i-TRiLOGI

Ladder+BASIC

Version 6.45

Revision 1.1

Programmer's Reference

Copyright Notice and Disclaimer

Revision Sheet

Release No.	Date	Revision Description
Rev. 1	3/14/2013	Complete Update of all chapters for synchronization with TRiLOGI version 6.45 build 06
Rev. 1.1	5/03/2013	Added notes on using RESET and SETSYTEM 252,0 commands and performing a 'Reset' after program transfer to backup timer & counter set value data.

Conditions of Sale and Product Warranty

Triangle Research International Inc. (TRi) and the Buyer agree to the following terms and conditions of Sale and Purchase:

1. The FMD1616-10 Programmable Controllers are guaranteed against defects in materials or workmanship for a period of one year from the date of registered purchase. Any unit which is found to be defective will, at the discretion of TRi, be repaired or replaced.

2. TRi will not be responsible for the repair or replacement of any unit damaged by user modification, negligence, abuse, improper installation, or mishandling.

3. TRi is not responsible to the Buyer for any loss or claim of special or consequential damages arising from the use of the product. <u>The product is NOT certified to be FAILSAFE and hence must **NOT** be used in applications where failure of the product could lead to physical harm or loss of human life</u>. Buyer is responsible to conduct their own tests to meet the safety regulation of their respective industry.

4. Products distributed, but not manufactured by TRi, carry the full original manufacturers warranty. Such products include, but are not limited to: power supplies, sensors, I/O modules and battery backed RAM.

5. TRi reserves the right to alter any feature or specification at any time.

Notes to Buyer: If you disagree with any of the above terms or conditions you should promptly return the unit to the manufacturer or distributor within 30 days from date of purchase for a full refund.

TABLE OF CONTENTS

Page #

Chapter 1 Set Up i-TRiLOGI

1 I-TRILOGI 6 SETUP GUIDE

1.1 Introduction to the i-TRiLOGI Program

i-TRiLOGI is a trademark name used by Triangle Research International to describe its family of Ladder and Ladder+BASIC program editor, compiler, simulator and program up-loader software.

The Internet TRiLOGI 6.x client/server suite is written to run under all currently available 32-bit Microsoft Windows operating systems as Windows 98, Me, NT, 2000, XP, and Vista.

By installing the 32-bit Java Runtime environment: JRE 1.4.2_19 that is shipped with the I-TRiLOGI CD-ROM or download website, I-TRiLOGI has also been tested to run properly under 64-bit Windows 7 / 8 operating systems.

1.2 Download the Latest Version of I-TRiLOGI

If this is the first time you are installing I-TRiLOGI, you should locate and open the “upgrade.htm” document in the I-TRiLOGI CD-ROM which provides the web link, username and password to login to the TRi website to download the latest version.

If you already have a current version of I-TRiLOGI, you can click on the “Help” menu on I-TRiLOGI and select the “Upgrade TRiLOGI” command which will also open up the “Upgrade.htm” document. Either case, please login to the upgrade website and download the latest version of SetupTL6xxx.exe. Then proceed to Section 1.3 to install the JRE first BEFORE installing the I-TRiLOGI.

1.3 Install Java Run Time Enviroment JRE 1.4.2_19

1. In the root directory on the Internet TRiLOGI 6.x CD-ROM, you should find a folder “x86-Windows” which is where all the setup files for PCs running MS Windows are located.

i-TRiLOGI is written in Java and requires a Java Run Time Environment (JRE) installed on the PC in order to run properly. The i-TRiLOGI 6.xx CD ROM includes a copy of the JRE 1.4.2_19 setup program with the filename:

j2re-1_4_2_19-windows-i586-p.exe

Although your PC may already have a later version of JRE (in 2012 the latest JRE is version 7), it was discovered the JRE 1.4.2_19 is proven to work best with i-TRiLOGI software. This is especially so if your PC has a 64-bit Windows and some users reported that the JRE 7 on their 64-bit Windows did not work properly with I-TRiLOGI.

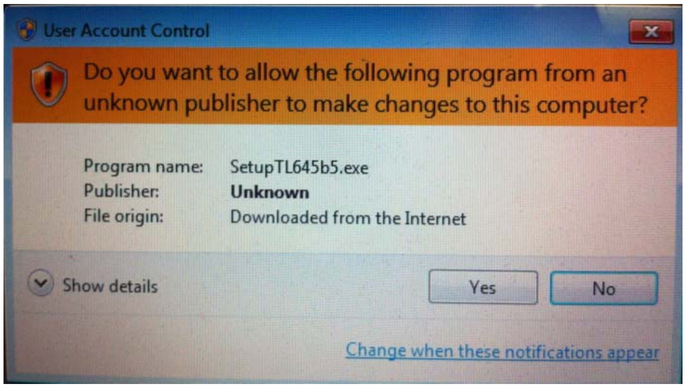
You can install JRE 1.4.2_19 even if your PC already have other JRE versions. I-TRiLOGI software will automatically invoke JRE 1.4.2_19 if it is installed on your PC. JRE must be installed **BEFORE** installing the i-TRiLOGI software. First, double-click on the file "j2re-1_4_2_19-windows-i586-p.exe" to install the JRE. Please follow all instructions provided by the Install Shield program and install it onto the given default path (which is C:\Program Files\Java\ on 32-bit Windows and C:\Program Files (x86)\Java on 64-bit Windows.

1.4 Install i-TRiLOGI 6.x

The following steps can be taken to install the i-TRiLOGI software:

1. After you have installed JRE 1.4.2_19, then double-click on the "SetupTL6.exe" file (or the "Setup6xx.exe" file if you have downloaded the latest upgrade) to install I-TRiLOGI.

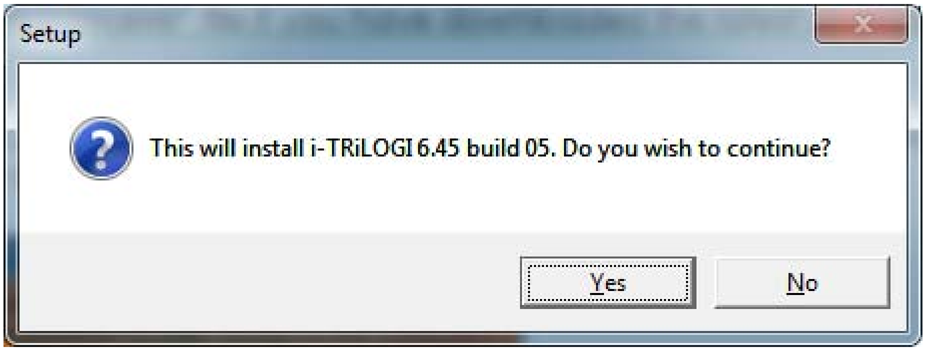
2. On any PC with an operating system that has user account controls, the first installation step will likely be a warning about an unknown publisher, as per the below screenshot. If this appears, click 'Yes' to continue.


User Account Control

Do you want to allow the following program from an unknown publisher to make changes to this computer?

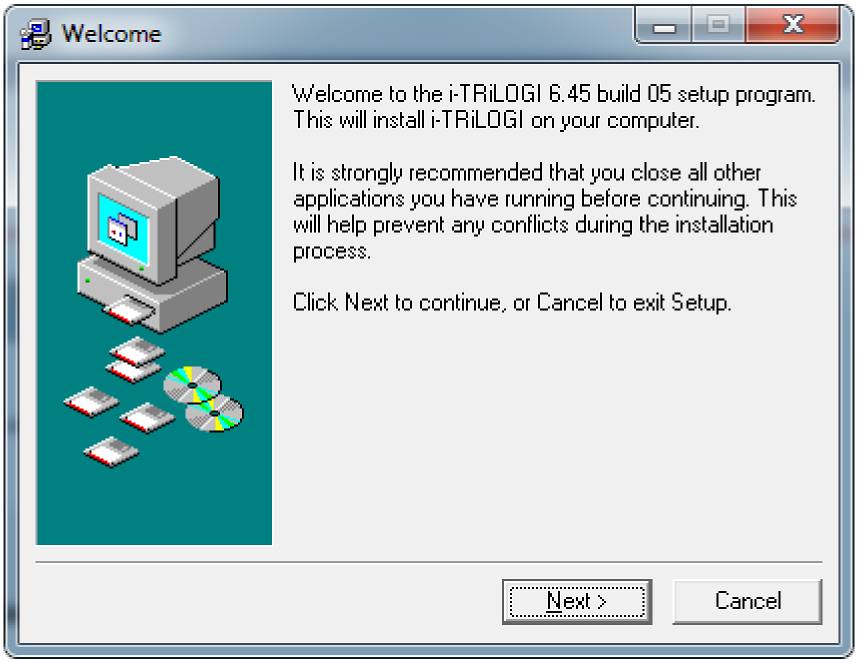
Program name: SetupTL645b5.exe
Publisher: **Unknown**
File origin: Downloaded from the Internet

3. Next you will be asked if you want to install the indicated version of i-TRiLOGI. Again, click 'Yes' to continue.


Setup

This will install i-TRiLOGI 6.45 build 05. Do you wish to continue?

4. Next you will see the first step in the installation process that recommends closing open programs. Click ‘Next’ to continue.

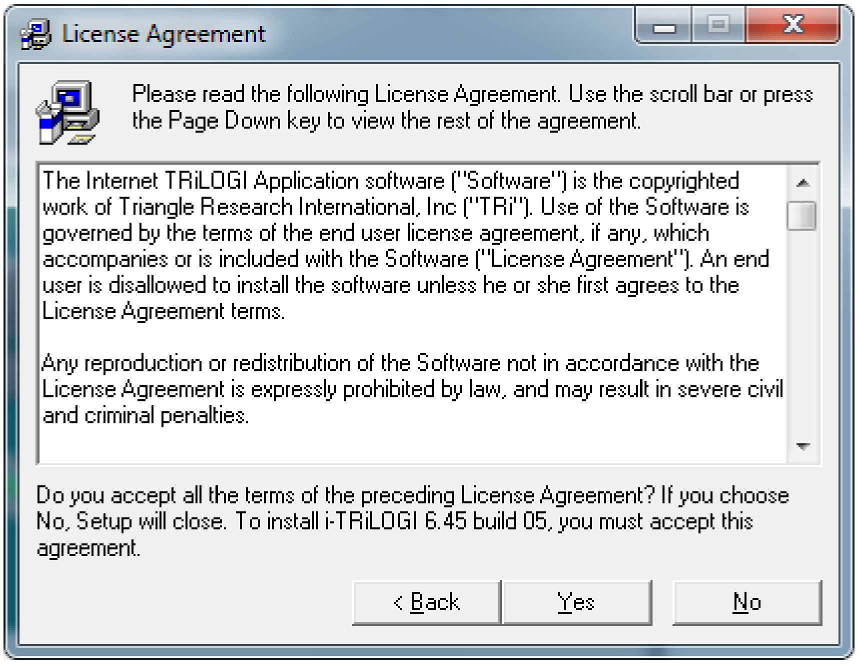

Welcome

Welcome to the i-TRiLOGI 6.45 build 05 setup program. This will install i-TRiLOGI on your computer.

It is strongly recommended that you close all other applications you have running before continuing. This will help prevent any conflicts during the installation process.

Click Next to continue, or Cancel to exit Setup.

5. The next step is to accept the license agreement. Click ‘Yes’ to accept it and continue.



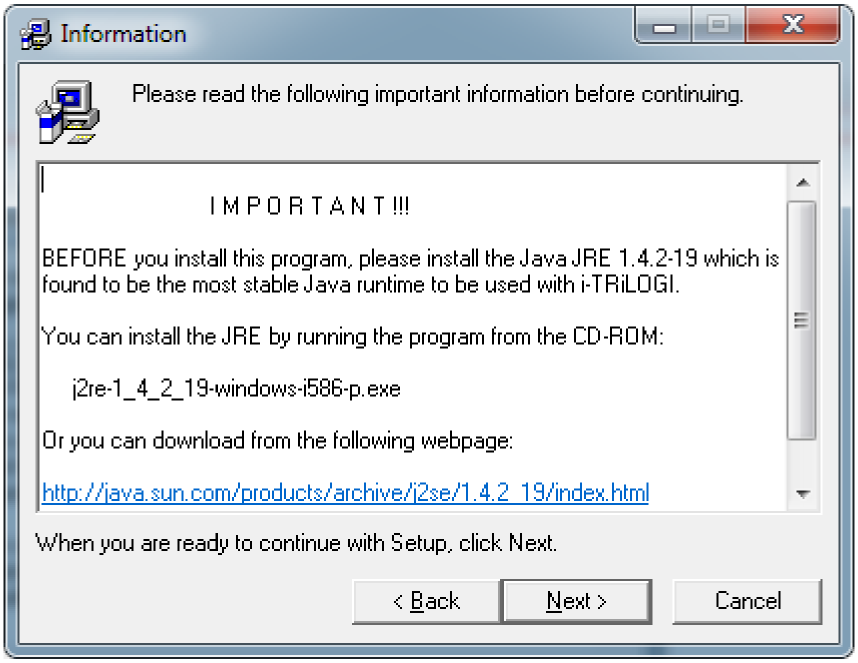
Please read the following License Agreement. Use the scroll bar or press the Page Down key to view the rest of the agreement.

The Internet TRiLOGI Application software ("Software") is the copyrighted work of Triangle Research International, Inc ("TRi"). Use of the Software is governed by the terms of the end user license agreement, if any, which accompanies or is included with the Software ("License Agreement"). An end user is disallowed to install the software unless he or she first agrees to the License Agreement terms.

Any reproduction or redistribution of the Software not in accordance with the License Agreement is expressly prohibited by law, and may result in severe civil and criminal penalties.

Do you accept all the terms of the preceding License Agreement? If you choose No, Setup will close. To install i-TRiLOGI 6.45 build 05, you must accept this agreement.

6. Then you must confirm that the recommended version of Java is already installed (version 1.4.2). If it is installed, click 'Next' to continue. Otherwise, cancel the installation and see section 1.3 Install Java Run Time Enviroment JRE 1.4.2_19



Please read the following important information before continuing.

IMPORTANT!!!

BEFORE you install this program, please install the Java JRE 1.4.2-19 which is found to be the most stable Java runtime to be used with i-TRiLOGI.

You can install the JRE by running the program from the CD-ROM:

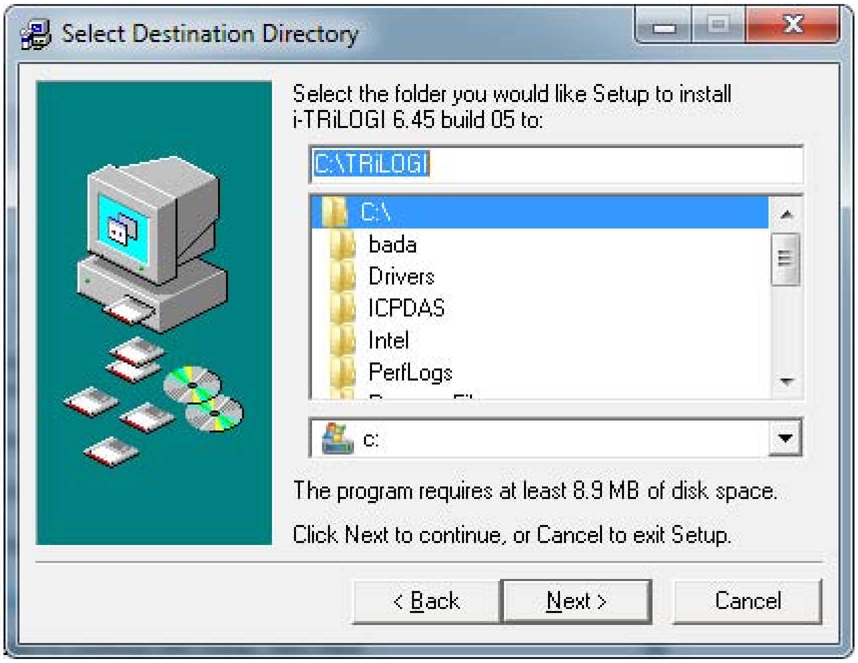
j2re-1_4_2_19-windows-i586-p.exe

Or you can download from the following webpage:

http://java.sun.com/products/archive/j2se/1.4.2_19/index.html

When you are ready to continue with Setup, click Next.

7. The next step is to select the installation folder. It is recommended to use the default option, which is "C:\TRiLOGI"



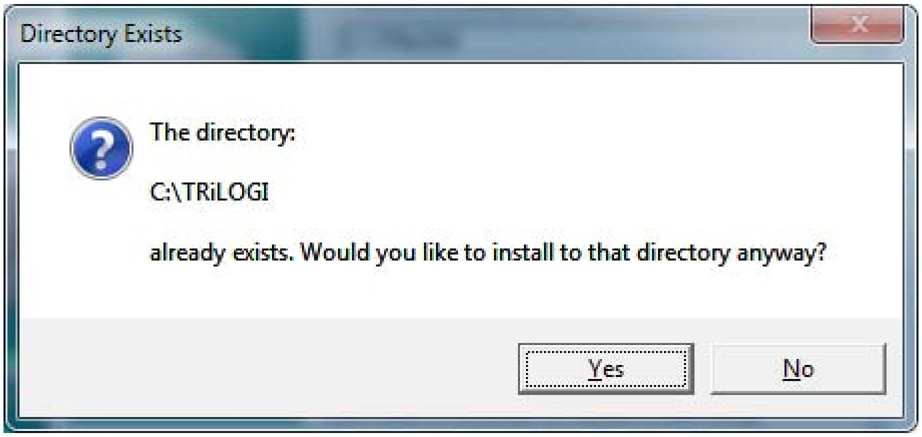
Select the folder you would like Setup to install i-TRiLOGI 6.45 build 05 to:

C:\TRiLOGI

The program requires at least 8.9 MB of disk space.

Click Next to continue, or Cancel to exit Setup.

8. If this installation is an upgrade, then the installation folder may already exist and you will get prompted to use the same folder or choose a new one. If you don't see this, skip to the next step. Otherwise, click 'Yes' to use the same installation folder.

