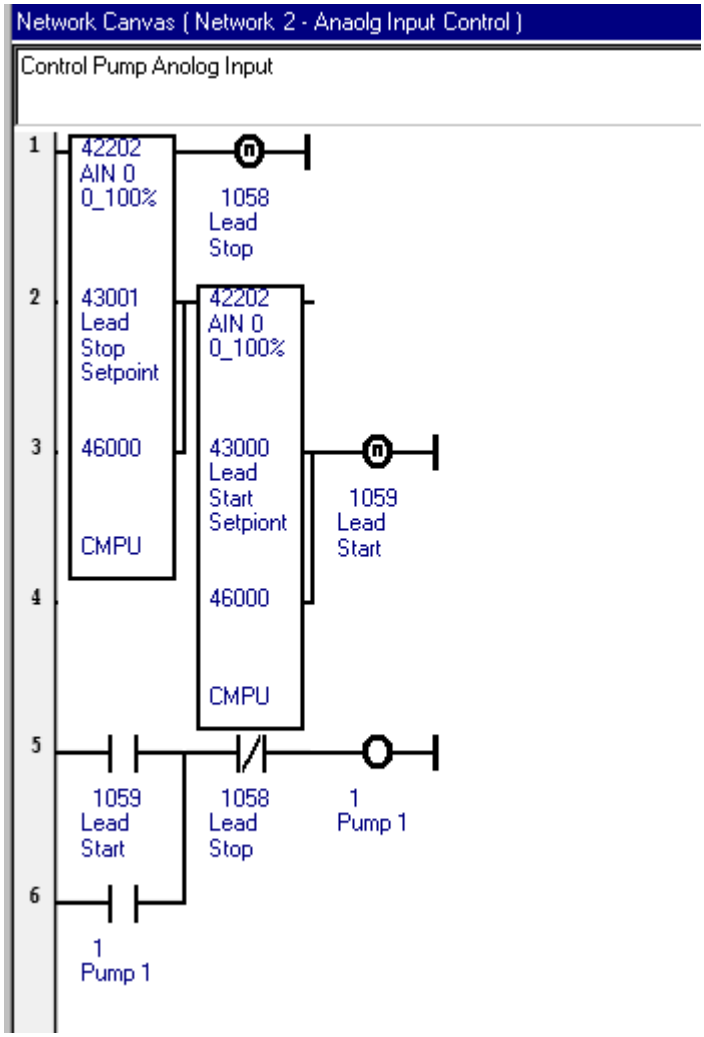


Figure 34: Modified Set Point Network

Step 5 Put start up values into the set point registers

The TelePACE Ladder Logic functions that are used to put values into registers are the PUT and PUTU functions. The PUT function is used for signed numbers and the PUTU function is used for unsigned numbers.

In this program the set point values are stored in analog output registers so that they may be modified through the I/O Database. For this reason the Ladder Logic program should put values into the start and stop set point registers when the program initially starts.

? Why would the set point values only be put into the analog output registers when the program initially starts?

Go to network 1 of the program and insert a network before network 1.
Insert the elements as shown in
Figure 35: One Shot Coil and Contact. Note that the one-shot coil is inserted in row 1 column 1.

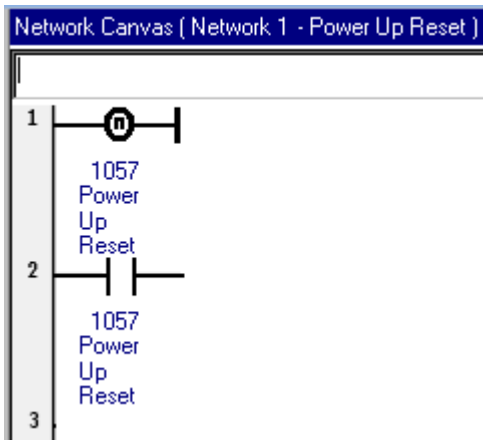


Figure 35: One Shot Coil and Contact

To the right of the contact 01057 insert a PUTU function. Configure the function as shown in *Figure 36: Insert / Edit PUTU Function*.

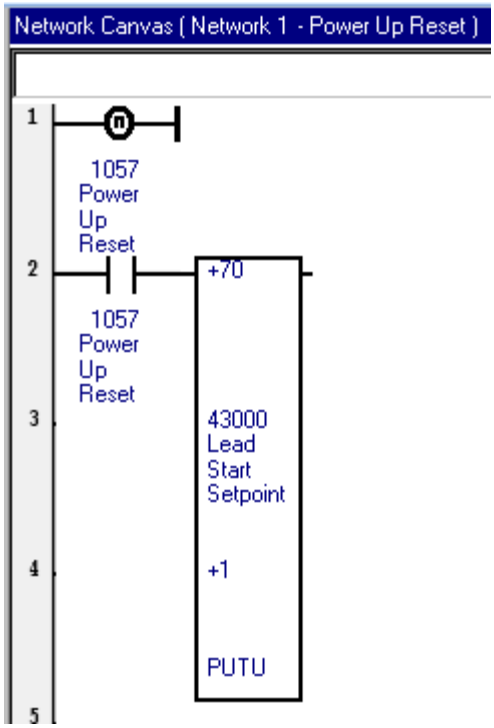


Figure 36: Insert / Edit PUTU Function

? Why is the value 70 used as the source register?

? When will this value be PUTU into register 43000?

Insert another PUTU function for the stop set point register. The network should look as shown in *Figure 37: Completed Start Up Network*.

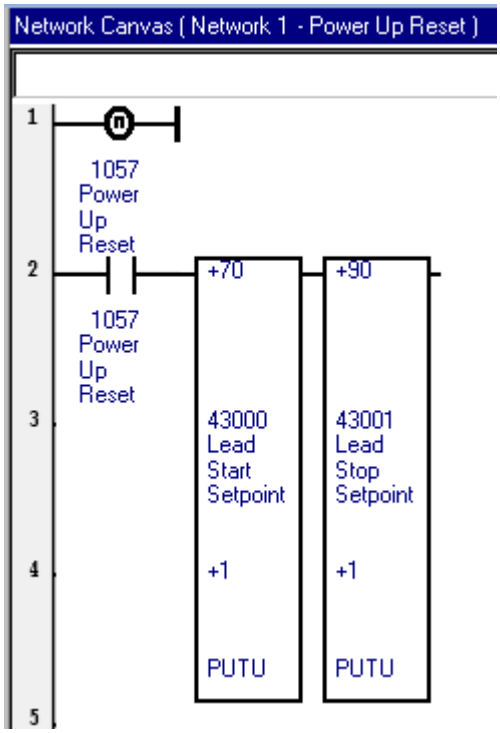


Figure 37: Completed Start Up Network

Save the Ladder Logic program as TRAIN006.tpj

Part Three: Run Timer, Start Counter

In this part of the training course the program is going to be modified to add a pump run timer and a number of starts counter for Pump 1.

Step 1 Setup a One Minute Timer

For maintenance purposes it is useful to know how many hours a pump has run. The first step in creating a run timer for the pump is to create a timer that will run in one minute intervals. To do this a T1 Timer function will be used.

Display the Timer function in the TelePACE Ladder Editor Help.

The graphic of the T1 timer in the Help screen shows that it has four connections, two on the left side and two on the right side. This graphic is a representation of how the Time function appears in the Ladder Editor. The connections on the left of the block are referred to as inputs and the connections on the right side of the block are referred to as outputs.

? What connections to the timer are required for the accumulator register to increment?

? How is the Timer enabled for timing?

? When the Timer is enabled what is being timed?

? In this program what would the limit variable contain?

? How is the Timer reset?

Insert a network after network 3.

Complete the network by inserting the elements as shown in Figure 39: .

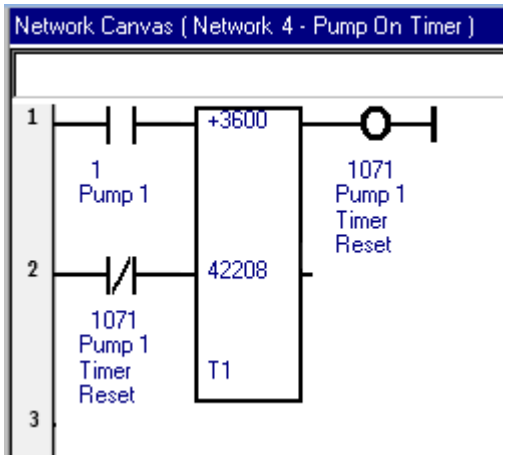


Figure 39: Network Timer

Save the program as TRAIN007.tpj

Write the program to the controller. Adjust the analog input such that coil 00001 is turned on. Use the Monitor On line function and monitor register 42208.

? When is coil 1071 energized?

? Why is contact 1071 a normally closed?
Adjust the pot simulating the tank level such that coil 00001 is turned off.

? What happens to the value in the accumulator register when coil 00001 is turned off?

? What happens to the value in the accumulator when the program is stopped and then restarted?

Step 2 Accumulate the Pump On Time

In this step the 1 hour intervals will be accumulated. There are several ways to accumulate the intervals. In this program an UCTR function will be used to demonstrate how a totalizer can be constructed using the TelePACE Ladder Logic.

? What are some other ways that the intervals could be accumulated?

To accumulate the number of hours that the pump is on each hour output from the timer will be added to a register. The value in the register used will increment by one on each hour count.

From network 4, insert a network after.

Insert an UCTR function to the right of the Timer function, between the Timer function and coil 01071.

Configure the UCTR function as shown in Figure 41: Insert / Edit Up CTR Function.
The network should appear the same as shown below.
Error! Reference source not found.

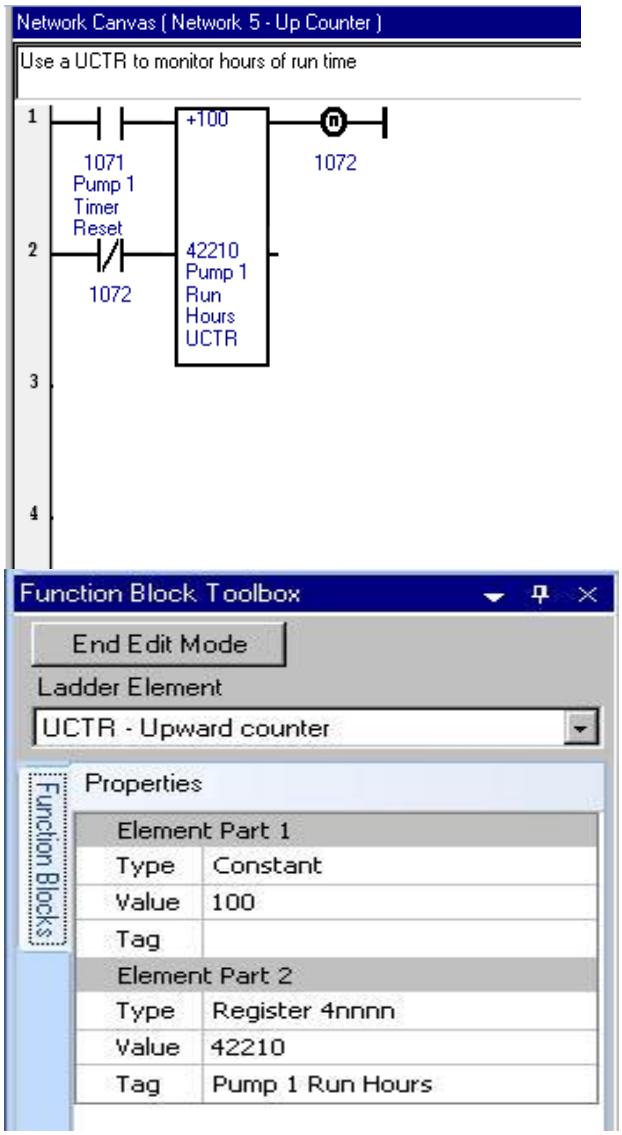


Figure 41: Insert / Edit Up CTR Function

Save the program as TRAIN008.tpj Load the program into the controller and monitor the operation of the program. Using the Monitor On line function monitor the accumulator register.