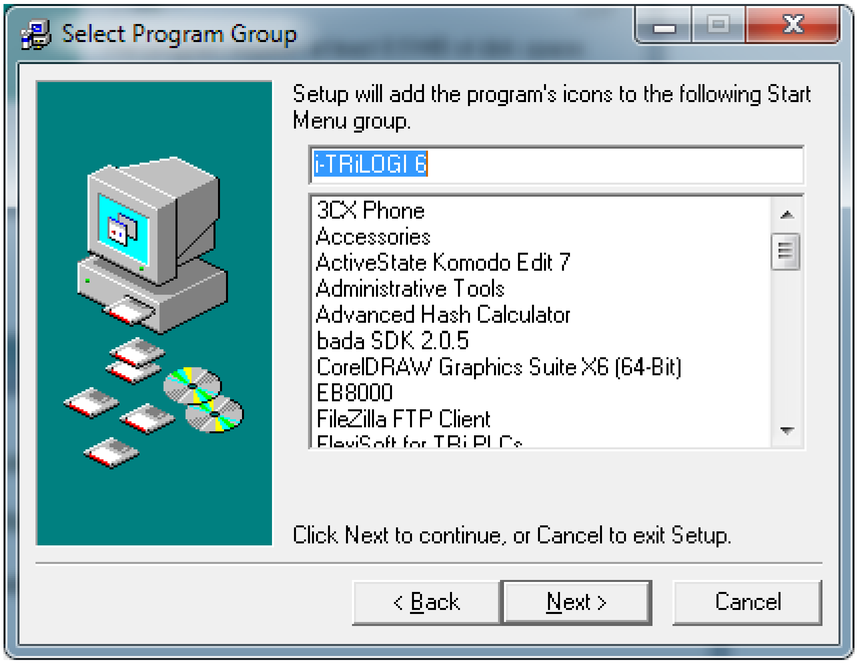


The directory:

C:\TRiLOGI

already exists. Would you like to install to that directory anyway?

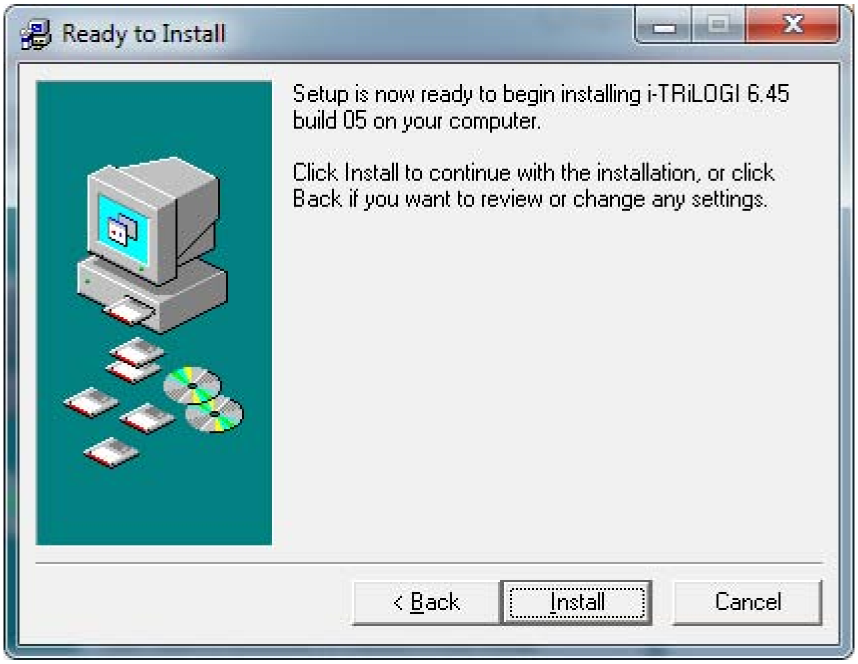
9. Now you will be asked to choose a name for the start menu group, which is what you will see in the PCs start menu program listing. Click 'Next' to use the default option.



Setup will add the program's icons to the following Start Menu group.

Click Next to continue, or Cancel to exit Setup.

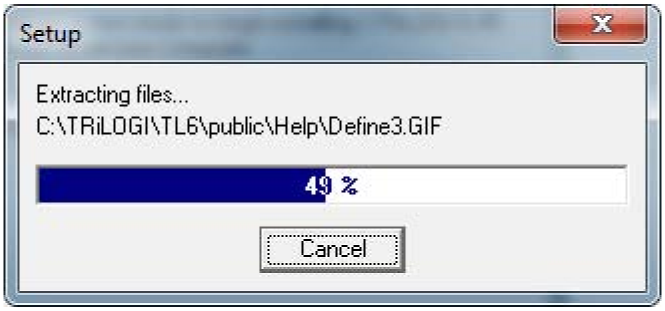
10. Now the installation setup is complete and you just need to click 'Install' to start the installation process.



Setup is now ready to begin installing i-TRiLOGI 6.45 build 05 on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

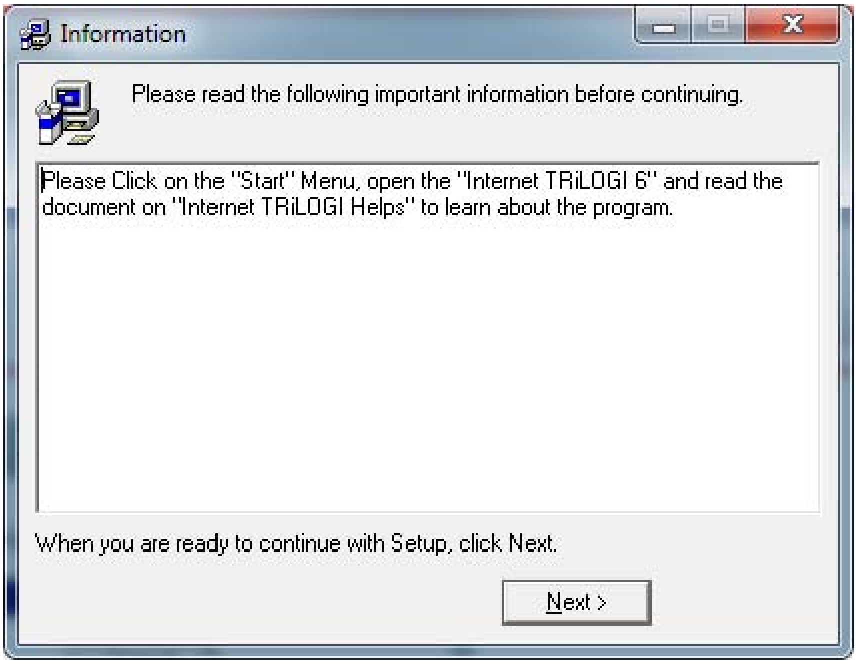
11. Installation is taking place. Wait for the Setup to complete automatically.



Extracting files...
C:\TRiLOGI\TL6\public\Help\Define3.GIF

49 %

12. Installation is nearly complete. An information box will appear and after reading it, click ‘Next’ to continue.

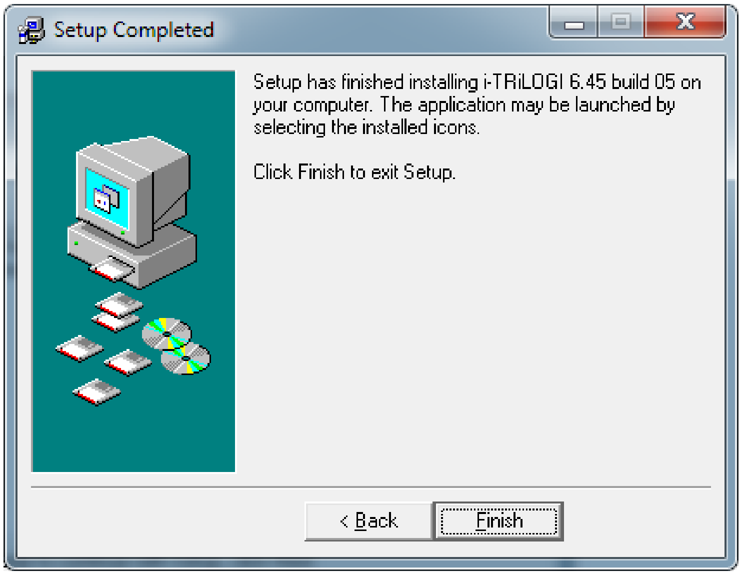


Please read the following important information before continuing.

Please Click on the "Start" Menu, open the "Internet TRiLOGI 6" and read the document on "Internet TRiLOGI Helps" to learn about the program.

When you are ready to continue with Setup, click Next.

13. Finally, installation is complete. Click ‘Finish’ to close the window and proceed to the next section of this manual for instructions on running TRiLOGI and TLServer.



Setup has finished installing i-TRiLOGI 6.45 build 05 on your computer. The application may be launched by selecting the installed icons.

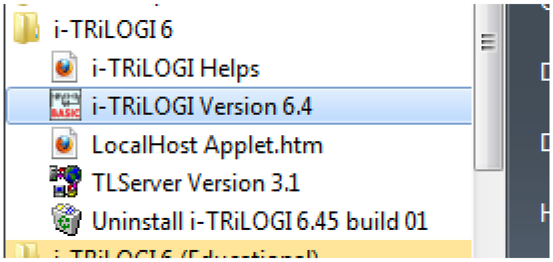
Click Finish to exit Setup.

1.5 Run TRiLOGI and TLServer

All i-TRiLOGI Version 6.x files will be installed in the following default folder:

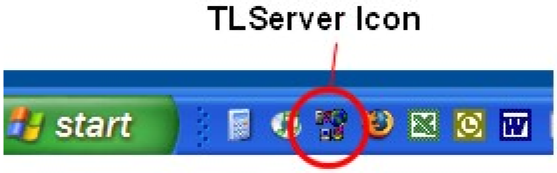
"C:\TRiLOGI\TL6"

You normally would not need to go directly to this folder to run TRiLOGI or TLServer. This is because during installation of i-TRiLOGI, a program Group folder "i-TRiLOGI 6.xx" will be created in the Start Menu to provide short cuts to the TLServer program, the i-TRiLOGI application and the TL6 Applet starter, as illustrated in the following picture.



In the example screenshot above, the TRiLOGI programming environment can be started by selecting “i-TRiLOGI Version 6.4” from the option list. TLServer can be started by selecting “TLServer Version 3.1”

Note: A short cut for TLServer is also created on the “Quick Launch” toolbar. For Windows XP user, you may have to right click on the tool bar area along the bottom of the screen and check the "Show Quick Launch" option in order for the quick launch tool bar to appear:



1.6 Using TRiLOGI 6.x with International Languages

The main improvement in the new i-TRiLOGI version 6.x and its accompanying TLServer 3.x is that in version 6.x, all data file storage and communication are conducted using the international standard "Unicode" (UTF-16) instead of the ASCII code found in the earlier versions of TRiLOGI software. When used with Unicode-aware Operating System such as the Windows 2000 and XP, it is now possible to create program that uses non-Latin based characters such as the Chinese, Japanese, Korean (also known as CJK), Thai, Hebrew, and Arabic etc in their comments and I/O label fields. In addition, version 6.x also includes some enhanced capabilities such as a EEPROM manager that allows reading/writing of the PLC's data EEPROM directly within the TRiLOGI software.

Internet TRiLOGI version 6.x and TLServer 3.x also allow users to customize their menu and on screen display so that the menu, screen display and some online help texts may be shown in other international languages including the CJK. There is even an option in the language text file for you to increase the text size by several points to make it clearer to display some international characters such as Chinese, or simply to make it easier to read the menu and help text.

Several language files are currently available for download from the I-TRiLOGI upgrade website:

Language	ISO Abbreviation	Language specific file
English	en	en_language.txt
Spanish	es	es_language.txt
Chinese	Zh	zh_language.txt

Users of other international languages can create their own specific language help text by copying the "en_language.txt" file into a file named as "xx_language.txt" where xx is the two character ISO abbreviation of the language. For example, to create a Korean language help text, you need to copy the "en_language.txt" file into a file name: "ko_language.txt" and then edit the file to translate the relevant English text into Korean language. Even

English language user may modify the content of their pull down menu and on screen display by modifying the "en_language.txt" file. You can open some of the supplied language files (e.g. English vs. Spanish) to understand how the English language items are translated to another language.

When the i-TRiLOGI version 6.x or TLServer 3.x program starts up, the program will automatically try to load the specific language file based on the locale of the user's operating system and use the file content throughout the session. However, if the program fails to find the corresponding language text file, then it will use a set of internally defined English language text strings for its entire menu and on screen display.

For English locale user, all i-TRiLOGI 6.x help files are stored in the following folder: "c:\TRiLOGI\TL6\public\Help". However, if you start TRiLOGI in locale xx, then when you attempt to open an online help file, TRiLOGI will first check if there exist a folder "c:\TRiLOGI\TL6\public\xx_Help". If it finds it then the help files will be loaded from this locale-specific folder. But if it cannot find the "xx_Help" folder it will then load the help files from the default folder: "c:\TRiLOGI\TL6\public\Help". Hence for those who wish to experiment with providing their own locale specific help files, they should copy all the help files from the default folder into the new locale-specific "xx_Help" folder and then make changes to files in the new folder. When they press <F1> in TRiLOGI the help files will be loaded from the "xx_Help" folder.

Chapter 2 Introduction to i-TRiLOGI Client/Server Architecture

2 INTRODUCTION TO I-TRILOGI CLIENT/SERVER

2.1 Client/Server Architecture

i-TRiLOGI is a **Client/Server** application suite. The entire program is broken into two parts: the Server and the Client.

2.1.1 Client

The i-TRiLOGI program is the one which you use to create your ladder logic + TBASIC program and is called the "Client" program. (If you are programming the PLC offline then you only need to run the client program without the server.) The beauty of the client/server configuration is that it does not matter whether the server and client are located at the same computer or at 10,000 miles apart and they work exactly the same way. The client and the server can communicate via any form of network connection, including the Internet. This makes it possible for the user to program the PLCs either locally or remotely via the Internet or even wirelessly via mobile Internet.

Another important advantage of client/server architecture is that multiple clients may access the same server simultaneously. Hence you can run multiple copies of the i-TRiLOGI clients at different places around the world simultaneously for troubleshooting a single PLC. You can also run the i-TRiLOGI client AND the TRi-ExcelLink (separate data collection software) clients simultaneously!

2.1.2 Server

Since i-TRiLOGI client program only communicates using TCP/IP networking protocol, it needs to connect to a "server" that listens on a TCP/IP port waiting for connections from i-TRiLOGI or other client programs.

FServer

Most of the newer TRI's brand of PLC, such as Nano-10, FMD and F-series PLCs have a built-in web server called "FServer" that is located on the PLC CPU chip, and the i-TRiLOGI client can connect to it directly for programming and online monitoring.

The FServer can perform other tasks that a typical web server can do, such as host web pages and receive connections from other devices/software. Please refer to the PLC's User's Manual for details.

TLServer

All TRi ‘Super’ PLCs (including legacy PLCs that do not have built-in Ethernet port, such as the T100MD+ and T100MX+ family) can still be programmed using the same i-TRiLOGI software via a PC-based server software called "TLServer" that converts TCP/IP packets to serial communication (RS232/RS485) .

TLServer behaves like a typical web server and it is capable of serving HTML webpages as well as Java Applet to an Internet Browser such as Firefox, Chrome or Internet Explorer. TLServer connects to the PLCs via the PC's serial communication port and it is the one responsible for conveying communication messages between the i-TRiLOGI client and the ‘Super’ PLCs. (Note: TLServer is not included with Education version of i-TRiLOGI since there is no real PLC involved.)

Therefore if your PLC does not have Ethernet connection, then you must run TLServer in order to program or configure the PLC. To Start TLServer, double-click on its Icon and a TLServer panel will appear. You can minimize TLServer but it should be actively running in order to service network requests from i-TRiLOGI via the Internet or local area network.

2.2 i-TRiLOGI Client Software Versions

i-TRiLOGI client software is available in two versions:

2.2.1 Java Application

The i-TRiLOGI program (as well as the JVM, see below) must be locally installed in the PC that it runs on.

2.2.2 Java Applet

The client computer only needs to use a Java-enabled Web browser such as the Internet Explorer 6.x or Mozilla Fire Fox to invoke the i-TRiLOGI applet that is stored in the TLServer. There is no need to install the i-TRiLOGI software in the local computer.

2.3 i-TRiLOGI Application vs Applet: Which is Better?

	Pros	**Cons**
Application	• Starts up immediately. • Can read/write TRiLOGI files to local hard disk or to TLServer. • Can access any TLServer on the network. • Program behavior predictable since the copy of JVM is local.	• Require local installation of TRiLOGI software at every client computer. • Require installation of JVM at every client computer. • May need to specify the proxy server IP address if running behind a firewall.
Applet	• No need to install any software or JVM at the client computer. • Possible to control your PLC via any CyberCafe or at a friend's house! • Maintenance and Upgrading of software is simple since only one copy of the TL6x.jar file needs to be changed. • Centralised storage of program files only at the server. This is good for providing PLC program training.	• Can only read/write TRiLOGI files to the TLServer but not to the local hard disk. • Can only access the TLServer from which it was loaded. • May take a few minutes to load itself the first time if connect via dial up modem. (Thereafter the browser should cache it for rapid start up.) • Program behavior may vary for different make or different versions of the browser.

Chapter 3: Interfacing to the PLC

3 INTERFACING TO THE PLC

This chapter will discuss how to interface to the 'Super' PLC in terms of the most direct physical connection and the necessary software components for the purpose of programming and monitoring the PLC.

These procedures will be covered in a bit of depth to explain certain aspects, so you may wish to reference the quick-connect guides instead, which are in the appendix of this document, if you would like simple connection directions without explanations.

Physical Connection

There are two direct ways to physically connect to 'Super' PLCs for programming and monitoring from the i-TRiLOGI software:

1) SERIAL PORT (RS232 / RS485)
2) ETHERNET (Wired, RJ45 terminated)

NOTE: There are other indirect ways to physically interface to the PLC, such as serial/Ethernet radio, serial/Ethernet modem, wireless Ethernet bridge, XBEE radio, and others. Many of these are described further in subsequent sections of this document, independent documents, or the individual PLC reference manuals.

Software Interface

The i-TRiLOGI programming environment (client) is required to interface to the PLC for programming and online monitoring whether the physical connection is serial or Ethernet, but the server could be either TLServer or the FServer depending on the PLC model and type of connection.

3.1 Physical Connection

Three methods of physical connection to the PLC are described here: Serial RS232, Serial RS485, and Ethernet.

3.1.1 Serial Port

All TRi 'Super' PLCs have at least one RS485 port and all but the Nano-10 have a RS232 port. Either of these ports can be used to connect a PC to the PLC for programming or monitoring.

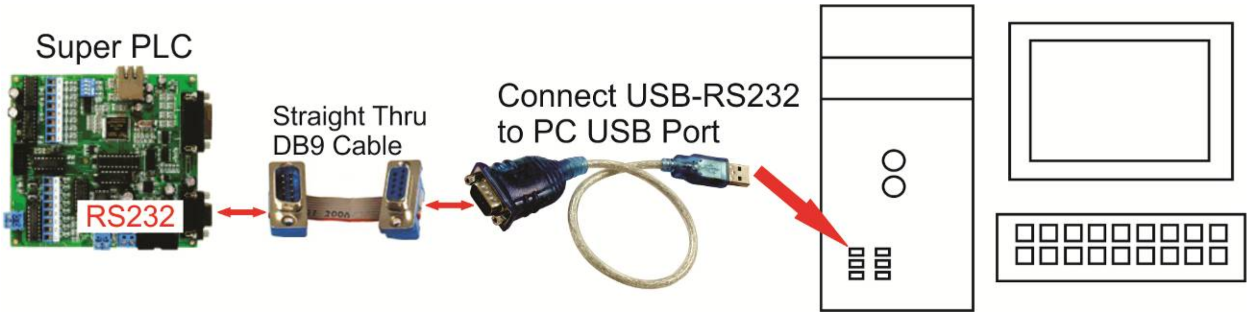
RS232

TRi 'Super' PLCs with a DB9 serial port (9-pin connector) can interface via RS232 directly to a PC if it has the same type of port or indirectly through a USB port. Most modern PCs now only have USB serial ports, so this would be the only option.

For USB connection, you will need a USB to RS232 adapter that plugs into the USB port of a PC and the DB9 serial port of the PLC. Typically a straight through DB9 cable will need to be added to the adapter to extend the connection since USB to RS232 adapters are fairly short.

Here is a link to a known compatible USB to RS232 adapter supplied by Triangle Research:

http://www.tri-plc.com/USB-RS232.htm



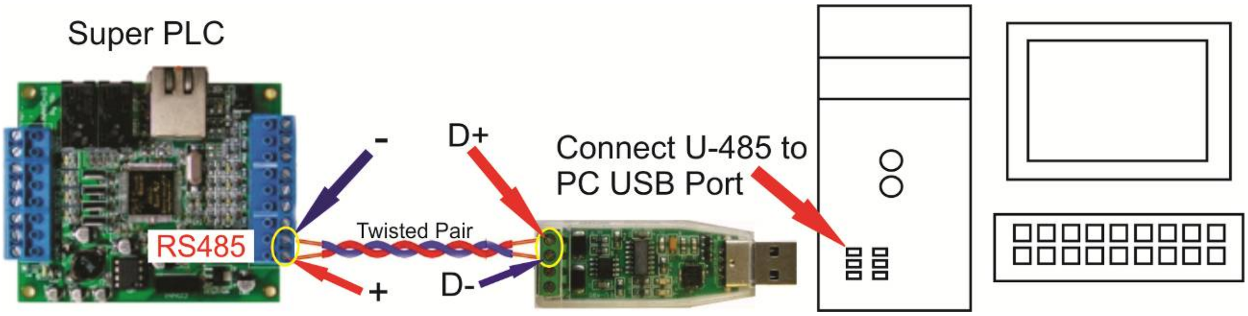
RS485

All TRi 'Super' PLCs have a two-wire RS485 serial port (screw terminal connector) and can only interface to a PC via RS485 indirectly through a USB port.

For USB connection, you will need a USB to RS485 adapter that plugs into the USB port of a PC and has a two wire connection to the PLC (usually labeled A & B or D+ & D-). Twisted pair cabling (shielded for noisy environments) should be added to the adapter to make a connection since USB to RS485 adapters would not typically provide wire connection.

Here is a link to a known compatible USB to RS485 adapter supplied by Triangle Research:

http://www.tri-plc.com/U-485.htm



3.1.2 Ethernet

Introduction

All TRi Nano-10, FMD series, F-series, and newer PLCs have an Ethernet port built in, which can be used for programming and monitoring with the TRiLOGI software, as well as communication with other Ethernet devices and software.

These PLCs can connect to the PC running TRiLOGI many ways as follows:

a) Wired connection to a router that the PC is also connected to (PC connection can be wired or wireless).
b) Direct connection to the PCs wired Ethernet port via crossover cable
c) Wireless connection to a wireless router if the PLC is connected to a wireless bridge (adapter)

Only the first option, which is most common, will be described here.

In a typical local area network (LAN) there would be one router (wired or both wired and wireless) that the network devices connect to, one modem that provides Internet to the router, and the devices connected to the router (such as the PLC and PC).

Before You Begin

The first thing you need to do is configure the network settings in the PLC to match those of the LAN. This is typically done as follows:

1) Find out what your routers gateway address is (typically 192.168.1.1 or 192.168.0.1) and what static IP addresses are free to use with your PLC.

   NOTE: if the routers gateway address is "192.168.1.1", the default PLC IP address (192.168.1.5) will most likely work unless it is already used by another device on the same network. If it is free to use, the next two steps can be skipped as the PLC will already be able to connect to the LAN.

2) Connect to the PLCs serial port from the PC with TRiLOGI and TLServer.

3) Edit the PLC network settings using the Ethernet & ADC Configuration tool from the "Controller" menu in TRiLOGI. Only the IP address is necessary to configure for basic connection to the LAN.

Please refer to the Quick Connection guide in appendix 1 to connect the PLC to your PC via serial port.

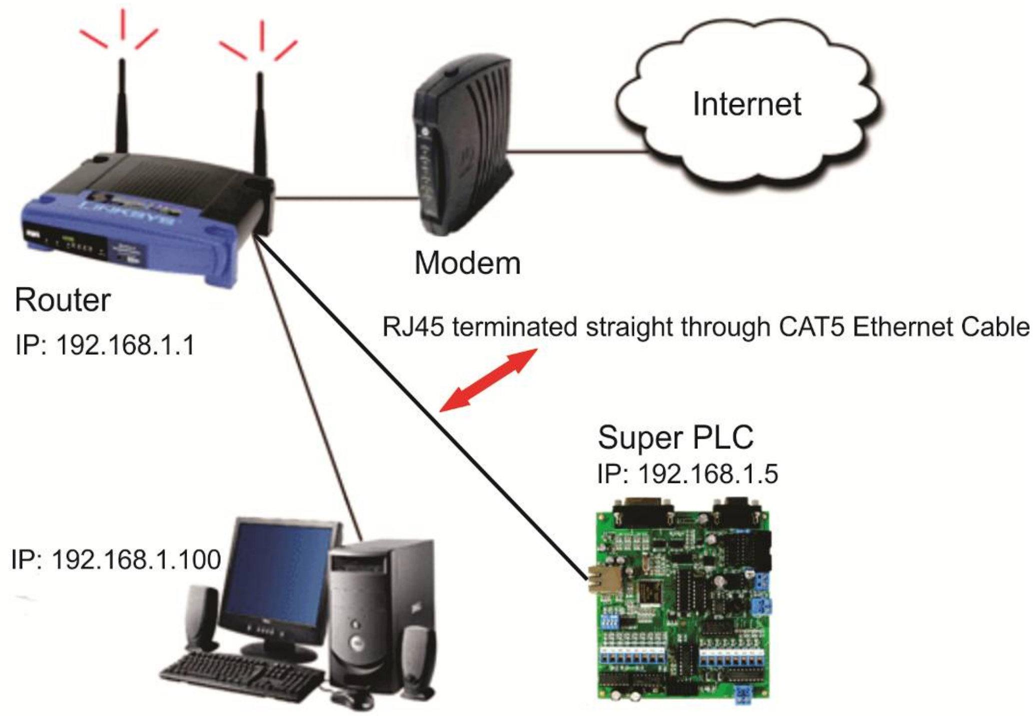
Please refer to chapter 2 of the PLCs user manual for more detailed information on network configuration.

Network Wiring

You will likely already have a network available that consists of an Internet modem that provides Internet to a wired or wireless router, which has at least one PC connected to it.

In this case, all that needs to be done is connect the PLC to the router with standard Ethernet cable. The final network configuration will likely resemble the below simple network diagram.

If you are on a corporate network, then you will need to consult the IT administrator to get the PLC connected to the network.



3.2 Software Interface

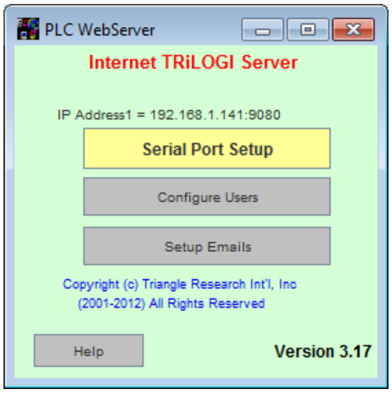
The first section will describe TLServer setup, which is used for serial port communication only. If you will be connecting via Ethernet, you can skip the TLServer section and go straight to TRiLOGI Communication.

3.2.1 TLServer Setup

TLServer is the server program that provides serial connection to the PLC and TCP/IP connection to the TRiLOGI programming software. It is described in more detail in Chapter 2 Introduction to i-TRiLOGI Client/Server Architecture.

TLServer is not needed for TCP/IP communication to the PLCs Ethernet port, only for serial communication.

1) First you will need to start **TLServer**, which can be done from the start menu or quick launch. See section 1.5 Run TRiLOGI and TLServer for more details on starting TLServer. The below-left window will open:

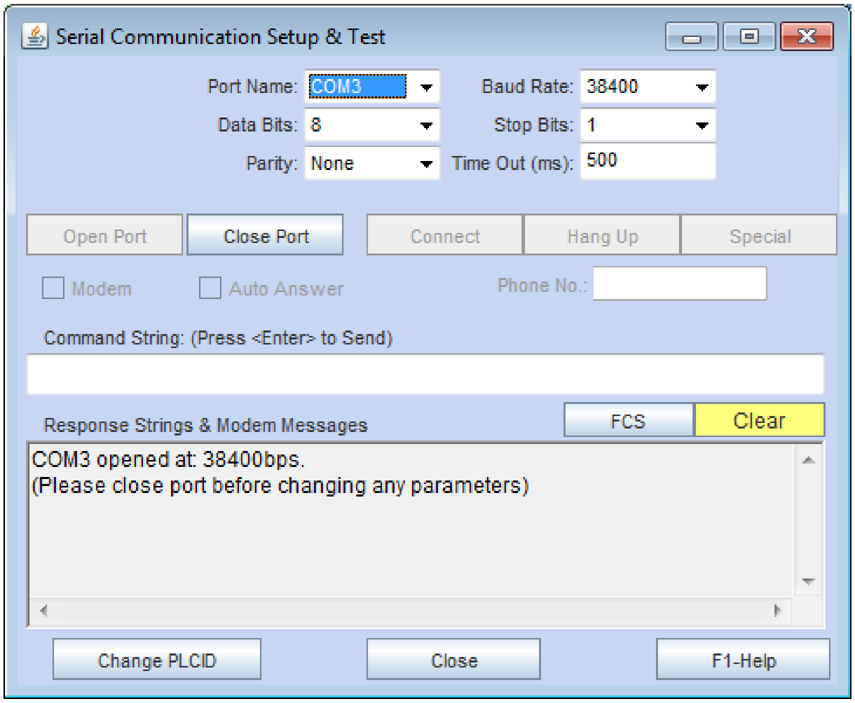

PLC WebServer

Internet TRiLOGI Server

IP Address1 = 192.168.1.141:9080

Copyright (c) Triangle Research Int'l, Inc
(2001-2012) All Rights Reserved

Version 3.17


Serial Communication Setup & Test

Port Name: COM3
Baud Rate: 38400
Data Bits: 8
Stop Bits: 1
Parity: None
Time Out (ms): 500

Command String: (Press <Enter> to Send)

Response Strings & Modem Messages

```
COM3 opened at: 38400bps.
(Please close port before changing any parameters)
```

2) Next you need to configure the serial port settings by clicking on **Serial Port Setup**, and the above-right window will open.

The default serial port settings are as follows:

Baud Rate	:	38400 bps
Data Bits	:	8
Stop Bits	:	1
Parity	:	None (N)
Time Out	:	500ms

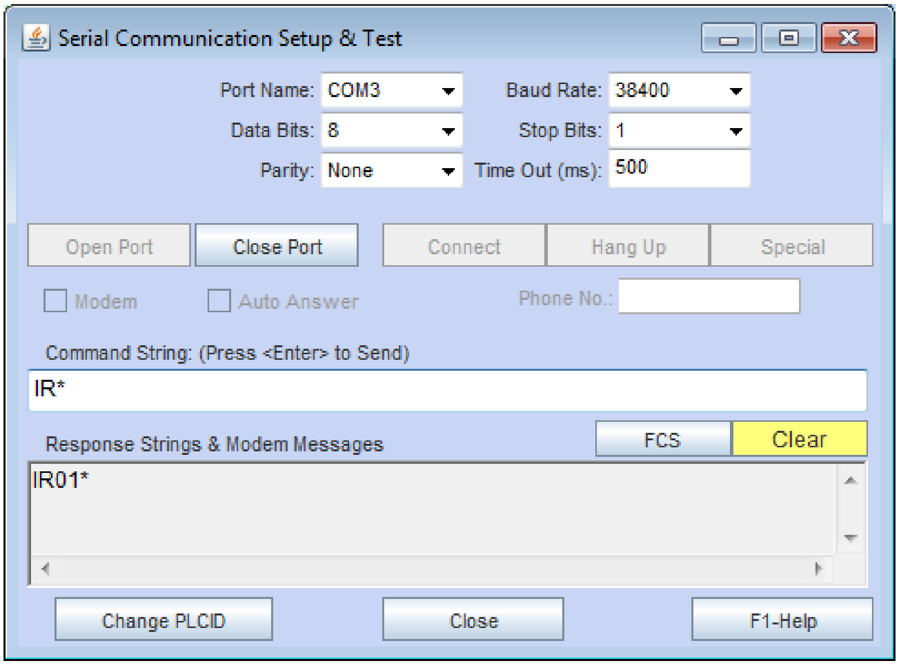
These settings match the default com settings for all ‘Super’ PLC serial ports, so they don’t need to be changed unless they have been specifically modified by a PLC program transferred to the PLC after purchase.

3) The only setting that needs to be checked and possibly modified is the **Port Name**. This is the com port that the PC uses to connect to the PLC.

Select the COM port that the USB adapter is installed on (or the built-in RS232 port if your PC has one) from the drop-down menu. Note that the USB adapter would need to be installed properly before starting TLServer or it wouldn’t be available from the drop-down list.

4) You can test the COM setup by entering a Host Link command in the “Command String” field. The following example is a good basic command to

test communication.



Port Name: COM3
Baud Rate: 38400
Data Bits: 8
Stop Bits: 1
Parity: None
Time Out (ms): 500

Command String: (Press <Enter> to Send)

IR*

Response Strings & Modem Messages

IR01*

If you have only one PLC connected to your TLServer computer, then you can test the communication now using the following command string:

Command String : **IR***
Response String : **IR01***

The response string tells you that the ID address of this single PLC is 01, which is default. If the ID was changed, then it will be shown in place of 01.

If you get the message **(Warning: No Response From PLC!)**, then communication is not setup properly. First repeat the steps from PLC connection to TLServer setup and if there is still no response, refer to the TROUBLESHOOTING SERIAL COMMUNICATION Appendix at the end of this document.

If you received the correct response, then setup is complete and communication is confirmed, so you can proceed to the next section to get communicating from TRiLOGI.

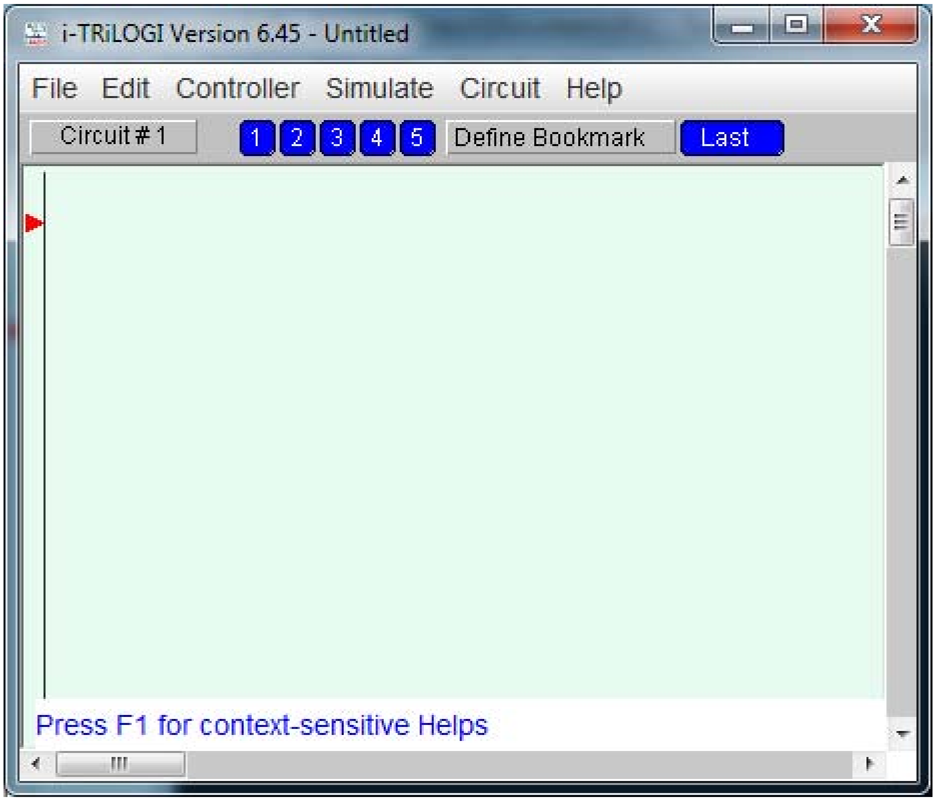
3.2.2 TRiLOGI Communication

TRiLOGI is the client program that is used as the PLC programming environment and to perform the two main types of communication from TRiLOGI:

a) Program Transfer
b) Online Monitoring

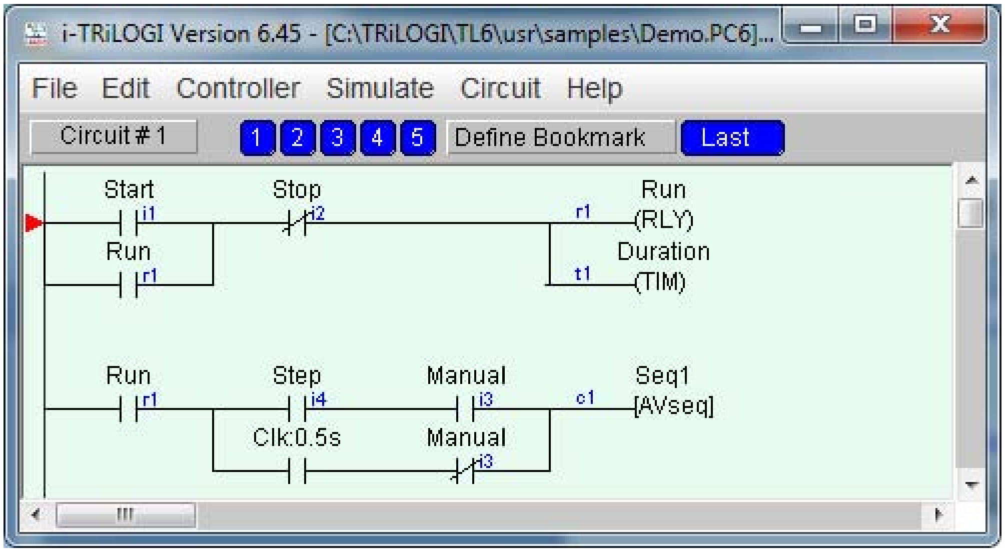
Whether you are connected to the PLC via serial port or Ethernet, you can now perform both of these tasks. If you have connected via serial port, then TRiLOGI will connect to TLServer (must be running on the PC connected to the PLC). If you have connected via Ethernet, then TRiLOGI will connect to the PLCs web server, which is called FServer.