Part Four: Add Lag Pump Control

This exercise will be built by the student.
After ample time, the class will go over each network together.
Good Luck!

In parts one, two and three of this programming example the training.lad program has been developed to control the operation of a pump.

In the program developed so far:
The pump starts when the analog input 30001 value is 70.0% of maximum.
The pump stops when the analog input 30001 value is 90.0% of maximum.
The run time of the pump is measured and accumulated.
The number of times the pump starts is saved.

In many applications using pumps it is often required to have a second pump to back up the first pump if the first pump cannot maintain the set point. In this configuration the pump that is started first is the lead pump and the backup pump is the lag pump. This type of control is called Lead / Lag pump control.

In this part of the training course the training.tpj program is going to be modified to add Lead / Lag pump control. The sequence for starting and stopping the Lead and the Lag pumps will be:
Stop Lead Pump:
When the value of analog input register 30001 is equal to or greater than 90.0% of the maximum value.
Stop Lag Pump:
When the value of analog input register 30001 is equal to or greater than 80.0% of the maximum value.
Start Lead Pump:
When the value of analog input register 30001 is equal to or less than 70.0% of the maximum value.
Start Lag Pump When the value of analog input register 30001 is equal to or less than 60.0% of the maximum value.

Save the tested program as TRAIN009.tpj

Part Five: Alternating Lead / Lag

In this last section of the TelePACE Ladder Logic programming example the program is modified to switch the lead and lag pumps every 50 hours. When a pump is selected for lead operation it will run as the lead pump for 50 hours and then the lag pump will become the lead pump and it will run for 50 hours. The pumps will cycle as lead and lag, every 50 hours, continuously.

To begin this part of the programming example the Ladder Logic program TRAIN010.tpj has been created. In this program the lead and lag run timers have been changed to increment the accumulated minutes registers every 2 seconds. The lead and lag switching will occur every 10 seconds.

Open a new file in the TelePACE Ladder Editor.
Initialize the I/O Database in the controller.
Load the program TRAIN010.tpj into the controller.
Run the program and monitor the program execution.
Adjust the analog input to 65% and monitor program execution.
Adjust the analog input to 50% and monitor program execution.
Adjust the analog input to 85% and monitor program execution.
Using the tools available determine how the program works.
The instructor will explain the program execution after the class has had the opportunity to thoroughly examine the program.

Save the Ladder Logic program as TRAIN010.tpj

Pat yourself on the back for a job well done!

Implementing Subroutines in TelePACE

A subroutine is a group of ladder logic networks that may be executed conditionally. Each subroutine begins with a network containing an SUBR element on its first rung. A subroutine ends when another subroutine element is encountered, or when the end of the ladder logic program is reached. In order to use a subroutine, a CALL element with the same subroutine number is enabled by the program.

Ladder logic programs can grow to include many networks, causing the scan time to become longer than may be desired. The use of subroutines allows the programmer to decide when any part of the logic will be executed. One subroutine might need to be executed once per second, while another only once per hour. This can dramatically reduce the controller's scan time.

If a given piece of code needs to be executed repeatedly during a single scan cycle, one subroutine may be called repeatedly with multiple CALL elements instead of wasting memory on multiple instances of the logic. With only 12k words available for ladder logic, this can become important.

Digital and analog outputs that are set in a subroutine remain in their last state when that subroutine is not called. For example, a coil that is turned on by a subroutine remains on when the subroutine is disabled. The output will only turn off when the subroutine is called and it turns the output off.

Subroutines do not have to be programmed in any particular numerical order. For example, subroutine 1 can follow subroutine 2, subroutine 200 can follow subroutine 400.

A subroutine can call other subroutines. This is called nesting. The maximum level of nesting is 20 calls. In practice, more than two or three levels of nesting can become quite confusing.

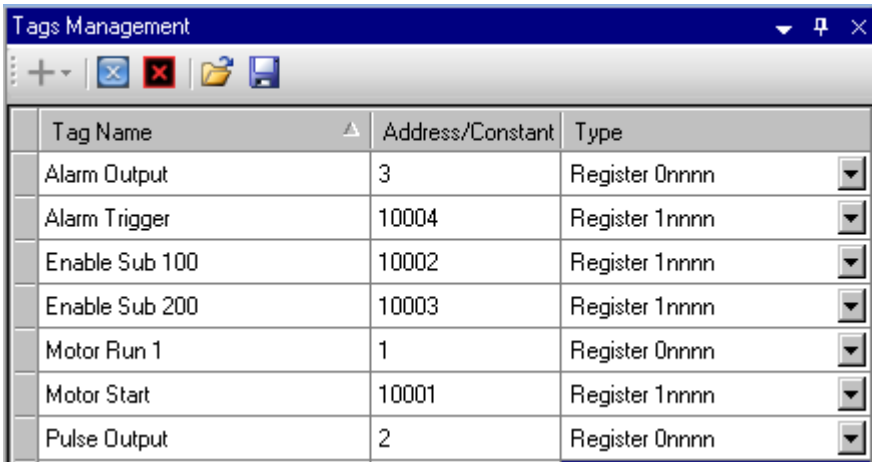
Subroutine calls cannot be recursive. For example, subroutine 1 cannot call itself or call another subroutine that calls subroutine 1. This prevents potential infinite loops in the ladder logic program.

The following exercise will provide a simple demonstration of the use of a subroutine in a program. It will then be expanded to include a second nested subroutine.

Step 1 Create Tag Names

Open a new file in TelePACE.

Enter the following tag names using the Edit Tag Name dialog or import them from the supplied SUBRDEMO.CSV file.



Tag Name	Address/Constant	Type
Alarm Output	3	Register 0nnnn
Alarm Trigger	10004	Register 1nnnn
Enable Sub 100	10002	Register 1nnnn
Enable Sub 200	10003	Register 1nnnn
Motor Run 1	1	Register 0nnnn
Motor Start	10001	Register 1nnnn
Pulse Output	2	Register 0nnnn

Step 2 Create Main Program and Subroutine

The main program will simply allow the user to control a motor with a switch. The main program could be many networks in length, but this short example will suffice.

Select the appropriate Controller Type.

Create a Default Register Assignment with the appropriate lower I/O board.

Insert the single rung of main program logic shown below, along with the CALL in rung 2.

Note that a warning is generated when the CALL is inserted before the matching SUBR. In an offline edit this is acceptable. In an online edit, however, it is not allowed to insert the CALL first as a jump to a non-existent subroutine could cause the controller to lock up.

Note that the network type is “Main.”

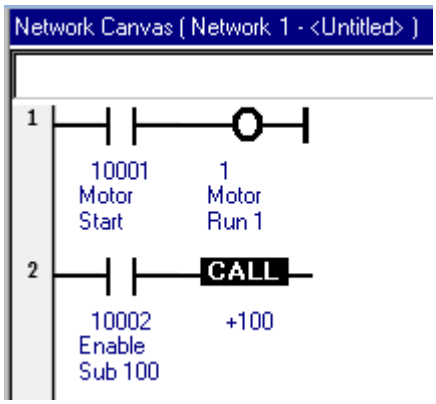


Figure 43: Main Program Control Network

Go to Edit –Network - Network After from the current network.

Note that this network's type is currently "Main."

Insert the logic as shown in Figure 44.
Note that the network's type is now shown to be "Sub 100."

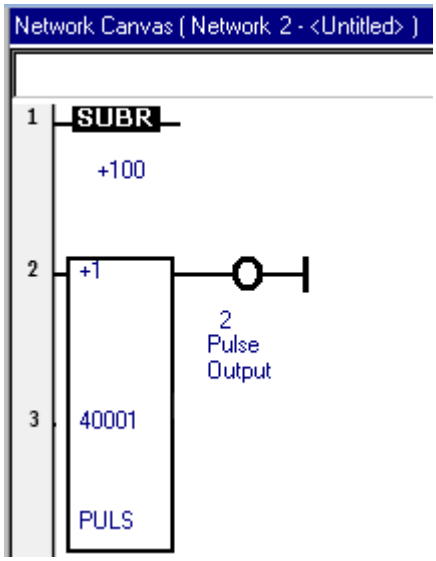


Figure 44: Subroutine 100 Network

Save the program as SUBRDEMO.tpj

Download the program to the controller and monitor program operation.

Note that the main program logic will execute at all times. (ie switch 1 should always be able to control the first LED)

Note that the second LED will only flash when Subroutine 100 is enabled with switch 2.
See that the power rail and SUBR label only turn red (active) when the subroutine is enabled.

Note that if the subroutine is disabled (switch 2 turned off) while the second LED is on, that LED will remain on until the subroutine is again enabled and the PULS is activated.

Step 3 Create a Nested Subroutine

Go to Edit –Network – Add Network After network two.

Note that the network type states “Sub 100” at this time.
Place an SUBR instruction in rung 1 of the new network, which now names this subroutine to be “Sub 200.”

Add a fourth switch controlling the third LED, as shown in Figure 45.
Go back to the first subroutine in network 2, and add a second CALL instruction controlled by a switch, as shown in Figure 46.

Note that the warning seen when the previous CALL was inserted did not occur this time. This is because the subroutine being referenced did exist before the CALL was added.
Download the program to the controller and go online to monitor execution.