First you will need to start **TRiLOGI**, which can be done from the Start menu. See section 1.5 Run TRiLOGI and TLServer for more details on starting TRiLOGI. The below window will open:



Press F1 for context-sensitive Helps

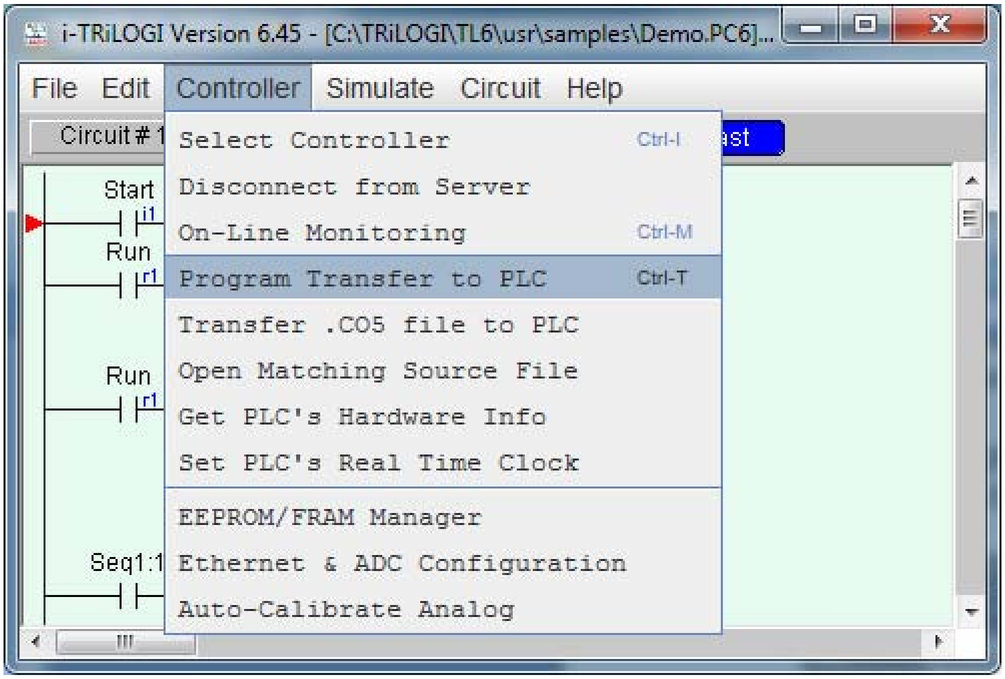
For the purposes of this tutorial, the Demo.PC6 program will be referenced. This is a sample program that is included in the i-TRiLOGI installation. This sample program, as well as all the other included sample programs, are accessible from the "**C:\TRiLOGI\TL6\usr\samples**" folder. Below is the Demo.PC6 program:



Program Transfer

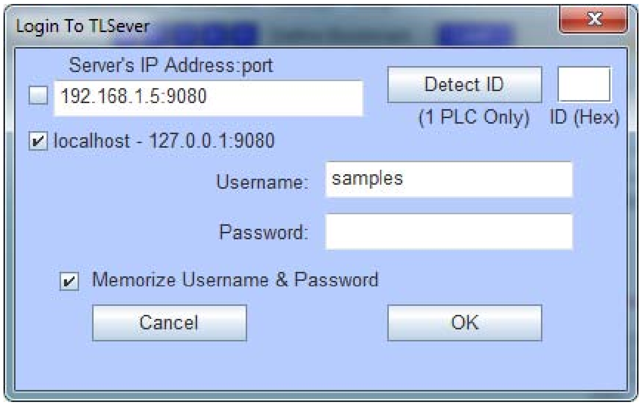
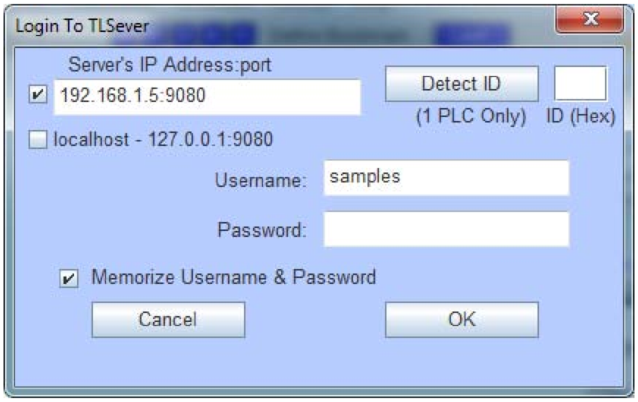
All steps are the same whether the PLC is connected via serial port or Ethernet unless otherwise specified.

1. **Select "Program Transfer to PLC"** from the "Controller" menu (or press CTRL + T on the keyboard).


i-TRiLOGI Version 6.45 - [C:\TRiLOGI\TL6\usr\samples\Demo.PC6]...
File
Edit
Controller
Simulate
Circuit
Help
Circuit # 1
Select Controller Ctrl-I
Disconnect from Server
On-Line Monitoring Ctrl-M
Program Transfer to PLC Ctrl-T
Transfer .CO5 file to PLC
Open Matching Source File
Get PLC's Hardware Info
Set PLC's Real Time Clock
EEPROM/FRAM Manager
Ethernet & ADC Configuration
Auto-Calibrate Analog
Start
i1
Run
r1
Run
r1
Seq1:1

2. The "Login to TLServer" window will open as shown below. Although it mentions TLServer, it is actually used to login to either server (TLServer or

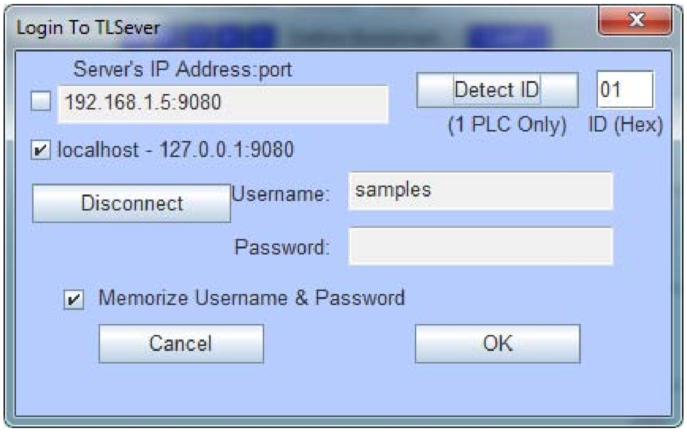
FServer).

Connect via serial port to TLServer	Connect via Ethernet to FServer
 Login To TLSever Server's IP Address:port ☐ 192.168.1.5:9080 ☑ localhost - 127.0.0.1:9080 Username: samples Password: ☑ Memorize Username & Password 	 Login To TLSever Server's IP Address:port ☑ 192.168.1.5:9080 ☐ localhost - 127.0.0.1:9080 Username: samples Password: ☑ Memorize Username & Password
<u>Default Settings:</u> **Server IP** : localhost – 127.0.0.1:9080 **Username** : “samples” **Password** : none (blank field)	<u>Default Settings:</u> **Server IP** : 192.168.1.5:9080 **Username** : none (ignored) **Password** : none (ignored)

The above-left screenshot is the default configuration, which is set to login to TLServer using the “**localhost – 127.0.0.1:9080**” option. The default Username is “**samples**” with no password required. Use this configuration if you are connected via serial port and if TRiLOGI is running on the same PC as TLServer.

The above-right screenshot is an example of a configuration used to connect to FServer using a specified IP address and port number. In this example, the default PLC IP address (192.168.1.5) and port number (9080) is specified, but these should be replaced by the actual IP address and port number configured in your PLC. There is no username and password set for Ethernet communication by default, so these fields can contain anything since they will be ignored.

3. After configuring the login settings, you will need to **click “Detect ID”** and the PLC ID should be returned in the adjacent field, as shown below, if communication is functioning correctly.



Login To TLSever

Server's IP Address:port

☐ 192.168.1.5:9080

☑ localhost - 127.0.0.1:9080

Username: samples

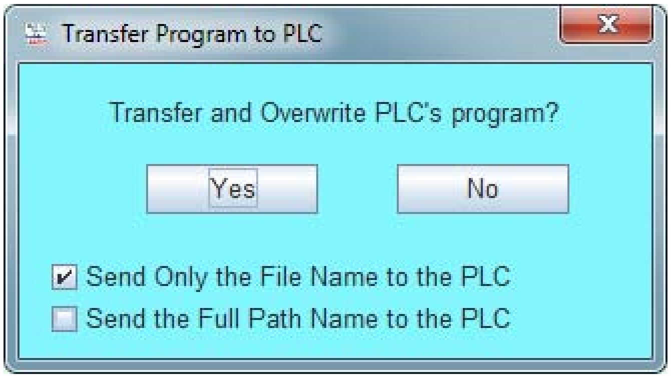
Password:

☑ Memorize Username & Password

ID (Hex): 01

4. **Click OK** to complete the server login process.

5. **Click 'Yes'** to continue when prompted with the program transfer confirmation window:



Transfer Program to PLC

Transfer and Overwrite PLC's program?

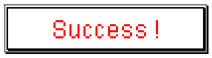
☑ Send Only the File Name to the PLC

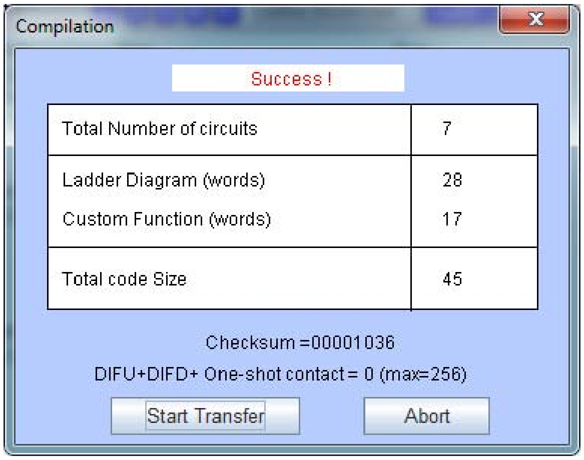
☐ Send the Full Path Name to the PLC

Note that the "Transfer Program to PLC" window also allows you to select:

a) whether only the program file name should be sent with the program or
b) if the entire directory path of where the program is saved should be sent

For more information on this, refer to section 8.3 Controller Menu

6. **Click 'Start Transfer'** to initiate the actual program transfer if  Success ! is displayed, which indicates program compilation was successful.



Success !

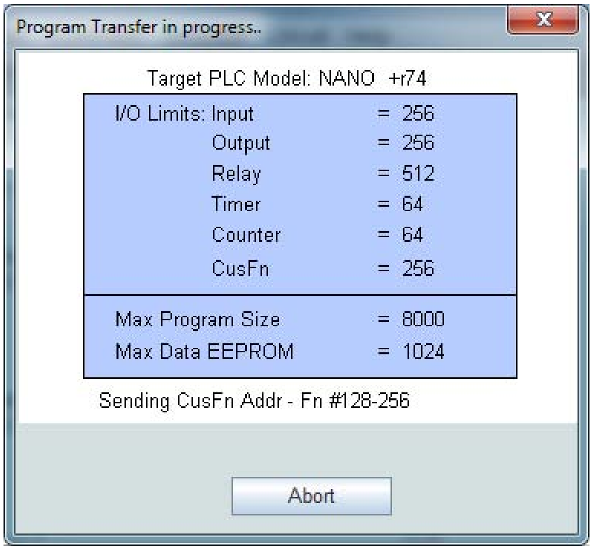
Total Number of circuits	7
Ladder Diagram (words)	28
Custom Function (words)	17
Total code Size	45

Checksum =00001036

DIFU+DIFD+ One-shot contact = 0 (max=256)

Program compilation occurs automatically and you would not be able to proceed with the transfer if it failed. In that case, the general location of the error will be displayed with a description to help with troubleshooting.

7. **Wait** while the program transfer takes place (below-left screenshot).

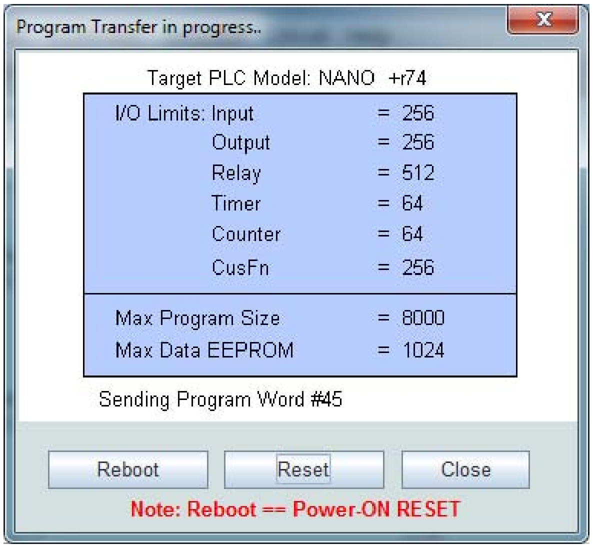


Target PLC Model: NANO +r74

I/O Limits: Input = 256
Output = 256
Relay = 512
Timer = 64
Counter = 64
CusFn = 256

Max Program Size = 8000
Max Data EEPROM = 1024

Sending CusFn Addr - Fn #128-256



Target PLC Model: NANO +r74

I/O Limits: Input = 256
Output = 256
Relay = 512
Timer = 64
Counter = 64
CusFn = 256

Max Program Size = 8000
Max Data EEPROM = 1024

Sending Program Word #45

Note: Reboot == Power-ON RESET

8. **Click 'Reset'** when the program transfer is complete (above-right screenshot).

IMPORTANT: (For Nano-10, FMD88-10, and FMD1616-10 PLCs) If your program is using any timers or counters, you must click '**Reset**' when the program transfer is complete so that the set values are properly stored to FLASH memory in the PLC.

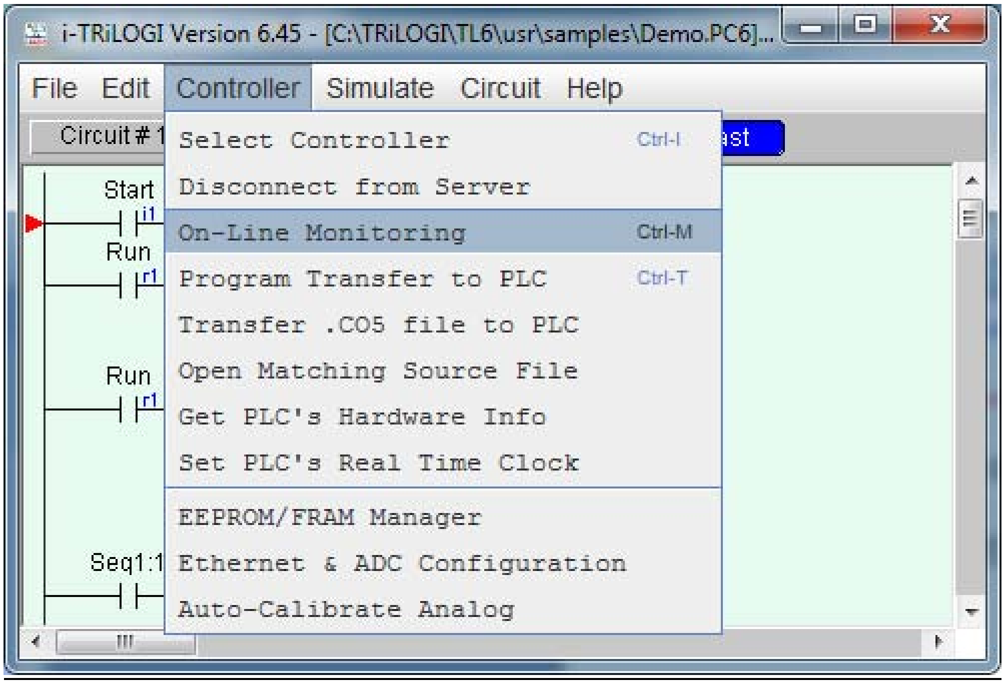
If you click on 'Close', timer and counter set values would not be retained and would be lost when PLC power is turned off or cycled unless the program executes a 'RESET' or 'SETSYSTEM 252,0' command first. This is necessary whether or not any FRAM-RTC module is installed. The program will still run normally other than that.

If you click 'Reboot', the PLC will be hard reset (CPU is power cycled). This is not used for normal program transfer and is only necessary if there is a specific purpose for cycling PLC CPU power. Just as for a Reset, any timer and counter set values would be permanently stored in the PLC.

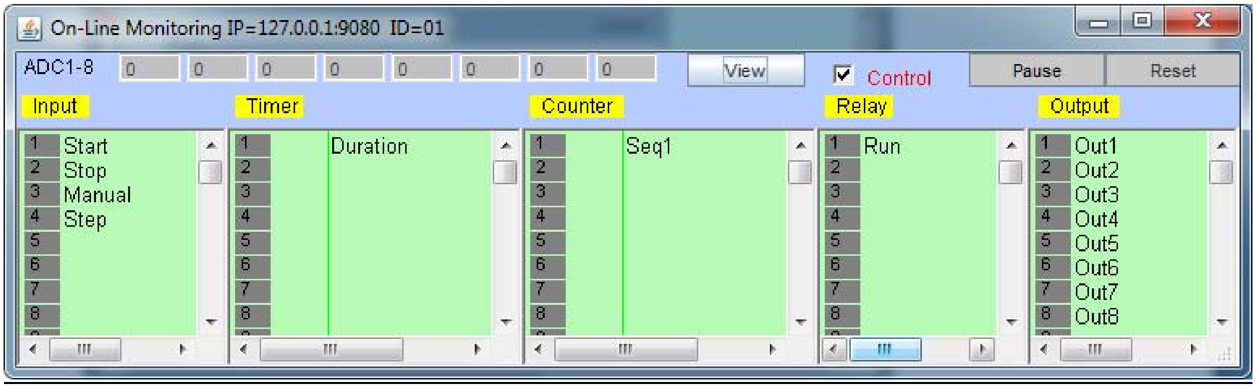
The program transfer is now complete and online monitoring can be performed.

Online Monitoring

a) **Select "On-Line Monitoring"** from the "Controller" menu (or press CTRL + M on the keyboard).


i-TRiLOGI Version 6.45 - [C:\TRiLOGI\TL6\usr\samples\Demo.PC6]...
File
Edit
Controller
Simulate
Circuit
Help
Circuit # 1
Select Controller Ctrl-I
Disconnect from Server
On-Line Monitoring Ctrl-M
Program Transfer to PLC Ctrl-T
Transfer .CO5 file to PLC
Open Matching Source File
Get PLC's Hardware Info
Set PLC's Real Time Clock
EEPROM/FRAM Manager
Ethernet & ADC Configuration
Auto-Calibrate Analog
Start
i1
Run
r1
Run
r1
Seq1:1

b) The online monitoring window should open immediately if you are still connected to the server.



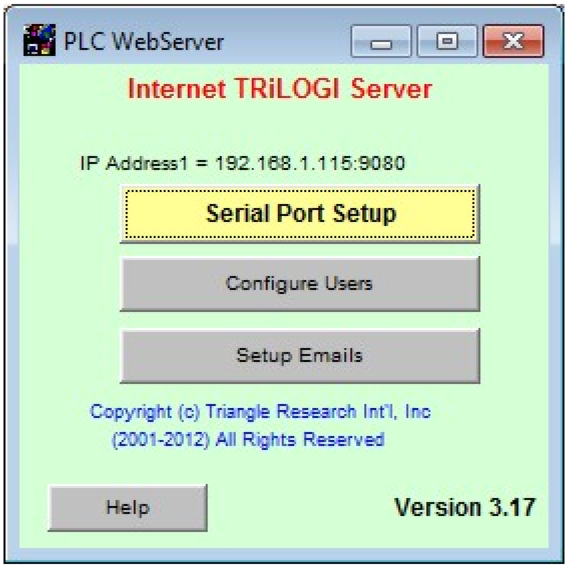
If not, then **follow steps 2-4** from the Program Transfer section above to login to TLServer or FServer. The above monitoring window will appear after successful login.

Chapter 4 USING THE TLSERVER – WEB SERVER FOR TRILOGI

4 USING THE TLSERVER – WEB SERVER FOR TRILOGI

4.1 Overview

When you first start TLServer, the following window will appear. Refer to chapter 1.5 Run TRiLOGI and TLServer for instructions on starting TLServer.


PLC WebServer
Internet TRiLOGI Server
IP Address1 = 192.168.1.115:9080
Serial Port Setup
Configure Users
Setup Emails
Copyright (c) Triangle Research Int'l, Inc
(2001-2012) All Rights Reserved
Help
Version 3.17

Traditionally, the TLServer program is used as the software gateway to connect the (now legacy) M-series PLCs to the corporate LAN or the Internet so that they can be controlled and programmed by an i-TRiLOGI client from anywhere in the world. M-series PLCs did not have built-in TCP/IP network capability and therefore had to rely on the PC and TLServer software to provide the network connectivity. TLServer provides the TCP/IP interface to the network client and it then pass the data to/from the PLCs via their RS232 or RS485 serial ports only.

On the newer Super PLCs such as the Nano-10, FMD or F-series PLCs, there is already a built-in Ethernet port which allows them to be connected to a router and hence directly accessible to/from the Internet. TLServer is therefore not needed for normal Ethernet communication with the PLC.

However, TLServer is still an important tool for i-TRiLOGI software to connect to the PLCs under the following circumstances:

1) A router connection is not readily available to the PLC.
2) The IP address and/or port number of the PLC's Ethernet port is unknown
3) The PLC' Ethernet setup parameter table is corrupted, rendering the Ethernet port inaccessible.

With TLServer, as long as there is an available serial port (RS232/RS485 or USB-RS232/RS485 adapter) on the PC, the i-TRiLOGI software will be able to connect to the PLC without using the PLC's Ethernet port at all. This allow i-TRiLOGI software to reset the PLC's Ethernet configuration in case of corruption or lost configuration data.

When a client program such as i-TRiLOGI wants to read from or write to a PLC, it sends a command to the TLServer using the TCP/IP protocol transported via the Intranet, the Internet or a local host connection. The TLServer, upon receiving the command, will carry out the actual reading or writing to the PLC via the PC's RS232 or RS485 port. The data received from the PLC is then relayed back to the client program via TCP/IP protocol.

TLServer is also a Web Server which serves up web pages that contain the i-TRiLOGI Java Applet to enable you to use any Java-enabled Web browser to access the PLC without the need to install a local copy of the i-TRiLOGI application software.

TLServer 3.17 now supports large font size increment so that it can display larger graphic on high resolution monitor on small screen.

Note: TLServer also provides "File and Email Services" to the PLC. That means that a PLC can send a command to the TLServer to open a file and save its data into the PC's harddisk. TLServer 3.17 now also features an "Email Relay Server" to help PLC send out emails via authenticated SMTP servers.

The new Email Service works differently from the original email function provided in TLServer 1.0 in that TLServer does not poll the PLC, instead it is the PLC that initiates an email request asynchronously. This makes it possible for a PLC to dial-in via a modem to request the TLServer to help it send out an email without demanding a constant connection the TLServer. However, the original email function is still supported in Version 3.x because that has the advantage of being able to service email requests for multiple PLCs linked via the RS485 network.

When TLServer is first started, it will query the operating systems for the IP addresses of the computer that it runs on. (It may take a while if the O/S is slow to return the IP address). It will then display the obtained IP addresses (maximum of two) on the TLServer front panel so that the user can quickly determine the IP addresses that they can use to access the PLC. The following are some possible IP address scenarios:

1. If the computer is not linked to any network or the Internet and does not have any network adapter installed, then only the local host IP address will be displayed, e.g. 127.0.0.1: 9080 where 9080 is the port number). Note: regardless of whether your PC is networked or not, the local host IP address: 127.0.0.1 is always available to the client program running on the same PC where the TLServer is running, even though it may not be displayed on the TLServer's front panel. (i-TRiLOGI or TRi-ExcelLink are all

known as "client" programs). So whether your PC is networked or not, you can still use TLServer and i-TRiLOGI on a localhost connection. In that case the i-TRiLOGI and TLServer work together on the same PC just like a normal Window based programming software. We recommend using the localhost IP address: 127.0.0.1:9080 if you are running both the client and the server on the same PC.

2. If the computer has an 'always on' connection to the Internet directly then the IP address will be your Internet IP address.

3. If the computer is networked to the corporate Intranet, or you have connected this computer to a router to share internet connection with a few other computers, then the IP address shown is an internal IP address, also known as the "Intranet" IP address. The intranet IP address is assigned by either the System Adminstrator or the router (known as DHCP server). You can access this computer from other computers in the same LAN, but the intranet IP address is not accessible from outside of the LAN. To access the TLServer from outside of the LAN, You will need to configure your router's internal settings to define the PC that runs the TLServer as a "Virtual Server". You can then access the TLServer using the router's public IP address and the router will does the job of translating the public IP address to the intranet IP address and route the messages to/from the PC that has been defined as the virtual server. This process is known as Network Address Translation (NAT).

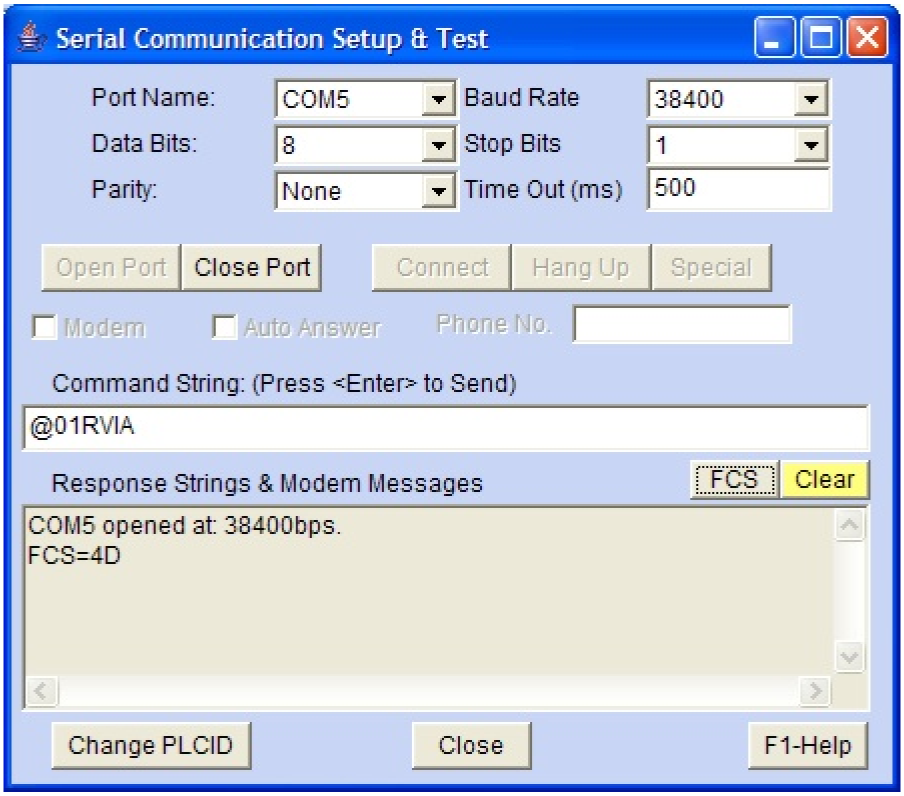
4. Dial-Up Users: If you are testing the internet capability of TLServer using dial-up connection, you must connect to the Internet first before starting TLServer so that TLServer can report the correct Internet IP address to you. You will not see the local host IP address (127.0.0.1), only the Internet IP address will be shown.

The moment TLServer is running, it is ready to accept connection from the i-TRiLOGI client. You can also configure TLServer's communication port setting, add/remove users from the system and set up TLServer to query the PLC for outgoing email requests and process them accordingly. For explanation of the function of each button, click the image link of respective buttons below. You can also call up their context-sensitive help by pressing <F1> key after pressing the relevant button on the TLServer front panel.

NOTE: TLServer version 3.17 has a new and improved Email Setup area. To view the changes, click on the "Setup Emails" button below as described above.

4.2 Serial Port Setup Serial Port Setup

4.2.1 Setting Up and Testing TLServer's Serial Communication Port



Port Name: COM5
Baud Rate 38400
Data Bits: 8
Stop Bits 1
Parity: None
Time Out (ms) 500

Command String: (Press <Enter> to Send)
@01RVIA

Response Strings & Modem Messages

```
COM5 opened at: 38400bps.
FCS=4D
```

This dialog box allows you to configure the serial port of the host computer to match the setting on the PLC for proper communication. Most of the items here are self-explanatory. If you have more than one PLC connected to the host computer via RS485, all the PLCs must have the same serial port settings as the TLServer. The Open Port button allows you to test whether the communication port is available to TLServer. You can also click the Close Port button to temporarily relinquish the port to other applications. Note that you will need to close an opened port before you can change its parameter.

The (Command String) text entry field allows you to test communication with the PLC using its native or MODBUS ASCII protocols. If you enter a string here and press <Enter>, the ASCII string will be sent to the PLC connected to the serial port and the response string will be displayed in the bottom text box. If the comm port is not yet opened this command will automatically open it.

Note that only multi-point host link commands are accepted here. The only point-to-point command acceptable here is the "IR*" command which queries the ID address of the PLC.

If you have only one PLC connected to your TLServer computer, then you can test the communication now using the following command string:

Command String : IR*
Response String: IR01*

The response string tells you that the ID address of this single PLC is 01. You can then try other host link commands using this ID address. (e.g. @01RI0000* to query the states of inputs #1 to #8) If you have more than one PLC connected you should not use the "IR*" since all connected PLCs will try to respond simultaneously, thus resulting in a garbage return string.

To change the ID of a PLC, e.g., from 01 to 05, you can send the command string "@01IW0500*" to the PLC. There is also a Change PLCID button which does this for you automatically. You can click on the "Detect ID" button to check the current ID and then the "Change ID" button to write the new ID to the PLC. If the response string box become too cluttered, click the Clear button to clear the response box content.

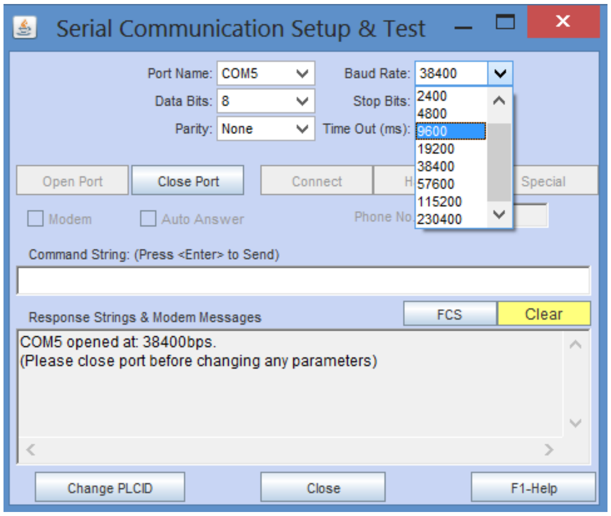
The FCS button can be used for computation of the "FCS" (Frame Check Sequence) characters from whatever text string you enter in the Command String box. In the above example, since there is currently a text string "@01RVIA" in the Command String box, clicking on the FCS button will compute the FCS of the string "@01RVIA". You can then append the computed FCS to the command string to form a complete command string with FCS and send to the PLC: "@01RVIA4D*"

4.2.2 Changing Communication Settings

Most likely you may want to leave the comm port settings at their default values: **38,400 bps, 8 data bits, 1 stop bit, no parity**. One reason for changing the comm port settings may be due to the need to change the PLC's serial port to lower values (e.g. for communication via radio using 9600 bps). Changes to the comm settings are saved to the TLServer configuration file: "TLserver1.cfg" when you quit TLServer.

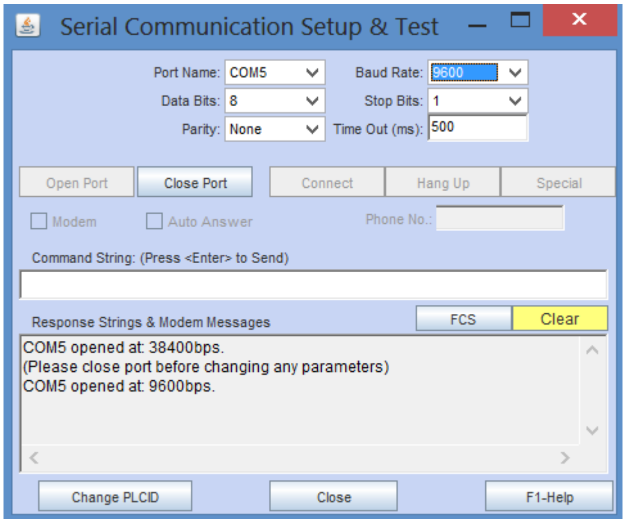
One other scenario is when you need to power cycle a M-series PLC with DIP switch #4 turned ON (to halt the CPU to disable the "1st.Scan" pulse). The Nano-

10, FMD and F-series PLCs will be forced to the default baud rate of 38400 bps with 8 data bits, 1 stop bit, and no parity when started in pause mode (for all 3 serial ports). These newer PLCs will NOT be forced to 9600 bps unlike that of the RS232 port on the M-series PLCs.Since the PLC's serial port is set to 9600 bps when power ON with DIP Switch #4 set, you will need to change the baud rate temporarily in order to communicate with the PLC (e.g to disable a program that causes trouble).


Serial Communication Setup & Test
Open Port
Close Port
Connect
Special
Modem
Auto Answer
Phone No
Command String: (Press <Enter> to Send)
Response Strings & Modem Messages
FCS
Clear
Change PLCID
Close
F1-Help

Port Name: COM5
Baud Rate: 38400
Data Bits: 8
Parity: None
2400
4800
9600
19200
38400
57600
115200
230400

COM5 opened at: 38400bps.
(Please close port before changing any parameters)


Serial Communication Setup & Test
Open Port
Close Port
Connect
Hang Up
Special
Modem
Auto Answer
Phone No.:
Command String: (Press <Enter> to Send)
Response Strings & Modem Messages
FCS
Clear
Change PLCID
Close
F1-Help

Port Name: COM5
Baud Rate: 9600
Data Bits: 8
Stop Bits: 1
Parity: None
Time Out (ms): 500

COM5 opened at: 38400bps.
(Please close port before changing any parameters)
COM5 opened at: 9600bps.