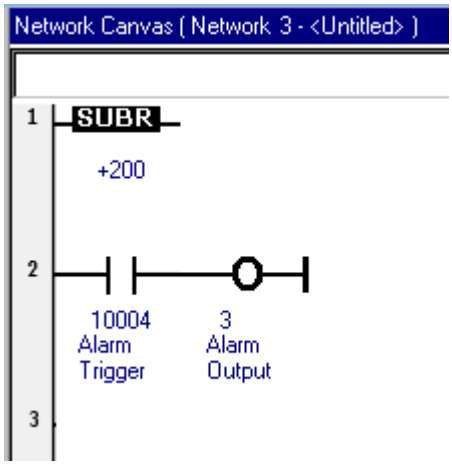


Figure 45: Subroutine 200 Nested Network

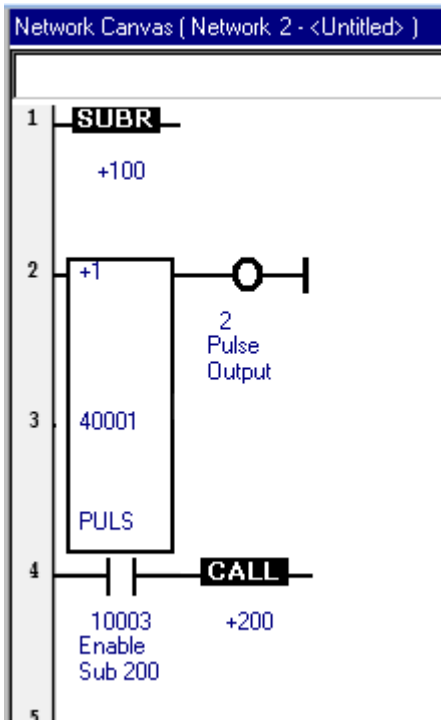


Figure 46: Subroutine 100 Network with CALL

Save the program as SUBRDEMO2.tpj.

Note that the main program logic still works, as it always will.

With Subroutine 100 disabled (switch 2 turned off), note that turning on the third switch does not cause Subroutine 200 to be executed. This is normal. Subroutine 100 is not being called, and therefore Subroutine 200 will not operate either.

Turn on switch 2, which enables Subroutine 100. Notice that the second LED flashes.

Turn on switch 3, which now enables Subroutine 200. Note that the power rail and SUBR label in the subroutine only now become red. (active)

Only at this time should switch 4 be able to turn on the third LED, as both subroutines are now enabled.

Note that if either or both of switches 2 and 3 are turned off, Subroutine 200 will be disabled. LED 3 will remain fixed in its current state until the subroutine is reactivated.

Flow Accumulation

The FLOW function accumulates flow from pulse-type devices such as turbine flow meters. This function is designed to be used in the measurement of products that do not require a complex flow calculation. If any correction factors must be applied, such as for temperature or pressure, that must be done in separate logic in the controller.

The only correction that is applied is the K Factor. This is a value supplied by the manufacturer of the meter, which specifies the number of pulses per unit of product. For example, “103.45 pulses per gallon.”

The turbine meter or other pulse source is wired to a counter input on the SCADAPack. An appropriate entry must be added to the Register Assignment, which provides two registers for a double unsigned integer (32 bit) value to store the pulse accumulator. Incoming pulses are detected by the counter circuitry, and each time a pulse is seen the count value is incremented by one.

The pulse accumulator register is monitored by the FLOW function, which divides by the K Factor to calculate the current flow rate. This is done on each scan of the controller logic. The flow rate updates once per second if there is sufficient flow. If the flow is insufficient, the update slows to as little as once every ten seconds to maintain resolution of the calculated rate.

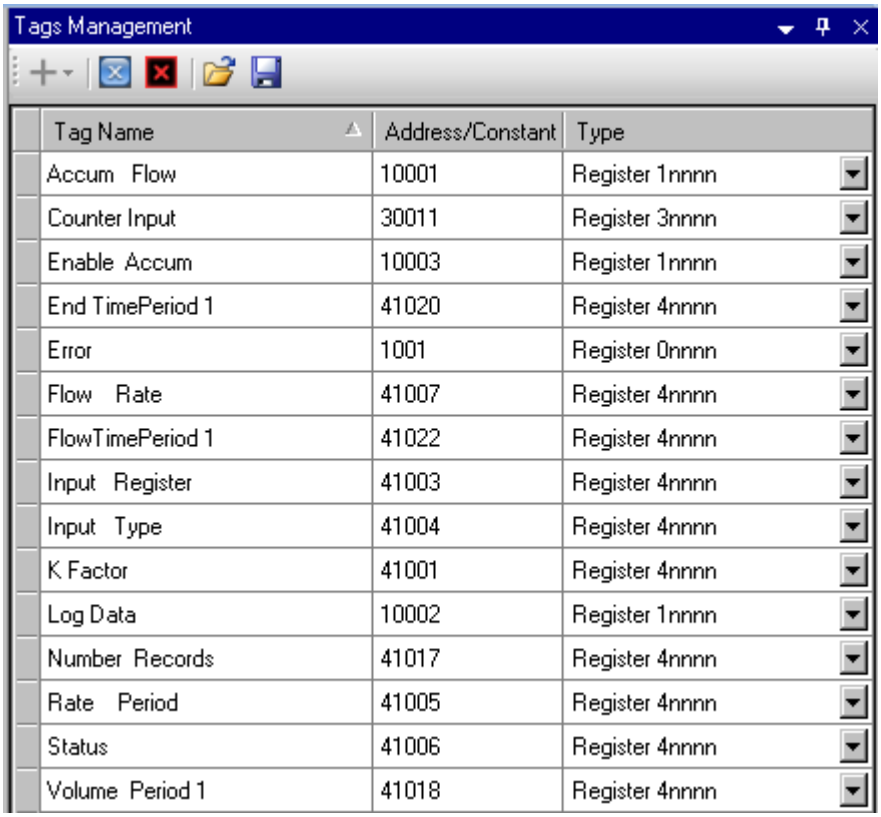
The FLOW function then accumulates the measured flow over a period of time. When the Log Data input goes from OFF to ON, the accumulated volume, flow time and the time at the end of the period is saved in the history registers. Older history is pushed down and the oldest record is discarded. The Log Data input must be triggered at least once every 119 hours (at the maximum pulse rate). Otherwise the volume accumulator will overflow, and the accumulated volume will not be accurate.

Typically the Log Data input will be toggled once per day, to capture the accumulated flow during that day. It can, however, be toggled at any desired time if the program is accumulating flow for other purposes. There are a maximum of 35 sets of history records, for 35 days or periods of accumulation.

In the following exercise, the turbine meter will be simulated by a pulse generator on a 5699 demonstrator I/O board. If the 5699 board is not available, the class may instead add a T.01 timer with a Limit of 65535 to the program. This timer will need to be self-resetting. The Accumulator register may then be monitored as a 16 bit accumulator.

Step 1 Create Tag Names

Open a new file in TelePACE.
Enter the following tag names using the Edit Tag Name dialog or import them from the supplied FLOWDEMO.CSV file.



Tag Name	Address/Constant	Type
Accum Flow	10001	Register 1nnnn
Counter Input	30011	Register 3nnnn
Enable Accum	10003	Register 1nnnn
End TimePeriod 1	41020	Register 4nnnn
Error	1001	Register 0nnnn
Flow Rate	41007	Register 4nnnn
FlowTimePeriod 1	41022	Register 4nnnn
Input Register	41003	Register 4nnnn
Input Type	41004	Register 4nnnn
K Factor	41001	Register 4nnnn
Log Data	10002	Register 1nnnn
Number Records	41017	Register 4nnnn
Rate Period	41005	Register 4nnnn
Status	41006	Register 4nnnn
Volume Period 1	41018	Register 4nnnn

Step 2 Build Ladder Logic

The required ladder logic will be developed in this step, and the FLOW function will be configured. The FLOW function has an Element Configuration dialog where required setup parameters may be entered.

Select the appropriate Controller Type in the Controller menu.
Create a Default Register Assignment with the appropriate lower I/O board.

Delete the entry DIN Controller Digital Inputs, then replace it with CNTR Controller Counter Inputs. Assign a Start Register of 30011. To be used if using a 5691 simulator module.

Enter the ladder logic as shown in Figure 47.

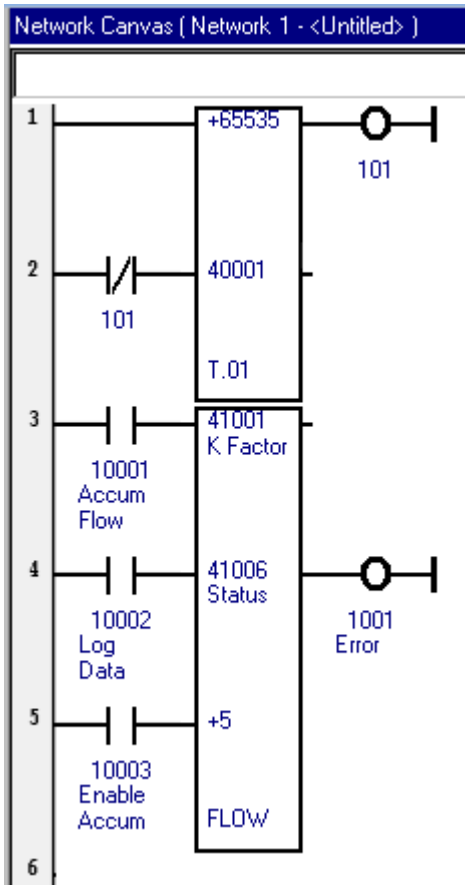


Figure 47: Ladder logic for FLOW demonstration

Select the FLOW function by clicking on it once with the left mouse button.

Complete the FLOW function Element Configuration as shown in Figure 48.
Save the program as FLOWDEMO.tpj.

Ensure that switch 3 is turned on to enable the Flow function, and that switches 1 and 2 are both off before running the program.

Download the program to the controller, then go online to monitor execution.

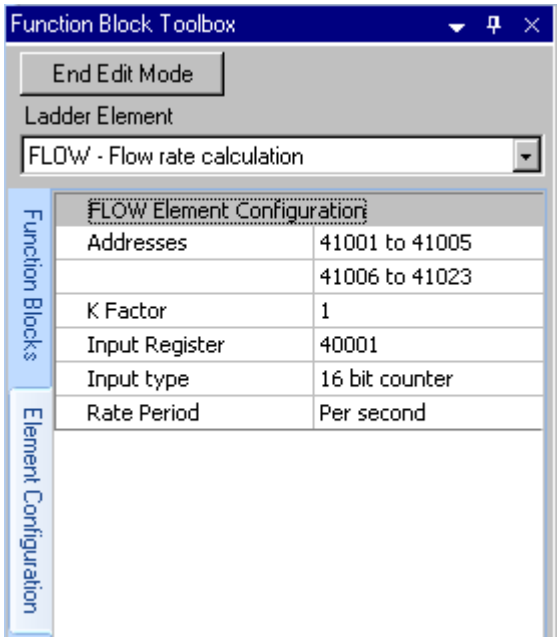


Figure 48: FLOW function configuration

Step 3 Monitor Program Operation

To be used if using a 5691 simulator module:

- With a pulse signal being accumulated by the SCADAPack, it will be possible to demonstrate the operation of the FLOW function. The student may vary the measured flow rate by adjusting the pulse frequency. Use the Register Editor to monitor the flow rate and accumulation.
- Once online with the controller, create a new Group in the Register Editor.
- Add register 30011, the pulse accumulator, to the group using the Unsigned Double format.
- Select the FLOW function, then right click on it. Select Monitor Element. This will add all of the configuration registers and the output registers to the Register Editor group, with each register in its correct data format.