However, do remember to change the baud rate setting back to 38400 bps after you have reset the PLC with the DIP switch OFF, otherwise you may have problems communicating with the PLC later on since changes to comm settings are automatically saved.

NOTE: The Nano-10, FMD and F-series PLCs will be forced to the default baud rate of 38400 bps with 8 data bits, 1 stop bit, and no parity when started in pause mode (for all 3 serial ports). These newer PLCs will NOT be forced to 9600 bps unlike that of the RS232 port on the M-series PLCs.

4.2.3 Modem Support

1. Dial Modem: TLServer 3.x incorporates support for dialing a modem connected to the PC's com port. This is useful if the PLC has to be located at a remote location yet still has access to the public telephone line or to a cellular phone. You can then connect the PLC to a standard analog modem such as the US Robotic 33.6Kbps or Hayes Acura smart modem. The TLServer can then dial the phone number of the remote modem and make a connection. Once a connection is established, the remote PLC is immediately accessible to client applications such as i-TRiLOGI or TRi-ExcelLink, etc over the Internet, Intranet or localhost as if it were connected to the TLServer via the serial port directly

Notes:

- Due to the time delay for modulation/demodulation process as well as transmission delay, two-way communications via modem tends to be noticeably slower than connection made by direct cable connection. This is quite normal and does not indicate a problem with the communication setup.

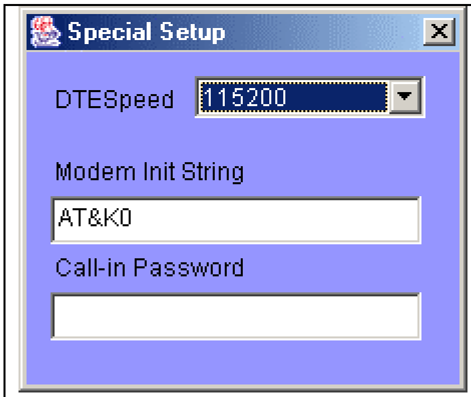
- The PC's modem must be able to emulate the COM port of the PC in order for the TLServer modem function to work. Some of the newer computers employ "win modem" or "soft modem" which may only work with Windows' dial-up networking. These kinds of modems are implemented in software and they do not necessarily emulate a standard PC COM port properly. They also demand quite a bit of CPU power to process the communication properly. Therefore these type of modems may not work too well with the TLServer. If your built-in soft modem does not work properly with TLServer, you should get an external PCMCIA card modem and these are quite inexpensive nowadays and they will work much better with the TLServer modem support function.

To setup TLServer to dial a modem, first close the active COM port by clicking on the "Close Port" button. Select the COM port where the modem is connected to. (you can find out the which COM the modem is connected to by checking the "Control Panel -> Modems -> Properties") Click to select the "Modem" checkbox. You will then be able to enter a telephone number to dial. The 3 buttons: "Connect", "Hang Up" and "Special" become enabled when you select the "Modem" mode. Note that the "Baud Rate" field now becomes the "DTE speed" which specify the line rate between the PC and the modem (this has nothing to do with the actual baud rate between the modems which will be automatically negotiated based on the quality of connection). Normally you should leave the DTE speed set to the highest value (115200) unless your modem manufacturer specifies otherwise. The PLC can be operating at a different baud rate from the PC to modem-line-rate because of the modulation/dimoduation action of the modem.

*** Important:** The PLC-to-modem connection must be properly prepared before you can use TLServer to connect to the PLC's modem. You can find more details about this in the "PLC-to-Modem Communication Setup" appendix.

Once you have entered a proper phone number, click on the "Connect" button to start dialing the modem (make sure that the "Auto Answer" check box is not checked). If the remote modem is busy or does not answer the call you will see the corresponding error messages in the response box. Click on the "Hang Up" button anytime to abort the dialing operation.

If you click on the "Special" button a special dialog box will appear as follow:



DTESpeed 115200

Modem Init String AT&K0

Call-in Password

- You can change the DTE speed by selecting a new value from the choice menu.
- You can specify a special AT command to be sent to the modem during modem initialization. Normally you can leave this field to its default value which is AT&K0.
- You can also specify a special "Call-in Password" which is only used if the TLServer puts itself in auto-answer mode (see description later). Any incoming connection made by a remote modem must give the correct password upon connection, otherwise the connection will be immediately dropped. The Call-In password feature is disabled if the corresponding textbox is empty.

2. Auto Answer: If you select the "Auto Answer" checkbox and click on the "Connect" button, the TLServer will setup the modem to automatically answer the incoming call on the first ring. There are many uses of this capability:

- Any number of PLCs in the field can periodically dial in to a single TLServer and write or append the values of their internal variables to data files on the PC's hard disk using the PLC File Service commands. This is extremely useful for deploying the PLC for data-acquisition purposes. You can format the data using CSV format so that the file can be readily imported into an MS Excel or Lotus 123 spreadsheet.
- The remote PLC can dial in and ask TLServer to send out its data to any email address immediately.
- The remote PLC can dial in and synchronize its real time clock with the TLServer.
- The TLServer can play the role of an ISP where the PLC can dial in and be accessible to the Internet.

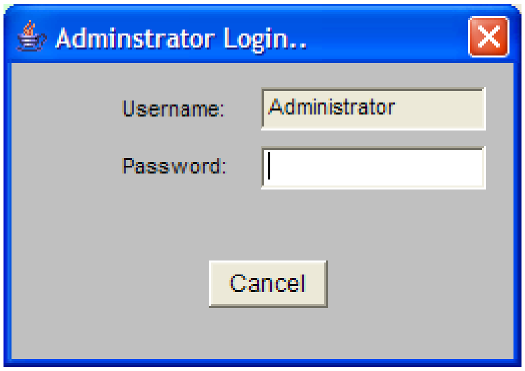
Some sample ".PC6" programs that enable a PLC to dial in to the TLServer and request for file or email services are provided in the following folder:

"C:\TRiLOGI\TL6\usr\samples\FileService_Modem"

To prevent unauthorized incoming call, you can specify a "Call-In Password" string as described above. If the "Call-In Password" contains any text other than an empty string, then the incoming caller, upon connection, must immediately send a CR-terminated string that matches the "Call-In Password" string in order to maintain the connection. If the password is incorrect the TLServer will disconnect the remote modem to prevent unauthorized access. If the call-in password is validated the TLServer will acknowledge this by sending a CR-terminated string "<OK>" to the remote PLC via the modem. It is the duty of the incoming caller to check the acknowledgement string to ensure that the connection is not dropped by the TLServer.

4.3 Configure Users

By definition, only the “Administrator” is authorized to add/delete users and change password. Hence when you click on the “Configure Users” button, you are assumed to be the Administrator and you will be asked to enter the Administrator’s password to gain entry. <u>By default, no password has been defined for the Administrator</u>, so you should just press <Enter> key to gain entry the first time.



Username: Administrator

Password:

Once you get through the Administrator Login screen, a dialog box will popup, which allows you to add new users who are allowed access to the TLServer and the PLCs. You can also change the password, username or the access level of an existing user or delete an existing user. A new user defined here will be given his/her own exclusive subdirectory to store ladder programs. For PCs, this directory is located at:

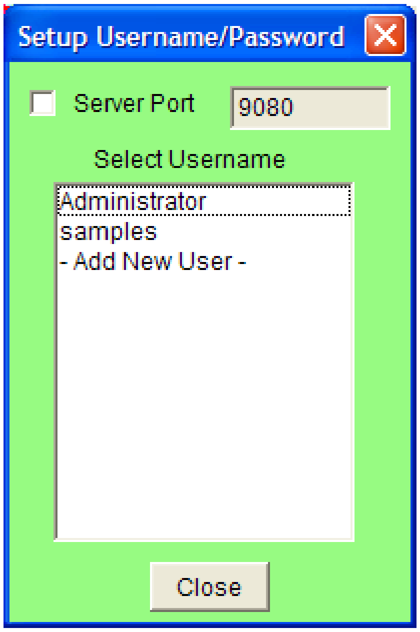
"C\TRiLOGI\TL6\usr\<username>"

where <username> is the same as the Username defined here.

4.3.1 Select Username

Double-clicking on an existing username opens up the username/ password dialog. You can now add password to the Administrator if you wish to prevent unauthorized access to the predefined usernames and passwords. There is also a pre-defined user named “samples” with no password where many samples TRiLOGI files are stored.

If you select a username and then press the <DEL> key, you can delete the user provided its directory is empty. (You can use Window Explorer or TRiLOGI Application to delete the contents of the user's directory first before deleting him/her from TLServer).



4.3.2 Add New User

Clicking on this field allows you to add new users to the system. You can add as many users as you like subject to memory and hard disk limit.

4.3.3 Server Port

If you click the check box to the left of the "Server Port" label, you can change the default "port" that the TLServer listens on. When the client accesses the TLServer. Whatever you define here must be matched by the same port number

E.g. if the port number = 8000, then localhost access must be:

http://127.0.0.1:8000/

However, if the port number is defined as 80 (default port for HTTP server), then you can access the server using just the IP address without the port number: http://127.0.0.1/

4.3.4 What Port Number Should TLServer Use?

In most cases you can simply use the predefined port number “9080”. However, you may like to read the explanation box below regarding definition of a "Port". You can see that the default port for most public web-servers is port 80. You can define TLServer to listen at port 80 as well; in that case there is no need to specify the port number in the URL. However, there are reasons why you may or may not want to do that. It depends on whether you are installing TLServer on a corporate intranet or on the public Internet and whether the client (TRiLOGI) is to access TLServer within the intranet environment or from the public Internet. Please read Appendix 2: “PLC & PC Hardware Setup and Configuration” for an explanation of how to use the port number properly.

Ports, or addresses within a computer, are used to enable communication between programs. Port addresses are 16-bit addresses that are usually associated with a particular application protocol. An application server, such as a Web server or an FTP server, listens on a particular port for service requests, performs whatever service is requested of it, and returns information to the port used by the application program requesting the service.

Popular Internet application protocols are associated with well-known ports. The server programs that implement these protocols listen on these ports for service requests. The well-known ports for some common Internet application protocols are shown below:

Port	Protocol
21	File transfer protocol
23	Telnet protocol
25	Simple mail transfer protocol
80	Hypertext transfer protocol

4.4 Setup Emails

Setup Emails

4.4.1 Introduction

Before the Ethernet port was integrated into the newer Super PLCs such as the Nano-10, FMD and F-series, TRi PLCs can only use the TLServer as TCP/IP gateway to perform network related functions, including sending of emails. For newer PLCs with built-in Ethernet, in many cases the PLC can be setup to send email directly without relying on the TLServer. Please see the following FAQ posting for a description of all the current email methods:

http://www.tri-plc.com/faqans5.htm#ans61

However, TLServer can still be very useful as email helper in the following situations:

1. There is no Ethernet cable available to the PLC but the PLC are networked via RS485 to a PC.
2. The PLCs are not permitted to make TCP connection directly to SMTP server residing outside of the LAN.
3. The SMTP server requires authentication to send email that the PLC cannot handle natively.

The following describe in details the several ways in which the TLServer can be used to help the PLC send out emails.

Email Method 1 - Using TLServer as Email Master

The TLServer supports a legacy email feature (carried over since version 1.0) where the TLServer act as a master to a bunch of PLCs connected together via their RS485 port. TLServer can be configured to scan the internal variables of any of the up to 256 PLCs ID connected to it to determine if any of these PLCs has a request for the TLServer to send out email. If so the TLServer will then retrieve the email data from the relevant PLC, constructs a complete email and send it out. Click here for more details.

Email Method 2 - Using TLServer as Email Slave

In this setup, the TLServer will monitor the data it receives via its serial port for a special <EMAIL> tag, which triggers a sequence of actions where it receive the sender, recipient, subject and mail content via the serial port. When all the data have been received the TLServer will connect to the SMTP server and send out the email. In this setup only 1 PLC may be connected to the TLServer to send out emails since it is a one-to-one interaction between the PLC and the TLServer program. For more details on this email method, please view the "File and Email Services" section.

Email Method 3 - Using TLServer as Email Relay Server - NEW!

As you will see shortly, since version 3.16, the TLServer can be setup to work with both SMTP servers that require authentication and those that don't. However, the default email function on the Nano-10, FMD and F-series PLCs does not support account authentication to communicate with the external SMTP server. If the PLC is connecting to the SMTP server provided by the ISP then in many cases authentication is not needed since the ISP will only allows connection from IP address that belongs to its subscribers.

However, if your SMTP Server is not supplied by the ISP of the network to which the PLC connects, then most likely authentication will be required. In such a case the simple <EMAIL> tag method described on the PLC's manual will not work since the PLC does not handle authentication with the SMTP server.

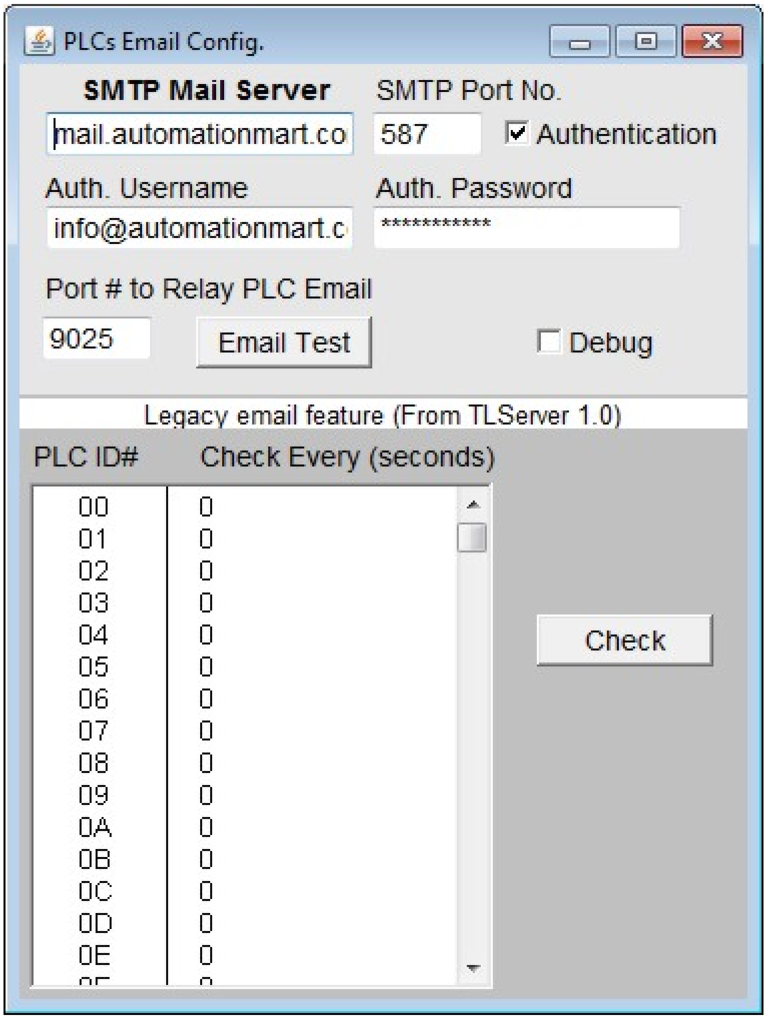
It is possible to program the PLC to handle unencrypted authentication with the SMTP server using complex TBASIC statements in order to send email (see examples described on the forum under "Frequently Asked Questions"). However, to simplify programming effort, we have introduced a new, "Email Relay Server" feature to TLServer starting from version 3.17. The "Email Relay Server" feature is for the TLServer to act as a pseudo, unauthenticated SMTP mail server to the PLCs and receive the email data from the PLC. It then in turn connects to an authenticated SMTP server and "relay" the email data to the actual SMTP server. In order to use this feature, the PLC would configure its SMTP IP address to be that of the IP address of the PC that is running the TLServer, and the SMTP port to that of the "Port # to Relay PLC Email".

The Email Relay Server can be any PC on your LAN that runs the TLServer that is already configured to work with an external SMTP mail server (whether autheticated or not). Your PLCs on the same LAN can then send their email via this TLServer instead of directly to the external SMTP server. This could be advantages in terms of ease of programming and also should the company decide to change its ISP or SMTP server provider, only the TLServer needs to be re-configured and the PLCs on the network need not be configured individually to address the change.

4.4.2 Configure And Test SMTP EMail Server

NEW!: TLServer version 3.17 offers a whole new look to the Email Setup, which has all the same features as before plus a couple of additions, including Port No. specification, optional Authentication and a Test Email function to test the settings you have entered. See below for descriptions of the additions.

To setup the server to handle email requests, click on the "Setup Emails" button on the TLServer to open the following dialog box:



SMTP Mail Server: mail.automationmart.co

SMTP Port No.: 587

☑ Authentication

Auth. Username: info@automationmart.c

Auth. Password: ***********

Port # to Relay PLC Email: 9025

☐ Debug

Legacy email feature (From TLServer 1.0)

PLC ID#	Check Every (seconds)
00	0
01	0
02	0
03	0
04	0
05	0
06	0
07	0
08	0
09	0
0A	0
0B	0
0C	0
0D	0
0E	0

SMTP Mail Server: This will be the same Outgoing Mail Server that you use in your email program such as the Eudora or Outlook Express. If in doubt, ask your ISP or System Administrator for help. This server must be setup properly before the TLServer can send any email.

SMTP Port No.: This is the port that your Outgoing Mail Server will use to send emails. The default port number is 25, which is the most common port; however, some Mail Servers will use a different port and you should check with your ISP or System Administrator if you are not sure.

Authentication: Some ISPs will only allow mail to be sent through their Mail Server if it originates from a trusted IP address (typically, only the Internet IP address that the ISP assigns to you). If this is the case, then you may need to authenticate your IP address if it is different from the one provided to you (If you are sending mail from a location registered to another ISP using your Mail Server). If your ISP uses Authentication, then you will need to check the Authentication box and provide the correct Auth. Username and Auth. Password. If you are unsure about this, then you should contact your ISP or System Administrator for help.

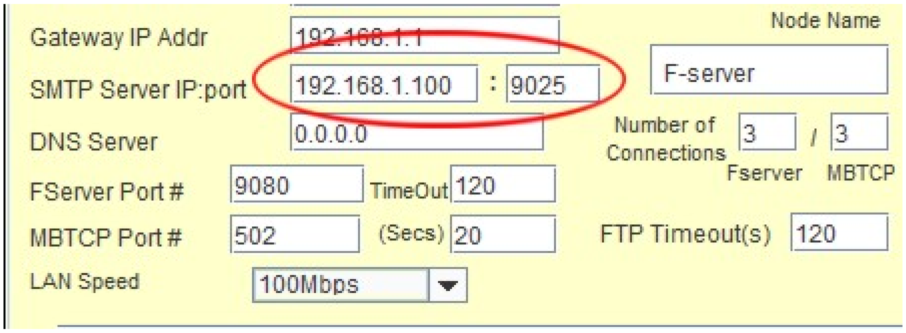
Auth. Username: If Authentication is required, you will need to provide the correct Username. This is provided by your ISP and you should contact your ISP or System Administrator if you don't know what it is or if you don't know if you need it.

Auth. Password: If Authentication is required, you will need to provide the correct Password. This is provided by your ISP and you should contact your ISP or System Administrator if you don't know what it is or if you don't know if you need it.

Port # to Relay PLC Email: If the TLServer is setup to be the email relay server for the PLCs, TLServer will be listening on this port for a SMTP connection from the PLC. The default port number is 9025. This port number is to be used as the SMTP port # on the PLC's "Ethernet & ADC Configuration" screen in order for the TLServer to relay the email content to the actual SMTP server that requires authentication. Please see Email Method 3 -Using TLServer as Email Relay Server for description onon this topic)

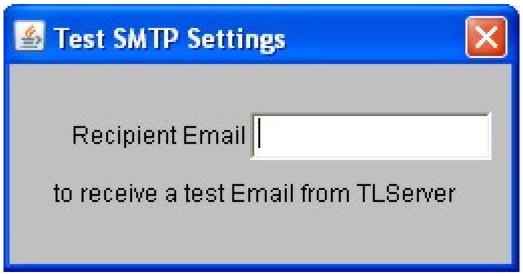
Important: your PC's firewall must be setup to allow connection to this port from other devices on your network in order for TLServer to work as an email relay server for the PLC.

E.g. If the PC running the TLServer has IP address = 192.168.1.100, and the "Port # to Relay PLC Email" is set to 9025, then if the PLC wants to use TLServer as a SMTP relay server, it should configure its Ethernet port settings as shown below:



Gateway IP Addr: 192.168.1.1
SMTP Server IP:port: 192.168.1.100 : 9025
DNS Server: 0.0.0.0
FServer Port #: 9080
MBTCP Port #: 502
TimeOut (Secs): 120 / 20
LAN Speed: 100Mbps
Node Name: F-server
Number of Connections: Fserver 3 / MBTCP 3
FTP Timeout(s): 120

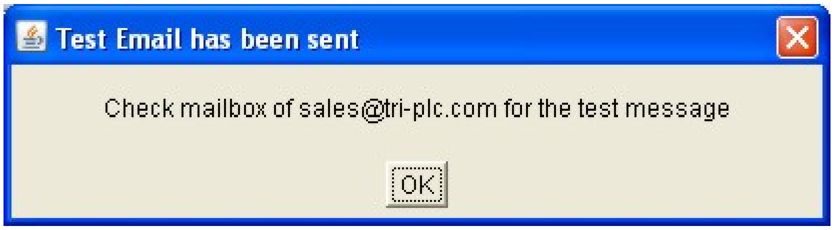
You can test your email configuration by clicking the button once you have entered all the correct settings. Once you click the "Email Test" button TLServer will initiate the sending of a test email using the Mail Server you specified. A new window will pop up, as shown below, that allows you to enter the recipient email address.



Recipient Email

to receive a test Email from TLServer

Once you have typed the recipients email address, you can press enter to send a default email (can't be changed). If it works, you will see a message window that says the email has been sent, as shown below, and you will receive an email with the following content: "Test Message sent from TLServer"



Check mailbox of sales@tri-plc.com for the test message

If some settings in the email configuration are not correct or you do not have a working Internet connection, you will see an error message.

☑ Debug

If you use the TLServer email relay feature and encounter failure when the PLC attempts to email and if you cannot figure out what caused the problems, then you can start upthe TLServer with Java Console enabled and re-run the email test with the "Debug" checkbox selected. The Java console will output all interactive messages the TLServer exchanges with the SMTP server. From the error messages produced you may be able to better figure out the cause of the email failure.

Note: TLServer automatically enables debug messages when you use the "Email Test" button to send a test email via the SMTP server, regardless of the state of the "Debug" checkbox. The "Debug" checkbox is effective only when TLServer carries out the the email relay operations on behalf of the PLCs.

4.4.3 Setting Up TLServer As Email Master For Multiple PLCs on RS485

A PLC program raises an email request flag by setting the variable emEVENT[1] to a non negative value (see explanation on "Writing i-TRiLOGI Programs that Can Send Emails") whenever it needs to send an email. The TLServer, upon sensing that an email request flag has been raised, will extract the sender, recipient and message strings from the PLC's internal variables and send them out using the pre-defined SMTP outgoing mail server.

A single TLServer can service the email requests for one or more (max. = 256) PLCs connected to it via RS232 (single PLC only) or RS485.

The second half of the Email Setup screen with the title "Legacy EMail Feature (from TLServer 1.0)" are related to this email method. The setup is as follow:

PLC ID# column:

For you to select PLCs with ID from 00-FF (256 in total) to set the email service period.

Check Every (seconds)

This allows you to define how often the TLServer should check the PLC (the email service period) for the state of the outgoing email request flag.

Simply click on field next to the PLC ID# of interest to open up a text entry field (as shown in the figure for ID=02). Enter a non-zero value (in seconds) to define its email servicing period.

Note: Both SMTP Server and email service period definitions will be saved to the hard disk when you exit TLServer program. They will be reloaded when you start TLServer again.

Inactive PLC

TLServer attempts to communicate with every PLC that has a non-zero email service period. However, if the PLC is inactive (e.g. It has not been turned ON or has been disconnected from the server) the communication will fail. Since communication failure takes considerable amount of CPU waiting time and could affect the normal communication with other active PLCs, inactive PLCs are internally marked by the TLServer (shown as Comm Error!! message in the email setup dialog) and will not be checked according to their defined service period to avoid repeated communication failure. However, TLServer will rescan these inactive PLCs every two minutes to check whether they have come on-line. If an inactive PLC is found to respond to command it will be unmarked and put back in service for its email request.

Test Email

You can also manually force the re-scanning of all PLCs by clicking on the Check button once. Then scroll to the PLC of interest to check if there is a Comm Error!! message. Check the PLC communication port wiring if there is an error.

This email service period does not determine how often the PLC will send email, since email will only be sent when the email request flag is set even if you had set a very short email service period. It only affects how quickly the email will be sent whenever a PLC raises its email flag. You should set a short service period (say every 10 seconds) for urgent email (such as alarm condition). For non urgent email such as hourly or daily production report you can set a much longer servicing period to reduce the communication loads on the PLCs.

4.4.4 Writing i-TRiLOGI Programs that Can Send Emails via TLServer as Email Master

In order to send an email, the i-TRiLOGI program needs to use the string variables A$, B$, C$ to store the headers and D$ to Z$ to store the messages. (not all strings need to be used, unused strings are still available for normal program use) The special variable emEVENT[1] is used as an email request flag which should be initialized to -1 when the program is not requesting email service. When the i-TRiLOGI program wants to send an email, it first stores the sender, recipient and subject into the following variables:

A$	Sender email address - which can be used to identify the source of the email.
B$	Recipient email address - this one must be accurate
C$	Subject of the message.
D$	First line of Message
E$	Second line of email message
...	.....
Z$	The 23rd line of the email message
emEvent[1]	-1 = NOT sending any email. 0 to 23 = number of lines in the email message body which are contained in D$ to Z$.

The maximum number of lines in your email body is limited by the number of string variables D$ to Z$ (23 in total) available in the Nano-10, FMD series, and F-Series PLCs.

For example, if the PLC needs to send email to trilogi@yahoo.com with a 1-line greeting, then the program needs to activate a custom function that contains the following statements:

A$ =	"Demo1@PLC"	' sender
B$ =	"trilogi@yahoo.com"	' recipient
C$ =	"This is an email demonstration"	' subject
D$ =	"The time is"+STR$(TIME[1])+":"+STR$(TIME[2])+". How are you doing?"	' Message body
emEvent[1] =	1	

You must also setup the email service period (say every 10 second) in the "Setup Emails" screen for this PLC. When the TLServer scans the PLC and found that its emEvent[1] is set to 1, it will extract the headers and message body from the PLC's string variables. Only A$ to D$ will be extracted in this example since the message contains one line of body text only, as indicated in emEvent[1]).

TLServer will then contact the SMTP server to send out the email. In addition, after processing the email request, the TLServer will set the emEvent[1] variable to a value of "-1" (no email). Hence there is no need for the i-TRiLOGI program to worry about clearing the email request flag after the email has been sent. In addition, this provides a way for the PLC program to know whether the TLServer is functioning properly and whether the email request has already been processed. However, do take note that even if the emEvent[1] has been reset it does not guarantee that the email has been correctly dispatched to the recipient. Success of emailing is subject to the proper configuration of the TLServer, the network quality and availability of the SMTP server at the moment when TLServer tries to send out the email. For urgent situations you may consider sending out multiple emails periodically until the user has attended to the machine.

4.5 File and Email Services

Starting from version 2.0, the TLServer provides a number of “File and Email Services” to the PLCs via the serial comm port. TRi Ethernet PLCs such as Nano-

10, FMD series, and F-series PLCs can also perform these file and email services via TCP/IP from their built-in FServer, which is described in the Network Services section of the PLC user manual (chapter 2.3). This section will only describe how to do such tasks with TLServer, so the FServer would only be referenced when the TLServer is being used to make a connection to it.

Basically, a PLC can send service requests to the TLServer using "tags" (which are ASCII characters enclosed between the '<' and '>' characters) and the TLServer will perform the service requests upon receiving valid commands. All data between the <command [parameter]> tag and the </> tag will be treated as data for the requested service.

Since the PLC is the one that initiates the service request, it does not need to be linked to the TLServer all the time, only when it needs to request a service from the TLServer. This makes it possible for a remote PLC to connect to the TLServer, via the telephone line and modem to perform the required file or email services, then disconnects itself from the TLServer so that other PLCs can take turns to connect to the TLServer to request for services.

Note**:** All the files created or used in the write/append/read actions are located in the directory: <trilogi base directory> /FileService. (Hence the default path is C:\TRiLOGI\TL6\FileService). You may also read/write files that are located in the sub-directory below the "..../FileService" directory provided that the sub-directory already exists.

The currently supported files and emails services are described below:

1. Write data to file.

Format:

<WRITE [filename]>
data data data...
data
....
</>

E.g. To save data of DM[1] to DM[10] to a file name "testWrite.txt", execute the following statement from a custom function:

```
PRINT #1 "<WRITE testWrite.txt>"     ' Write data request
FOR I = 1 TO 10
    PRINT #1 DM[I];"    ";      REM delimited by space characters.
NEXT
PRINT #1     ' send a CR character.
PRINT #1 "</>"     ' End of Service request
```

The TLServer will close the file after it receives the end-of-service tag "</>" from the PLC and it will in turn send a "<OK>" string to the PLC to acknowledge that the **WRITE** request has been completed successfully. It is up to your PLC program to check for the "<OK>" tag to determine if

	the service it requested have been completed successfully.

2. Append data to file
Format:
`<APPEND [filename]>`
`data data data...`
`data....</>`
	E.g. To append the time of an event to a file name "testAppend.txt", execute the following statements in a custom function when the event take place:
	PRINT #1 "<APPEND testAppend.txt>" ' Append data request
	PRINT #1 "Event Time = ";TIME[1];":";TIME[2];":";TIME[3]
	PRINT #1 "A=";A
	PRINT #1 "</>" ' End of Service request
	• If the file does not exist a new file will be created. Otherwise, the PLC's real time clock data in the format "hh:mm:ss" and the value of A will be appended at the end of the file "testAppend.txt" every time the above statements are executed.
	• The TLServer will close the file after it receives the end-of-service tag "</>" from the PLC and it will in turn send a "<OK>" tag to the PLC to acknowledge that the **APPEND** request has been successfully completed. It is up to your PLC program to check for the "<OK>" tag to determine if it the service it requested have been completed successfully.

3. Email data to recipient **Format:** **<EMAIL [email address]>** **Sender: [sender email]**	E.g. To send data to an email address: whoever@yahoo.com with the subject "PLC Email Test", execute the following statements: PRINT #1 "<EMAIL whoever@yahoo.com>" ' change it to your own email. PRINT #1 "Sender: triuser@hotmail.com" ' it can be anything. PRINT #1 "Subject: PLC Email Test" PRINT #1 "Hello, this email is sent by your friendly TRiLOGI

Subject: [subject text] **data data data...** **data** **....** **</>**	PLC" PRINT #1 "Don't worry, everyting is working out great today!" PRINT #1 </> **Note:** • "Sender:" field should be in email format such as xxx@yyy.zzz, but it does not need be a valid email address. • "Subject:" field is optional and may be omitted totally • The TLServer will first save all the data it received into a temporary file named "Email.txt" in the default directory. After the TLServer receives the end-of-service tag "</>" from the PLC and it will then send out the email to the recipient email address. This email service will make use of the SMTP server defined in the "Setup Emails" portion of the TLServer configuration, so make sure that you have defined a correct SMTP server before testing the email service function. When the email has been successfully sent via the SMTP server, the TLServer will send an "<OK>" tag to the PLC to acknowledge that the **EMAIL** request has been successfully completed. It is up to your PLC program to check for the "<OK>" tag to determine if it the service it requested has been processed.

4. Read Data from File. **Format:** **<READ [filename]>** **</>**	This service allows the PLC to request the TLServer to open a text file and upload its content to the PLC. This may be useful for loading some previously saved parameters. Upon receiving this command and if the specified [filename] is successfully opened, the TLServer will begin sending all the ASCII characters contained in the text file to the PLC. Note that line breaks in a text file are sent to the PLC as CR character only and not as a CR+LF pair. As such, your PLC program can easily use the INPUT$(1) command to read in all the CR

	terminated text string one string at a time and then interpret or convert the data as necessary. After sending out the last byte in the data file to the PLC, the TLServer will send a CR-terminated acknowledgement string "<OK>" to the PLC to signal that the **READ** command has been properly completed.

5. Read Real Time Clock From TLServer **Format:** **<READ RTC[]>** **</>** **<READ Date[n]>** **</>** where n = 1,2,3,4 **<READ Time[m]>** **</>** where m=1,2,3.	This service allows the PLC to get the Real Time Clock data of the TLServer (i.e. the PC in which the TLServer runs on). The type of data is indicated in the Date[n] and Time[n] parameter which correspond to the DATE[n] and TIME[n] system variables in TBASIC: i.e. Date[1] = year; Date[2]=month; Date[3]=day; Date[4]=DayofWeek; Time[1]=hour; Time[2]=minute; Time[3]=second. For full synchronization, use the <READ RTC[]> tag which returns the values of the Date[1], Date[2], Date[3], Date[4], Time[1], Time[2], Time[3] in 7 CR-terminated ASCII strings. Upon receiving this command the TLServer will immediately send the relevant clock/calendar data as CR-terminated ASCII string(s) to the PLC. Your PLC program can easily use the INPUT$(1) command to read in the data and convert them into integers using the VAL command. Note that unlike the "READ file" service, the TLServer does not send "<OK>" string after performing the "READ RTC" service.

6. Get IP Address **Format:** **<IP>**	This command allows the PLC to query the TLServer's IP address. Upon receiving this command, the TLServer will send the following response string: If successful: xxx.xxx.xxx.xxx:nnnn (IP address:port of TLServer)

	If error occurred: ERR: (description of error) This command does not require a </> to close the tag.

7. Connect to Remote TLServer or FServer using NETCMD$

Format: **<CONNECT [IP address:port of TLServer or FServer]>**
[username string]
[password string]

TLServer replies with “<CONNECTED>” if successful or “ERR:…” if error encountered

[host link command 1]

TLServer sends the host link command to a remote TLserver or FServer and return the response to the PLC.

[host link command 2]

TLServer send the host link command to remote TLServer or FServer and return the response to the PLC.

.....

</>

Description: This service enables a 'Super' PLC to communicate with other PLCs via TCP/IP protocol using the TLServer and FServer or any combination of these. It allows your PLC to login to another remote TLServer or FServer via the LAN or the Internet and your PLC can then use the NETCMD$ command to exchange data with the PLC(s) that are connected to that remote server.

You execute this command by first sending the string **<CONNECT** xxx.xxx.xxx.xxx:9080**>** to the TLServer where xxx.xxx.xxx.xxx is the IP address of the remote TLServer or FServer. Then followed by the username and password needed to login to the remote server. Once the connection is established, the TLServer will return the response string **<CONNECTED>** to the PLC. When the PLC receive the **<CONNECTED>** string, it can use the TBASIC “NETCMD$” command to read or write data to the remote PLC as if the remote PLC is on the same local network. Multiple NETCMD$ commands can be executed but there should not be more than 2 seconds delay between each command. This is because If the FServer does not receive any serial string from the PLC for more than 2 seconds it will terminate the **<CONNECT>** session automatically. Once the command exchange has been completed, you'll send a **</>** tag to end the connection gracefully.

Note: similar to local RS485 communication, the source PLC that talks across the network using the **<CONNECT>** service should have a different ID (00 to FF) from the target PLC that it is trying to talk to. E.g. A PLC with ID=01 should not talk to another PLC with ID=01 even though the other PLC is connected to a remote server. By changing the ID of one of them, you will avoid the problem of a response string from the other PLC being misinterpreted by the sending PLC as an incoming host link command.