Registers 41009, 41011, 41013 and 41015 are not required for this demonstration. Select them and hit the Delete button to remove them from the Register Editor group.

Check register 30011. Its value should be incrementing at a rate set by the Counter Freq potentiometer on the 5699 I/O board. Adjust this pot and verify that the pulse accumulator rate also varies. It should vary between approximately 5 and 150 pulses per second (Hz).

Note that registers 41001 through 41005 are the configuration registers, as entered in the FLOW function's Element Configuration dialog.

Note that the Flow Rate, in register 41007, should be 0 at this point. The Accumulate Flow switch is still off, so the FLOW function is not operational.
Set the Counter Freq potentiometer fully clockwise. This will set the input pulse rate to approximately 150 Hz.

Turn on switch 1.

This causes the FLOW function to begin accumulating flow.
Monitor register 41007, the Flow Rate. Note that the flow should be approximately 900, if the pulse rate is 150 Hz and the K Factor is 10. The Rate Period is set to Per Minute. Thus the flow rate is calculated as: (150Hz / 10) * 60 seconds = 900. This may mean 900 gpm, 900 m3/min, or perhaps 900 barrels/min.

Set the Counter Freq potentiometer fully counter-clockwise. This will be about 5 Hz. If the pulse rate is too slow, the FLOW function will generate Status code 5, "Pulse rate is too low for accurate flow rate calculation." The Flow Rate should read approximately 30. Note that at this low pulse rate the flow calculation will only update once every 10 seconds.
Increase the flow rate until the Status register returns to 0. (No error)

Note that register 41018, (Volume Period1) is incrementing. This is the current period, and so the value is live. It only stops if the flow stops or the Accumulate Flow input is turned off.

Note that the End Time Period1 register is showing the current time in Unix time format.

Note that the FlowTime Period1 register counts the number of seconds of flow in the period.
Simulate the end of a day by toggling the Log Data input on and off with switch 2.

Note that the three Period 1 values are frozen at this point, then pushed down to period 2.
New Period 1 values begin to accumulate as the new day begins.
Once the Log Data input has toggled 5 times the original day's data will be pushed out the bottom of the history. This data is now lost.

Experiment with new values for the K Factor and the Rate Period. Note that the K Factor should not be changed while flow is being accumulated. First turn the Accumulate Flow input off to avoid flow calculation errors.

PID Feedback Loop Control

A PID controller compares a desired value (Set point) against a real-world measured value (Process Value). Any error detected will cause a corrective output from the controller to occur, in an attempt to reduce the error. This is called a feedback loop. The PID controller uses three tuning parameters called Proportional, Integral and Derivative to adjust how the controller reacts. These are also known as Gain, Reset Time and Rate.

The PID controller to be used in the following exercise is the PIDA function. This controller has an analog-type floating point output, which can be sent to one of a SCADAPack's analog out channels. This will send a 4-20 mA or 1-5 V signal to a control device such as a flow control valve.

The PIDA Process Value is taken from an external device such as a flow meter, a pressure transmitter or an RPM indicator. This value is typically input to the controller through a 4-20 mA analog input, or perhaps through a modbus message from the measurement device. The Set point is typically entered from an HMI screen, if its value needs to be changed. The tuning parameters are not normally available to the site operator, other than perhaps after entering a password.

A PID controller typically also has a manual mode, in which the operator can take control of the output and set it to any desired value. In the PIDA function a digital input is available to allow switching between Auto and Manual modes, and a Manual mode value may be entered from an HMI.

In the following demonstration, no actual feedback is possible. It will be up to the student to simulate feedback by adjusting the analog input potentiometer on their demonstrator I/O board.

Tuning of a PID loop can be a complex and involved process, beyond the scope of this document. However, included here is a brief discussion of some of the key parameters.

Proportional Gain – Increasing Gain tends to reduce the Error. If Gain is increased too far, however, the loop will begin to cycle or oscillate. Gain alone can never reduce the Error to zero.

Integral or Reset Time – Reset is added to eliminate the Error. As Reset time is decreased the controller output becomes more likely to oscillate. The oscillation period also tends to increase. In the PIDA function it is measured in Seconds per Repeat.

Derivative or Rate Time – This parameter is typically not used in applications such as flow control. If not used, its value must be set to 0 (zero) to disable it. Rate allows a degree of predictive action, in order to reduce dead time. This is the time between a PID output change and a resulting change in the process.

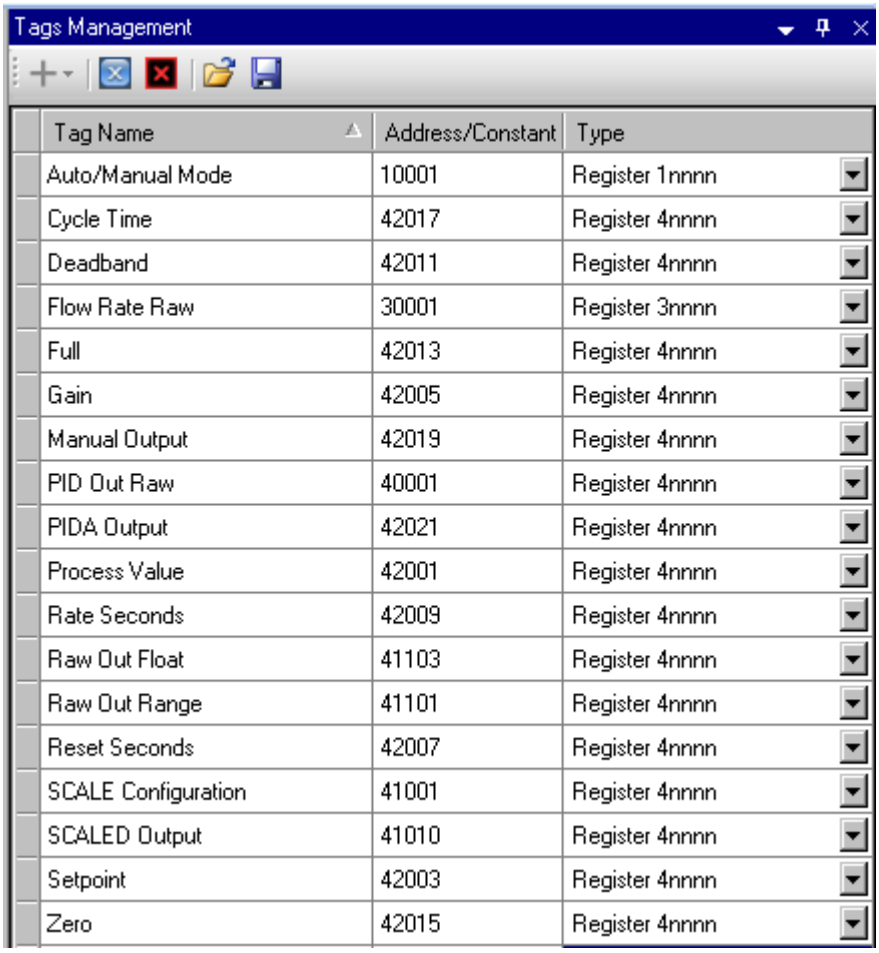
Introductory Tuning – The tuning of a PID loop can be very complex, particularly if Rate is involved. Below is a simple introductory technique to set the tuning. This brief discussion will not include Rate.

Initially set the Reset value to a very high setting, for example 1000 seconds.
While the PIDA is controlling the process in Auto mode, increase the Gain slowly until the PID Output begins to oscillate.
Measure the oscillation time. This is the Natural Period of the loop.
Divide the Gain used at this point by 3, and use this value for Gain.
Use the Natural Period of oscillation as the Reset Time.

CAUTION: The procedure above is a very basic guide for training purposes ONLY. Anyone performing a PID tuning procedure under actual field conditions must understand the process and follow all appropriate safety precautions.

Step 1 Create Tag Names

Open a new file in TelePACE.
Enter the following tag names using the Edit Tag Name dialog or import them from the supplied PIDADEMO.CSV file.



Tag Name	Address/Constant	Type
Auto/Manual Mode	10001	Register 1nnnn
Cycle Time	42017	Register 4nnnn
Deadband	42011	Register 4nnnn
Flow Rate Raw	30001	Register 3nnnn
Full	42013	Register 4nnnn
Gain	42005	Register 4nnnn
Manual Output	42019	Register 4nnnn
PID Out Raw	40001	Register 4nnnn
PIDA Output	42021	Register 4nnnn
Process Value	42001	Register 4nnnn
Rate Seconds	42009	Register 4nnnn
Raw Out Float	41103	Register 4nnnn
Raw Out Range	41101	Register 4nnnn
Reset Seconds	42007	Register 4nnnn
SCALE Configuration	41001	Register 4nnnn
SCALED Output	41010	Register 4nnnn
Setpoint	42003	Register 4nnnn
Zero	42015	Register 4nnnn

Step 2 Build Ladder Logic

This program will take an analog input from the demo I/O board and convert it to engineering units. The input will represent flow through a pipeline, with a minimum scaled value of 0 at 6553 and a maximum of 300 at 32767. The PIDA function will calculate a floating point output, with a range of 0 – 100%. This will be used to drive a control valve. The floating point number will then be converted back to a raw value and sent to one of the controller's analog outputs.

Select the appropriate Controller Type in the Controller menu.

Create a Default Register Assignment with the appropriate lower I/O board.

Click Add, then insert the SCADAPack AOUT module. Assign a Start register of 40001.

Enter the ladder logic as shown in Figure 49.

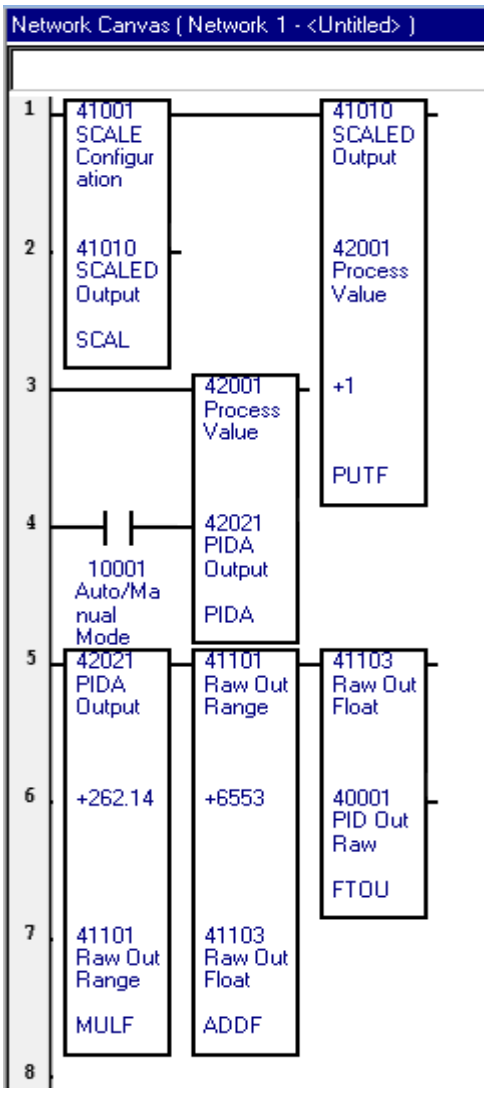


Figure 49: Ladder logic for PIDA demonstration

Select the SCAL function by clicking on it once with the left mouse button.

Complete the SCAL function Element Configuration as shown in Figure 50.

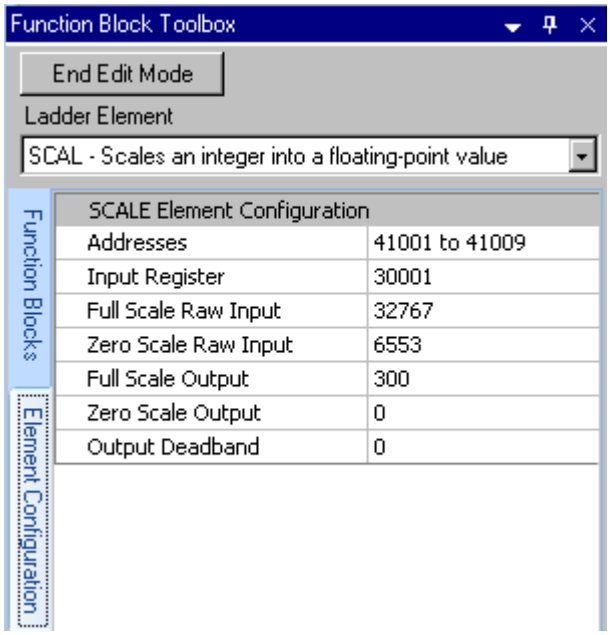


Figure 50: SCAL function configuration

The analog input circuitry will convert a 4-20 mA signal into a raw analog value between 6553 at 4 mA and 32767 at 20 mA. The offset is because the circuitry will read as low as 0 mA, providing a raw value of 0 at that point.

The SCAL function converts the raw values between 6553 and 32767 into scaled values of between 0 and 300 units of flow. The student may presume whatever flow measurement units they are familiar with. (eg gal/min, m3/hr, e3m3/day, etc)

Select the PIDA function by clicking on it once with the left mouse button.

Complete the PIDA function Element Configuration as shown in Figure 51.

Save the program as PIDADEMO.tpj.

Ensure that switch 1, the Auto/Man Mode selector, is OFF for Manual mode.

Download the program to the controller, then go online to monitor execution.

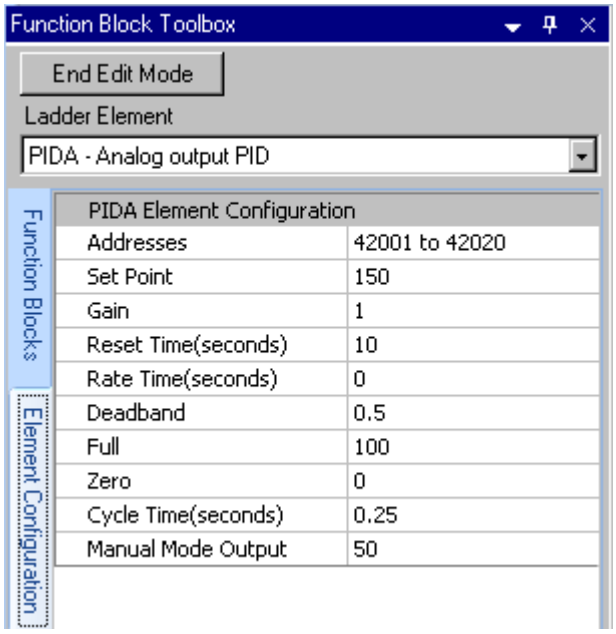


Figure 51: PIDA function configuration

The PIDA function, when they Auto/Man Mode switch is On, calculates a PID Output value. When the switch is off, the Manual Output value is sent directly to PID Output.
The entered Set point value specifies the desired flow rate. The PID Output will increase, to open the valve, if the flow rate drops below the set point. The PID Output will decrease, to close the valve, if the flow rate goes above the set point.

The PID Output value is a floating point number, ranging from 0 – 100%. It is multiplied in a MULF by 262.14, to scale it up to the raw range. (32767 – 6553 = 26214)
A value of 6553, equal to the 4 mA point or 0% valve rotation, is added to the raw range value. This produces a value at 0% of 6553, and at 100% of 32767.
The last step is to convert the floating point raw value to integer, and send it to register 40001. This was assigned to Analog Out 0 in the Register Assignment.

Step 3 Monitor Program Operation

With an analog input signal coming from the demo I/O board, simulating flow rate as the Process Value, it is possible to demonstrate a PID loop. The student may vary the flow rate and then watch the PID Output change in reaction. Use the Register Editor to monitor the Process Input (flow rate), the Set point and the resulting PID Output value.

Once online with the controller, create a new Group in the Register Editor.
Manually add register 30001, the raw analog input representing flow rate, and 40001, the raw analog output to the control valve.

Select the PIDA function, then right click on it. Select Monitor Element. This will add all of the configuration registers and the output registers to the Register Editor group, with each register in its correct data format.

Registers 42023 through 42029 are not required for this demonstration. Select them and delete them from the Register Editor group.

Monitor register 30001, the raw flow rate, and 42001, the scaled Process Input values at the same time as the analog input potentiometer is adjusted.

Note that at a raw value of 6553 the scaled value should be 0, and at a raw value of 32767 the scaled value should be 300. Also note that if the raw value is reduced to zero, the scaled value is -75. **Why is this?**

With the Auto/Man Mode switch in Manual mode, the PID Output will immediately follow the value entered into the Manual Mode Output register. If it is desired to have the value sent to the valve change more slowly, additional logic will be required to ramp the value up or down.

Note that when a Manual Mode Output value of 0 is entered the PID Out Raw value, register 40001, goes to 6553. When 100 is entered the raw output value goes to 32767.

Enter a Manual Mode Output value of 110. Note that the PID Output goes to 100. Try a value of -10. The PID Output will go to 0. It is clamped at these points by the Full and Zero settings.

Adjust the analog input pot so that the difference between the Process Input (flow rate) and the Set point (the Error) is less than the Deadband.

Now switch the PID controller into Automatic mode by turning on switch 1.

Note that the PID Output does not change. This is because the error is less than the Deadband, thus no new calculation is done.

Turn the potentiometer counter-clockwise to reduce the Process Value to a value slightly outside the Deadband range. Once the Error exceeds the Deadband, note that the PID Output begins to slowly rise. If the Process Value is flow rate, and the flow drops below the Set point, then the flow control valve must begin moving open. This allows more product through the valve, thus increasing the flow rate.

Turn the pot farther counter-clockwise, and note that the PID Output begins to rise more quickly. If the student does not reduce the error by turning the pot clockwise, the PID Output will continue to rise until it is clamped by the Full value (100%). This would cause the control valve to rotate to its full open position.

Turn the pot back clockwise until the Process Value rises above the Set point. Now the flow rate on the pipeline is too high, and as a result the valve needs to begin closing to allow less product through. As a result the PID Output should begin dropping.

At this point, try changing the values of Gain and Reset. Only change one at a time. Note that increasing Gain or reducing Reset will both cause the PID loop to react more vigorously.

Data Logging and SCADALog

The DLOG function block is used to populate a data log in a SCADAPack controller. Each time a low to high transition occurs on its Grab Data input a record is generated. A record consists of a number of data fields. As many as 16 data logs may exist in a controller at any time, and each data log may contain anywhere from one to eight fields. This will allow for as many as 128 data points to be captured in a single SCADAPack.

The log contents may be deleted at any time by toggling the Delete Log input low then high again. This may be done after data is read out of the log using the GETL function if desired. If the SCADALog software is instead used to read logs, the log contents may be automatically deleted after the logs are read if desired.

Each data log field may be assigned to one of six data types. These include 16 bit unsigned and signed integer, 32 bit unsigned and signed integer, 32 bit floating point and Time & Date. There is no option to log boolean values, but if this is desired a block of booleans may be transferred into a holding register using the MOVE function.

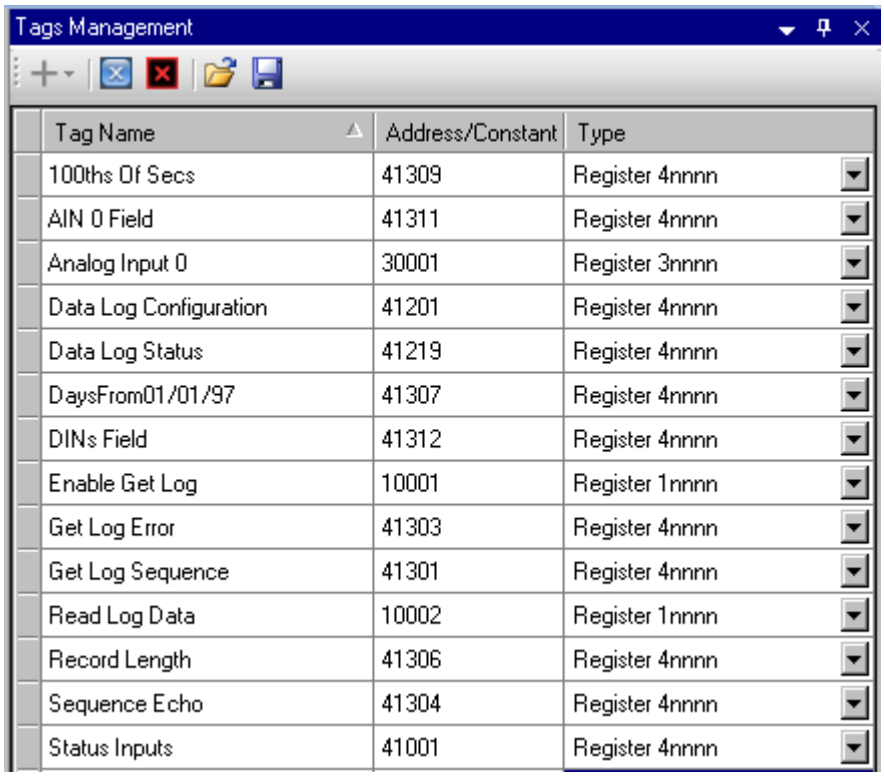
The Time & Date data type is stored in two 32 bit unsigned integer registers. If the Time & Date is read using the GETL function the user will need to interpret the data manually. SCADALog will convert it to actual time and date information automatically. The first 32 bit register contains the number of complete days since 01/01/97. The second 32 bit register contains the number of hundredths of a second since the start of the current day.

Each data log record is automatically assigned a sequence number, starting at 0. This is a 32 bit unsigned value. If SCADALog is used, there is no need to deal with the sequence number. If GETL is used, the programmer must track the sequence number to know its current value. (using a counter triggered by the Grab Data source) The programmer must specify the sequence number of each record to be retrieved.

The programmer must be aware of the maximum available memory for data logs in the controller they are working with. This is documented in the DLOG function Help section of the TelePACE manual. For example, a data log record containing Time & Date (4 words), two floats (2 words each) and two integers (1 word each) would require 10 words. Each time the Grab Data input is toggled this amount of memory is used.

Step 1 Create Tag Names

Open a new file in TelePACE.
Enter the following tag names using the Edit Tag Name dialog or import them using the supplied DLOGDEMO.CSV file.



Tag Name	Address/Constant	Type
100ths Of Secs	41309	Register 4nnnn
AIN 0 Field	41311	Register 4nnnn
Analog Input 0	30001	Register 3nnnn
Data Log Configuration	41201	Register 4nnnn
Data Log Status	41219	Register 4nnnn
DaysFrom01/01/97	41307	Register 4nnnn
DINs Field	41312	Register 4nnnn
Enable Get Log	10001	Register 1nnnn
Get Log Error	41303	Register 4nnnn
Get Log Sequence	41301	Register 4nnnn
Read Log Data	10002	Register 1nnnn
Record Length	41306	Register 4nnnn
Sequence Echo	41304	Register 4nnnn
Status Inputs	41001	Register 4nnnn

Step 2 Build Ladder Logic

This program will gather data every two seconds, triggering the DLOG's Grab Data input with a PULS function. The DLOG has three fields: Time & Date, Analog Input 0, and Status Inputs. As there is no boolean data type available, a MOVE function is used to copy the first 16 digital inputs into a holding register. The GETL function is included to demonstrate its use in retrieving log records and placing them into modbus registers.

Select the appropriate Controller Type in the Controller menu.
Create a Default Register Assignment with the appropriate lower I/O board.
Enter the ladder logic as shown in Figure 52.

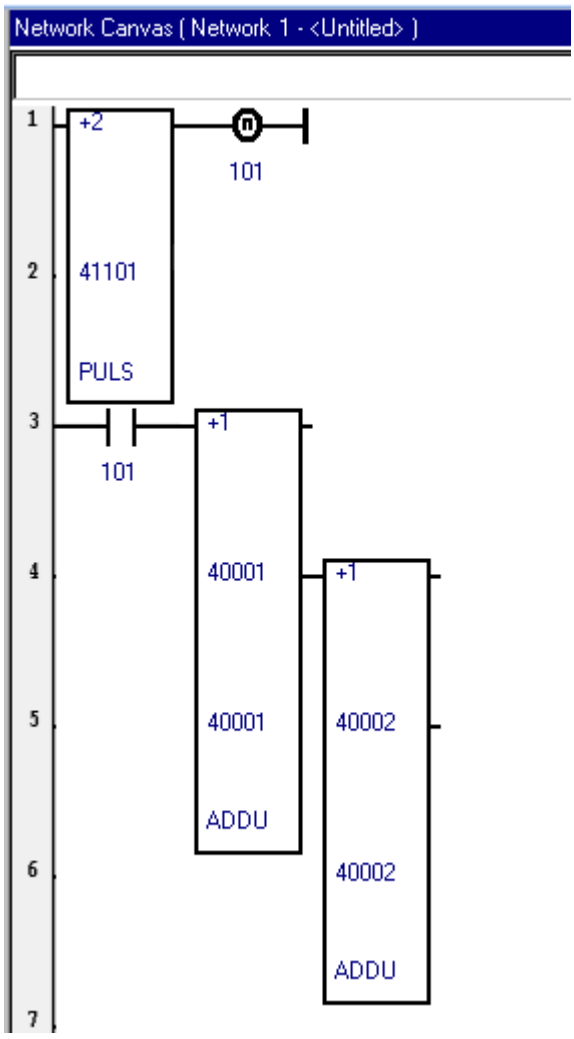


Figure 52: Ladder logic for DLOG demonstration

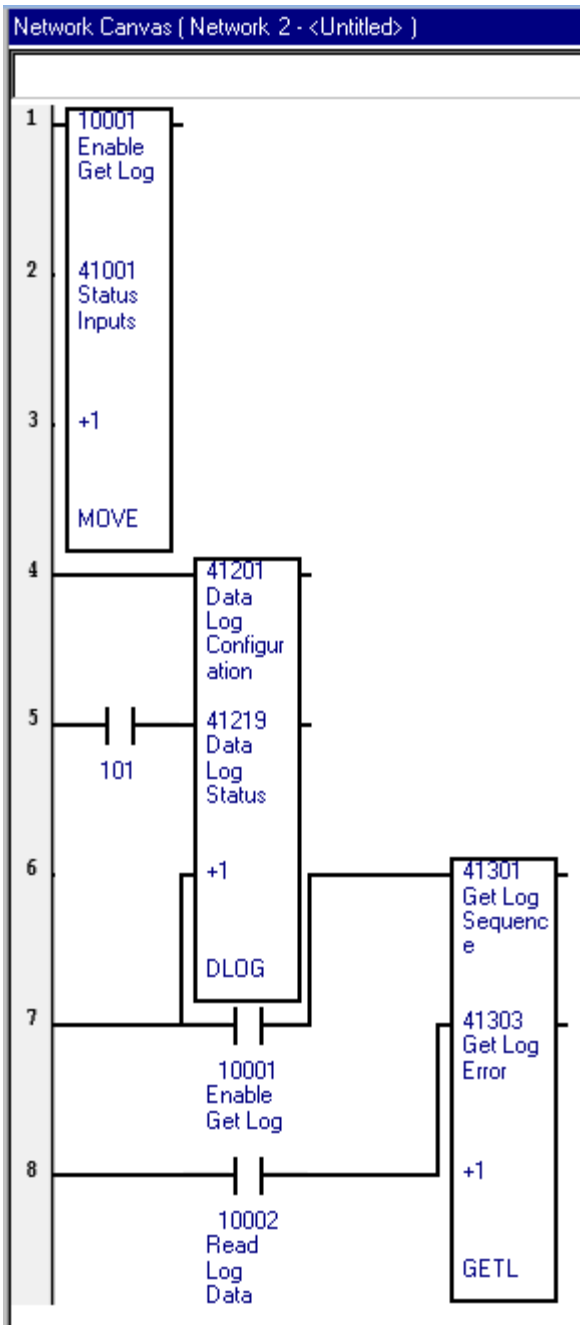


Figure 53: Ladder Logic for DLOG 2

Select the DLOG function by clicking on it once with the left mouse button.

Complete the DLOG function Element Configuration as shown in Figure 54.

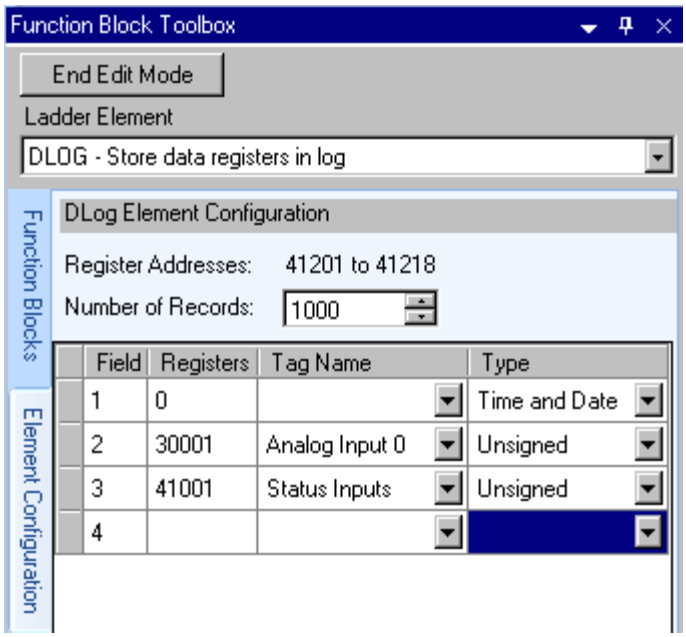


Figure 54: DLOG function configuration

The DLOG will gather a total of 1000 records in a first in – first out buffer.
Three fields will be gathered in each record. The first is Time & Date. This does not require modbus registers – just select the appropriate data type. Second is register 30001, which is Analog Input 0. Third is register 41001, which contains the first 16 digital inputs.

Save the program as DLOGDEMO.tpj.

Ensure that switch 1, the Enable GETL input, is ON to enable the GETL function.

Download the program to the controller, and then go online to monitor execution.

Step 3 Monitor Program Operation

With an analog input signal and digital inputs coming from the demo I/O board, the program will begin to gather data every two seconds. The student may adjust the analog input and toggle the 3rd and 4th switches to generate data. (Switches 1 and 2 are used in the demo, and should only be toggled as required) Use the Register Editor to monitor and test the operation of the GETL function.

Once this is done, SCADALog will be used to demonstrate automated reading of log contents. Once online with the controller, create a new Group in the Register Editor.

Press Add, then add the following registers in order to test the GETL function:

41303 – Unsigned	41301 – Unsigned Double
41306 – Unsigned	41304 – Unsigned Double
41311 – Unsigned	41307 – Unsigned Double
41312 – Unsigned	41309 – Unsigned Double

The DLOG has been gathering data every two seconds, starting with sequence # 0. Enter a desired sequence # to view into register 41301.

With switch 1 on to enable the GETL function, toggle switch 2 (Read Data) on and off again.
If a 23 error code is generated, the sequence # does not yet exist. Enter another value, and then toggle the Enable input (switch 1) off and on again to clear the error.

If no error is generated the log will return the field values for Time & Date, AIN 0 and DINs.
Enter another valid sequence number, then toggle the Read Data input to retrieve data from another record.

Note that the sequence number will need to be tracked in order to retrieve data in future.

Once you are satisfied with the operation of the DLOG and GETL functions, close TelePACE.
Open SCADALog, and start a new configuration by clicking on File | New.
Set up Communication to the controller by clicking on Communication | PC Communication Settings.
The Modbus RTU protocol is used as with TelePACE.

Read the controller's data log configuration by clicking on Data Log | Configuration, then clicking the

Read button. This will read the setup of all logs in the controller.

For each data field a descriptive title may be added. Click on the field, then click Edit Title.

Click Ok. The three columns are listed in the order the fields were added in the DLOG.

Click the Save button, and give the file a descriptive name. This saves the controller's log configuration and communication settings into an *.slc file.

To read log data from the controller, click on Data Log and then select Read Logs. Select which logs are to be read in this case log 1 is the only log, so leave it at All Logs. Also choose whether the log contents are to be purged (deleted) after being read. Then click Ok to start the read.

Click Save after the read is done. This again saves the *.slc file, but also saves a *-01.csv file. The -01 part specifies which log it is. It is saved as a comma-separated variable file.

<u>If it is desired to manipulate the log contents afterwards, the contents must be exported to another csv file. Editing the *-01.csv file will corrupt it, causing SCADALog to not be able to open it afterwards.</u>

To perform an Export, open the appropriate log. (Click on the log selection button at the top of the screen) Go to the File menu, and then click Export. The Export file will be saved as another csv file, but name it in such a way as to prevent confusion with the SCADALog log file.

Modbus Serial Master Communications

The MSTR function block is used to initiate a master message to exchange data with another device via the serial port of a SCADAPack controller. The message may use either the Modbus or DF1 protocols. Modbus is an open-source, non-proprietary protocol which is implemented by many equipment manufacturers in order to maximize inter-operability. DF1 is a protocol primarily used by Allen-Bradley and anyone wishing to communicate with their equipment. The following exercise will demonstrate only the Modbus protocol.

Each time a low to high transition occurs on the master's Enable input a message is sent. A timer starts at the same time. If a valid reply arrives before the timer's Time Out delay is reached, the Message Complete output goes true and the message succeeds. If no reply has arrived after the Time Out delay, or if the reply is corrupted, the Message Error output goes true.

Only one master message may be active on each serial port at any one time. The Modbus protocol must receive a reply to each message before the next message is sent. A separate message may be sent on a different port at the same time, as it will be on a different network. If it is desired to send multiple messages simultaneously, the Ethernet port of a SCADAPack 2 or 32 must be used, along with the MSIP function. As many as 20 MSIP messages may be active on an Ethernet port at any time.

Modbus is a low-level protocol which does not include such refinements as Retries and Report By Exception. If these features are desired then additional ladder logic code must be written to implement them. An example of Retry logic may be shown after the following demonstration. It is possible to add this functionality and more by instead using the DNP3 protocol.

Within the Modbus protocol there are eight primary functions. The programmer may choose to send a Read or a Write message. It is possible to read any of the four Modbus data types: coil, input status, analog input or holding register. When writing, it is only possible to write to the output registers – the coil and holding register types. A message type sending either one or multiple registers may be chosen. It is also possible to send Enron Modbus messages, though this will not be covered.

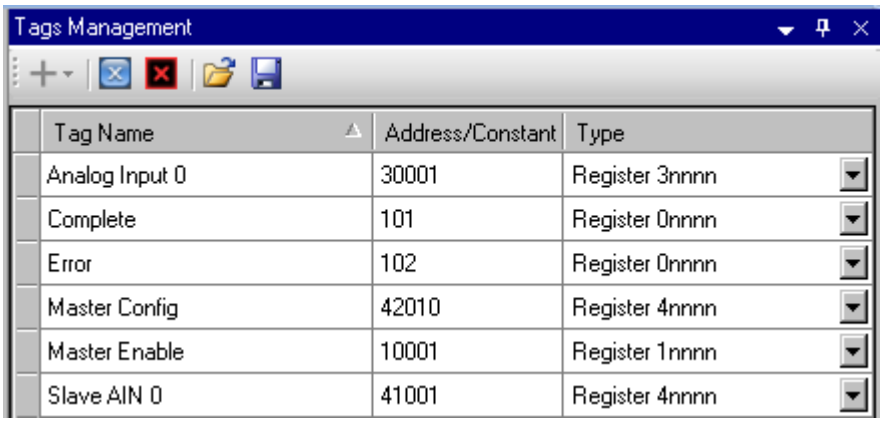
The desired protocol, baud rate, station number and other parameters must be set for each port in the Controller | Serial Ports dialog before any given port is used for Modbus communications. This information will be downloaded to the controller along with the ladder logic.

Choosing the desired serial port (along with its previously set parameters) is done in the MSTR element configuration. The slave device's serial port Modbus station number must also be specified, along with the data source and destination register addresses.

The MSTR function section of the TelePACE manual also discusses such issues as the maximum number of registers that can be read or written in a single message, and how to use the DF1 protocol in a MSTR message.

Step 1 Create Tag Names

Open a new file in TelePACE.
Enter the following tag names using the Edit Tag Name dialog or import them using the supplied MSTRDEMO.CSV file.



Tag Name	Address/Constant	Type
Analog Input 0	30001	Register 3nnnn
Complete	101	Register 0nnnn
Error	102	Register 0nnnn
Master Config	42010	Register 4nnnn
Master Enable	10001	Register 1nnnn
Slave AIN 0	41001	Register 4nnnn

Step 2 Build Ladder Logic

For this demonstration two student's SCADAPacks will be connected together to create a simple network. As only one controller may be the Master at any time, the Master Enable switch is used to select this. The PULS function in this program will turn on once every ten seconds. With the Master Enable switch on, the off – on transition generated by the PULS will trigger the MSTR function to send a message. If the message succeeds the upper output will go true very quickly. If it fails, the lower output will go true after two seconds.

Select the appropriate Controller Type in the Controller menu.

Create a Default Register Assignment with the appropriate lower I/O board.

Enter the ladder logic as shown in Figure 55.

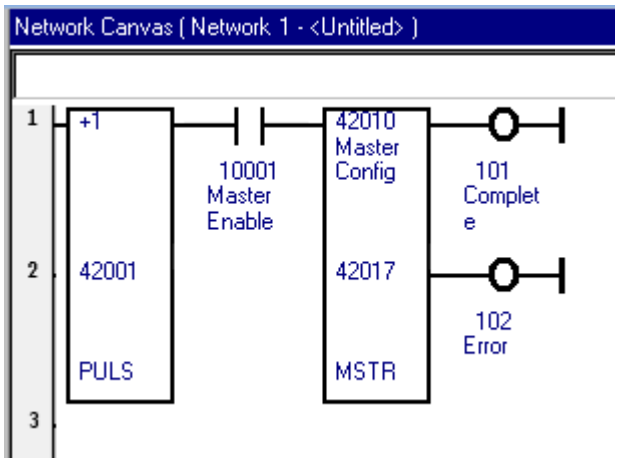


Figure 55: Ladder logic for MSTR demonstration

Go to Controller | Serial Ports and ensure com1 is selected.

Set the com1 parameters as shown in Figure 56.

The Station number will be YOUR controller's station number. The slave controller's station number will be set later, in the MSTR element configuration. Both of these will be assigned by the instructor.

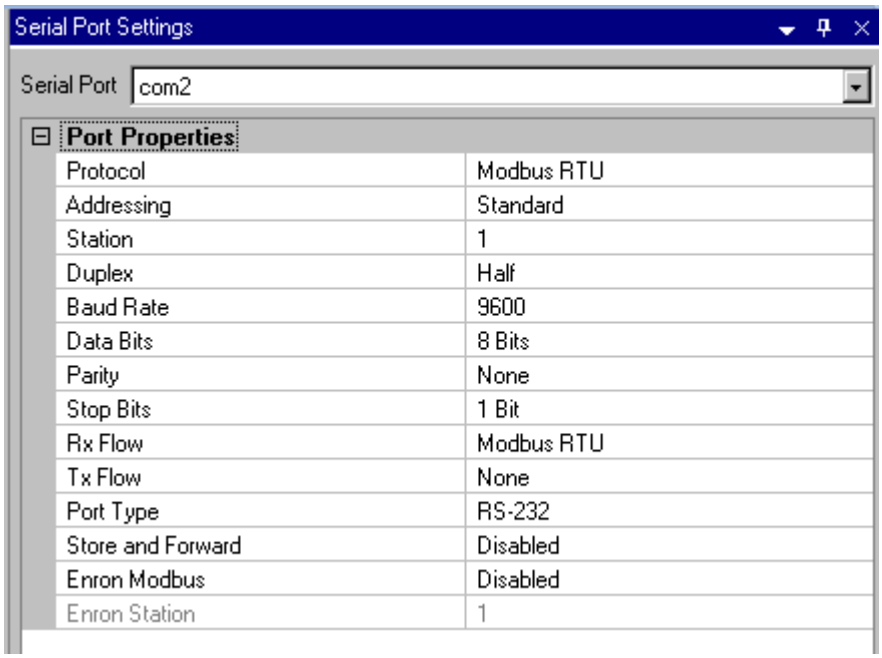


Figure 56: Controller Serial Port Settings

Select the MSTR function by clicking on it once with the left mouse button.

Complete the MSTR function Element Configuration as shown in Figure 57.

The Slave RTU Address will be the Modbus station number of the other student's controller.

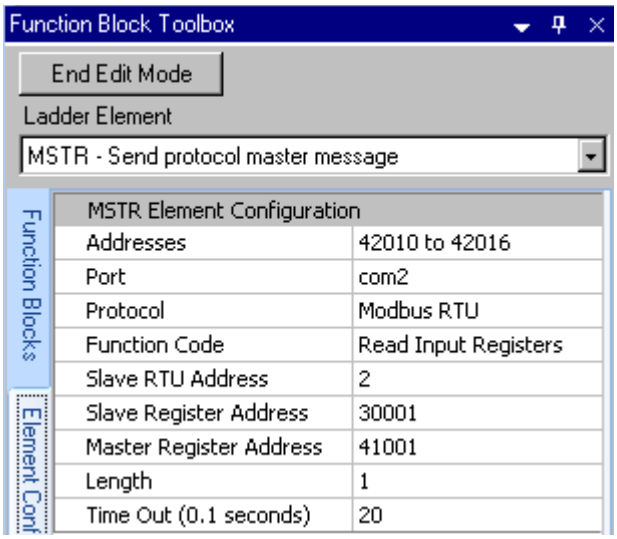


Figure 57: MSTR configuration to Read Input Registers

Save the program as MSTRDEMO.tpj.

Ensure that switch 1, the Master Enable input, is OFF to disable the MSTR function.

Download the program to the controller, and then go online to monitor execution.

Step 3 Monitor Program Operation

Each student will take a turn being the Master station. The other student in each pair will be the Slave station. Com port 2 of the two controllers will be connected with a null modem cable. In its simplest this could be a 3 wire cable, though a commercially built cable will also work. When the PULS output goes true on the Master station it will send a message to the Slave, either requesting an analog input value or setting the state of a digital output.

Once online with the controller, create a new Group in the Register Editor.
Press Add, then add the following registers in order to test the MSTR function:
30001 – Unsigned 41001 – Unsigned

The instructor will state who in each team is to start out as the Master station. That person should turn on their Master Enable switch. A master message will be sent the next time that controller's PULS function sets its output True. The message will be re-sent every 10 seconds until the Master Enable switch is turned off.

The student who is Master should watch their Complete and Error outputs. If Complete goes true then the message succeeded. If Error goes true then some configuration problem exists. If the message has succeeded, then the Master controller's register 41001 will now display the same value as the slave controller's register 30001. (Analog Input 0) If the message fails, the Error output will go on two seconds after the MSTR was enabled. This is the value the Time Out parameter has been set to. If this occurs, check the Controller | Serial Port settings for com1 in both controllers. The station number must be 2 in one and 3 in the other. Then check the MSTR configuration in the controller which is currently the Master. It should have the station number of the other controller set in it. Also check the serial cable. It must be a null cable, connected to com1 on both controllers.

The student at the Slave controller should adjust the value of Analog Input 0, and then the student at the Master should check to see if this new value is properly received. Once the first student's controller is working as Master, that person should turn Off their Master Enable switch and the other student should turn their switch On to reverse roles.

Now ensure that the new Master station is receiving live data every 10 seconds into its register 41001 from the other station's register 30001. Once both students have been able to read data from the other controller, it's time to try to write data to the other controller.

Each student should go to Edit Online mode.
Select the MSTR function, right click and go to Element Configuration.
Change the Function Code from Read Input Registers to Write Single Coil.
Change the Slave Register Address from 30001 to 00006. (6^{th} coil, tied to an LED)
Change the Master Register Address from 41001 to 10004. This is the 4^{th} switch.
Click ok, which sends the changes to the controller. Save the changes.
The 4^{th} (right-most) switch on the controller whose Master Enable switch is on will be the data source. Whether that switch is on (1) or off (0) its state will be written to the Slave controller once every 10 seconds. Ensure the slave's 6^{th} LED goes on within 10 seconds when the Master's 4^{th} switch is turned on, and goes off when the switch is turned off.

If this fails, perform the same troubleshooting as before.

Once the Write message is working, turn the Master Enable switch Off for the first student, and turn it on for the second. Ensure that the new Master can also write to its slave's LED.

Try increasing the pulse rate. It can go as fast as every 1 second. If this is done, remember that the Time Out period must be shortened as well. At a 1 second pulse, the On period is only 500ms. To make the Error output work the Time Out value must therefore be less than 500ms. To see the Error output go true the Update Rate in Register Editor must be faster.

Also try having BOTH Master Enable switches on at the same time. This technically is a bad idea. But due to the very short duration of the MSTR messages any collisions will be quite infrequent. At lower data rates, or if the message is being triggered more often, or if the message is longer, collisions will become more likely.

TelePACE 3.3x Commands in TelePACE Studio

TelePACE Studio's new user interface replaces menus, toolbars, and most of the task panes that were used in TelePACE 3.3x and as such has been designed to facilitate their use.
This document provides a list of programming commands found in TelePACE 3.3x and their corresponding locations in TelePACE Studio and is intended for the experienced TelePACE 3.30 user who wants to find familiar commands quickly in the new user interface.
The order of the subtopics in this section matches the default order for TelePACE version 3.30. The subtopic headings match the menu and toolbar names in TelePACE 3.3x.

File Menu

TelePACE 3.30 Location	TelePACE Studio Location
New	TelePACE Application button | New
Open	TelePACE Application button | Open
Save	TelePACE Application button | Save
Save As	TelePACE Application button | Save As
Page Setup	TelePACE Application button | Print menu
Print	TelePACE Application button | Print menu
Select Print Items	TelePACE prints the open Ladder Logic project. You no longer need to select the items to print.
Exit	TelePACE Application button | Exit button

Edit Menu

TelePACE 3.30 Location	TelePACE Studio Location
Undo	Edit Ribbon | Edit Group | Undo
Cut Selected	Edit Ribbon | Edit Group | Cut
Copy Selected	Edit Ribbon | Element Group | Copy
Cut Networks	Edit Ribbon | Network Group | Cut Networks
Copy Networks	Edit Ribbon | Network Group | Copy Networks
Paste	Edit Ribbon | Ladder Tab | Edit Group | Paste
Insert	
• Insert vertical shunt	Right click on the Network Canvas
• Insert element	Double click on a cell in the Network Canvas or Drag or Insert from the Function Block Tool box.
• Insert column	Not supported in TelePACE Studio
• Insert row	Not supported in TelePACE Studio
• Insert network before	Edit Ribbon | Network Group | Insert Before
• Insert network after	Edit Ribbon | Network Group | Insert After
Delete	
• Delete vertical shunt	Select the cell on the Network Canvas + Delete
• Delete elements	Select the elements on the Network Canvas + Delete
• Delete column	Not supported in TelePACE Studio
• Delete row	Not supported in TelePACE Studio
• Delete network	Edit Ribbon | Network Group | Delete Network
• Delete all networks	Edit Ribbon | Network Group | Cut Networks
Toggle Vertical Shunt	Right click on the network canvas
Tag Names	Edit Ribbon | I/O Group | Tags
Erase All Tags	Edit Ribbon | I/O Group | Tags
Export Tag Names	Edit Ribbon | I/O Group | Tags
Import Tag Names	Edit Ribbon | I/O Group | Tags
Network Title	Edit Ribbon | Network Group | Title
Element Configuration	Function Block Toolbox | Element Configuration Tab
Registers	Edit Ribbon | I/O Group | Register Editor

Search Menu

TelePACE 3.30 Location	TelePACE Studio Location
Next Network	Edit Ribbon | Navigate Group | Next
Previous Network	Edit Ribbon | Navigate Group | Previous
Go Network	Edit Ribbon | Navigate Group | Go To
Find Address	Edit Ribbon | Navigate Group | Search
Find Device	Edit Ribbon | Navigate Group | Search
Find Tag Name	Edit Ribbon | Navigate Group | Search
Repeat Last Find	Edit Ribbon | Navigate Group | Search
Replace Address	Not supported in TelePACE Studio
Multiple Coils	Edit Ribbon | Navigate Group | Search

Controller Menu

TelePACE 3.30 Location	TelePACE Studio Location
Type	Status Bar | Controller:
Serial Ports	Edit Ribbon | Communication Group | Serial
IP Configuration	Edit Ribbon | Communication Group | IP
Register Assignment	Edit Ribbon | I/O Group | Register Assignment
Outputs on Stop	Edit Ribbon | I/O Group | Register Assignment
Store and Forward	Edit Ribbon | Communication Group | Store and Forward
DNP Configuration	Edit Ribbon | Communication Group | DNP Configuration
DNP Status	Program Ribbon | View Group | DNP Outstation Diagnostics
DNP Master Status	Program Ribbon | View Group | DNP Master Diagnostics
Initialize	Configure Ribbon | Controller Group | Initialize
Real Time Clock	Configure Ribbon | Controller Group | Clock
Monitor Element	Network Canvas | Right Click | Monitor Element

List Forced Registers	Edit Ribbon | I/O Group | Register Editor | Forced Registers
Remove All Forces	Edit Ribbon | I/O Group | Register Editor | Forced Registers
Lock Controller	Configure Ribbon | Controller Group | Security
Unlock Controller	Configure Ribbon | Controller Group | Security
Override Controller Lock	Configure Ribbon | Controller Group | Security
Show Locked Status	Configure Ribbon | Controller Group | Security
C/C++ Program Loader	Configure Ribbon | Programs Group | C/C++
Flash Loader	Configure Ribbon | Programs Group | Flash Loader
Program Status	Configure Ribbon | Programs Group | Program Status

Communications Menu

TelePACE 3.30 Location	TelePACE Studio Location
Read from Controller	Program Ribbon | Program Group | Read
Write to Controller	Program Ribbon | Program Group | Read
PC Communications Settings	Program Ribbon | PC Communication Group
Connect to Controller	Program Ribbon | PC Communication Group
Disconnect from Controller	Program Ribbon | PC Communication Group

Activity Menu

TelePACE 3.30 Location	TelePACE Studio Location
Edit Off Line	Program Ribbon | Mode Group | Edit Off Line
Edit On Line	Program Ribbon | Mode Group | Edit On Line
Monitor	Program Ribbon | Mode Group | Edit On Line

Operation Menu

TelePACE 3.30 Location	TelePACE Studio Location
Stop	Program Ribbon | Execution Group | Stop
Debug	Program Ribbon | Execution Group | Debug
Run	Program Ribbon | Execution Group | Run

Options Menu

TelePACE 3.30 Location	TelePACE Studio Location
Screen Font	TelePACE Studio uses the Window system font.
Colors	Not supported in TelePACE Studio
Floating Point Settings	Configure Ribbon | Application Group | TelePACE Options
Tool Bar	Not used in TelePACE Studio
Title Bar	Title Bar
Status Bar	Status Bar
Single Tag Names	The full Tag name and Address are always displayed in TelePACE Studio
Double Tag Names	The full Tag name and Address are always displayed in TelePACE Studio
Tag and Address	The full Tag name and Address are always displayed in TelePACE Studio
Numeric Address	The full Tag name and Address are always displayed in TelePACE Studio
Allow Multiple Coils	Always supported in TelePACE Studio
Warning Messages	Configure Ribbon | Application Group | TelePACE Options

Help Menu

TelePACE 3.30 Location	TelePACE Studio Location
Contents	TelePACE Menu Bar | Help
About Program	TelePACE Menu Bar | About

New Features of TelePACE Studio not found in TelePACE 3.4

- **Quick Access ToolBar** – allows you to create a tool subset of any commands from the Tool Ribbons.
- **Register Assignment automatic features** – starting registers are automatically incremented as new modules are assigned so that there are no conflicts with existing modules. Only modules compatible with controller type are made available.
- **Automatic suggestion of Function Block (FB) registers – A**utomatic suggestion of FB registers is made to prevent overlap conflicts for FBs that involve an Element Configuration only. Note that register overlaps with Register Assignment of other simple FBs is not taken into account.
- **Tag Name management** – Edit tags at anytime without restrictions. Auto-generation of sequential tags is possible by selecting an existing tag as the common text root (e.g. AIN_1, AIN_2, AIN_3,…).
- **Unlimited undo/redo** – Complete editing history is maintained for the undo and redo editing feature.
- **Edit On-Line queues changes** – Editing changes are queued until Write to Controller is selected. On-line editing supports copy & paste now as well.
- **Custom Workspace** – You can rearrange the size, position and mix of viewing panes. Use Default Layout button to reset to default layout.
- **Run multiple instances of TP Studio** - This is useful for debugging two controllers at the same time, such as a master and slave application.
- **Validate command** – If the validate command finds an error, TelePACE will not allow the program to be download to the controller. It is expected that the scope of the command will expand, but currently the validate command only checks whether valid communication ports are selected for these properties based on the selected controller type:
  - Store and Forward communication ports
  - Function block element configuration
  - Register assignment communication ports
  - DNP routing entry ports.
- **Help** – Help provides a search utility, is printable by section or page, and has a page forward and backward controls.
- **Project Files in XML format** – TelePACE Studio projects are saved as a set of XML files plus one *.tpj file.