Examples of using the <CONNECT> tags with TLServer can be found in the TestEthernet.PC6 sample program in the following folder:

"C:\TRiLOGI\TL6\usr\samples\TLServer"

Note: *Your version of TRiLOGI may have the previous folder address, as shown below, but the PLC program is the same.*

"C:\TRiLOGI\TL6\usr\samples\XServer"

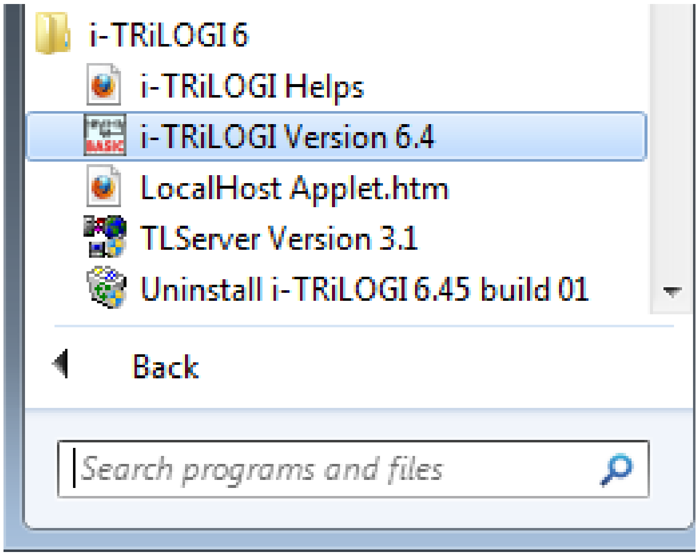
Chapter 5 Running The Internet TRiLOGI Client

5 RUNNING THE INTERNET TRILOGI CLIENT

5.1 Starting The i-TRiLOGI Application

Basically there are 3 methods in which you can start the i-TRiLOGI application, as follow:

1. If the i-TRiLOGI and JRE has been properly installed on your PC, you can just click on the "Start" button and select "i-TRiLOGI 6.x" in the "i-TRiLOGI 6" as follow:



2. You can also open My Computer and open the folder: C:\TRiLOGI\TL6\, then double click on the file "TL6.bat" to start i-TRiLOGI Application. Note that during the installation of the language pack the TL6.bat file will be replaced by the "TL6.bat" specific to that installed language. E.g. For running TL6 with Chinese language support, the TL6.bat will contain the following command line:

```
java -Duser.language=zh -jar TL64.jar
```

   The command line above forces the i-TRILOGI to run with the Chinese language (Zh) encoding. As such, do not start the i-TRiLOGI application by double-clicking on the "TL64.jar" file directly since that will only run i-TRiLOGI in the US English language instead of the language of choice.

3. The third alternative is to run the program from DOS command line: First, run the MS-DOS prompt (aka Command Prompt in Windows XP) and then navigate to the directory "C:\TRiLOGI\TL6" (assuming that is the folder

where you've installed i-TRiLOGI). At the directory, enter the following command line:

```
C:\TRiLOGI\TL6> java   -jar -Duser.language=xx   TL64.jar
```

where xx is the ISO defined two characters definition of the language code. E.g. "es" for Spanish, "ko" for Korean and "zh" for Chinese.

NOTE: The main-window window size is "memorized" in the config.tl6 file.

HELP!!!

When running i-TRiLOGI, you can get on-line help any time by pressing the <F1>. A Help window will open to show you the typical key/mouse actions. You can also click on the <More Help> button to get context-sensitive help loaded into your web-browser. i-TRiLOGI version 6.x and up now uses the default browser of your computer to display help data If the "config.tl6" line does not contain the path to an alternative browser in the: “Browser Path=” line (which it doesn't by default).

However, If you do not want to use the PCs default browser, then you'll need to use the "Notepad" program to manually edit the "config.tl6" file in the "C:\TRiLOGI\TL6\" directory. For example: If you wanted to use Internet Explorer as the browser for i-TRiLOGI, you will need to Modify the first line in "config.tl6" as follows:

```
Browser Path=C:/Program Files/Internet Explorer/IEXPLORE.EXE
```

to match the correct browser path info. There is no need to configure the applet browser path since the TL6 Applet automatically uses the same browser in which it was loaded to open the help files. Hence, the applet does not need to know the browser path at all.

5.2 Running TRiLOGI Applet Using Web Browser

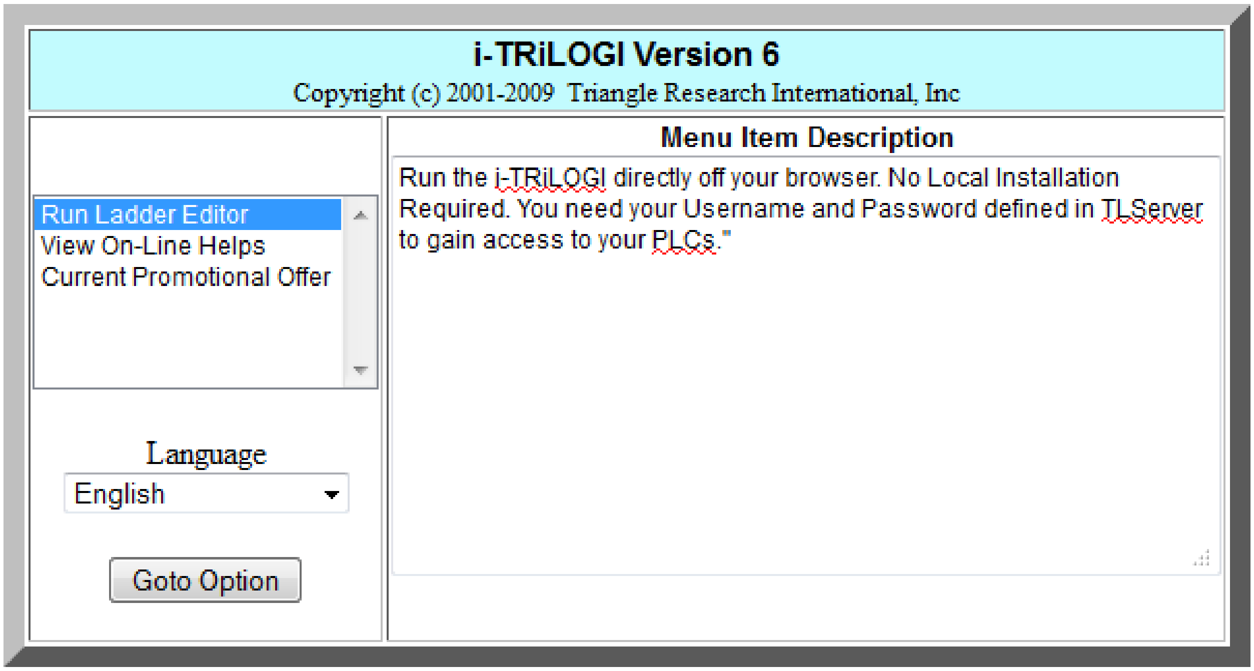
1. Before you could run the TRiLOGI Applet in a web browser, make sure that the TLServer 3.0 is already running.

2. Next, start up your Internet Browser. It should be either an Internet Explorer version 5.0 or later, Netscape Navigator/Communicator Version 4.5 or later or Mozilla Firefox version 1.5 or later.

3. Next, check the TLServer front panel for its IP Address. If you are running TLServer on a PC without network connection it will probably show: IP Address = 127.0.0.1:9080. If you have an Internet connection before you start up TLServer, then you will see the Internet IP address of your PC. If your PC has both a local area network connection as well as a direct Internet connection, you will see two IP addresses being reported. (Although localhost address 127.0.0.1 may not be reported but it is always there as long as both the Client and the Server reside on the same computer. Always use the localhost IP address 127.0.0.1 if both the Client and the TLServer are running on the same computer.)

4. Now, simply key in the IP Address, including the port number in your browser's "Address" (for IE5) or "URL" (for Netscape) text entry area. For localhost connection, enter:

   http://127.0.0.1:9080



i-TRiLOGI Version 6

Copyright (c) 2001-2009 Triangle Research International, Inc

Menu Item Description

Run the i-TRiLOGI directly off your browser. No Local Installation Required. You need your Username and Password defined in TLServer to gain access to your PLCs."

5. The browser will now issue a HTTP request to the TLServer. Since no filename has been specified, the default file in the web-server root directory "index.html" is loaded. This HTML file is written in Javascript to provide some other options. To start the TRiLOGI as an applet, select the appropriate option and the TL6Applet.jar file will be loaded from the TLServer into your browser for execution.

Note: The TLServer's root directory is not the same as the PC's root directory. In TLServer, the root directory is actually at "C:\TRiLOGI\TL6\public\". This is the directory where the index.html and TL6Applet.jar file are stored and these files are served to the web browser when you enter the TLServer's IP Address as mentioned above. Visitors have no access to the PC's file directory above the server's root directory so the content of your other PC files will not be at risk of being exposed to visitors to TLServer.

Disabling TRiLOGI Applet

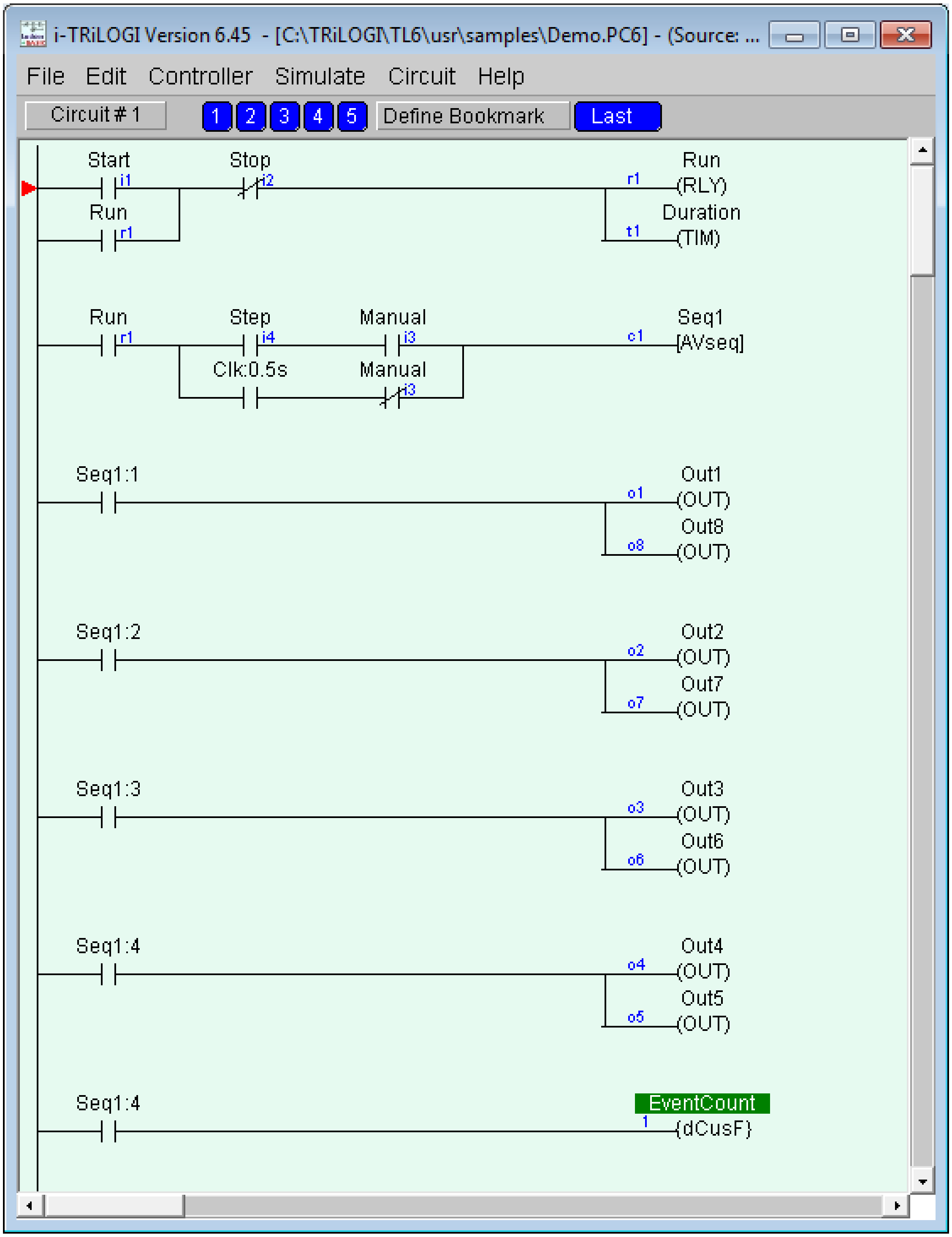
If you want to prevent visitors to TLServer from loading TRiLOGI Applet at all, just remove the "TL6Applet.jar" file from the directory: "C:\TRiLOGI\TL6\public\". In that case you can only access the PLC using the TL6 application program.

Chapter 6 Ladder Logic Programming Tutorial

6 LADDER LOGIC PROGRAMMING TUTORIAL

Your Assignment: Creating Your First Ladder Logic Program

In this tutorial, we would like to create a simple program as shown below:



Simply follow the steps below to create your first ladder logic circuit.

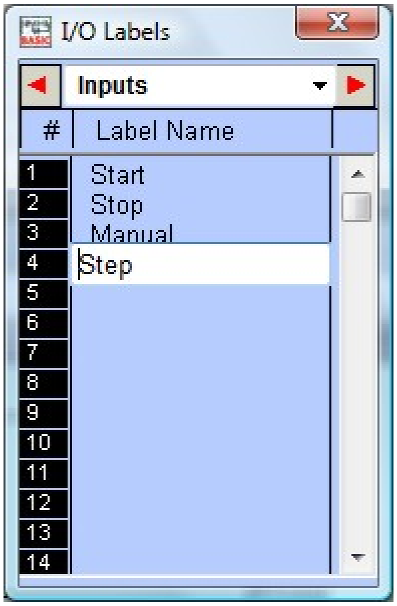
- Open pull-down "File" menu and select "New".
- You should now be in the "Browse" mode of the logic editor. The vertical line on the left end of the screen is the "power" line. The cursor is at the position where you can key in your very first ladder logic.

You can also view a video version of this tutorial here.

6.1 Ladder Logic Programming Tutorial: STEP1

Before we commence the circuit creation, let us define the I/Os to be used for this program. The following I/Os are required:

Inputs :	Start, Stop, Manual, Step
Outputs :	Out1, Out2,.... Out8
Relays :	Run
Timers :	Duration
Sequencer (counter) :	Seq1



Start
Stop
Manual
Step

1. Click on the [I/O Table] button located on right hand side of the upper status bar to open up the I/O label editing Window. You can also achieve the same with the <F2> function key.
2. Scroll to the "Inputs" window by using the left/right cursor keys or by clicking on the red color left/right arrow buttons or simply select it from the choice box between the left/right arrow buttons.
3. Move the deep blue color highlight bar to Input #1 position by clicking on it. Click again to open up a text field for entering the name for Input #1.

4. Press <Enter> key again and the highlight bar will be moved to Input #2.

5. Without using the mouse button, simply start typing the name "Stop" at Input #2. The text field will be automatically opened up at Input #2 for entry. Press <Enter> after typing in the name for "Stop" input.

6. Complete entry of the other two input label names "Manual" and "Step" as above. Note that i-TRiLOGI Version 6.2 and up allows I/O label names of up to 16 characters. However, if you wish to keep compatibility with Version 5.x to 6.1x, you should use no more than 10 characters to define the I/O names. Also, white spaces between names are not acceptable and will be automatically converted to the underscore character ('_'). e.g. If you enter the name: "F series PLC" for an I/O, it will be accepted as "F_series_P".

7. After entering label names for Inputs #1 to #4, move to the "Output" table by pressing the right cursor key or by clicking on the right arrow button. Enter all the output and relay label names in their respective I/O tables. We will discuss the "Timer" table in the next step.

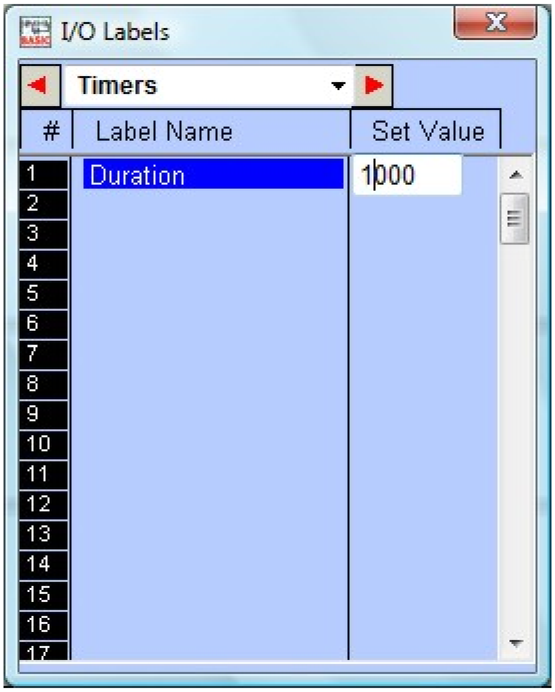
IMPORTANT

1. You can shift the Items in the I/O table up or down or insert a new label between two adjacent, pre-defined labels. Simply press the <Ins> key or Right-Click the mouse button to pop up the "Shift I/O" menu, which allows you to shift the selected I/O. However, please note that if you shift the I/O down, the last entry in the I/O table (e.g. Input #256) will be lost.

2. In i-Trilogi 6.2 and up, shifting of Custom Function Label names will now shift the function content along with the label name. (In previous versions of i-Trilogi, shifting of the I/O label would not shift the function content, therefore making it untenable to use I/O label shift to reorganize custom functions. Warnings are provided if such an action were to result in overwriting of an existing custom function.

3. i-TRiLOGI Version 6.2 and up allows I/O label names of up to 16 characters. However, if you wish to keep compatibility with Version 5.x to 6.1x, you should use no more than 10 characters to define the I/O names.

6.2 Ladder Logic Programming Tutorial: STEP 2



Duration

1000

1. Timer table has an extra column "Set Value" located to the right of the "Label Name" column.

2. After you have entered the label name "Duration" for Timer #1, a text entry box is opened up at the "Set Value" location of Timer #1 for you to enter the SV for the timer. SV range is between 0 and 9999. Enter the value 1000 at this location.

3. For a normal timer with 0.1s time base, the value 1000 represents 100.0 seconds, which means that the "Duration" timer will time-out after 100.0 seconds. If the timer had been configured as "High Speed Timer" using the TBASIC "HSTimer" command, then the time-base would become 0.01s, meaning the value 1000 represents only 10.00 seconds.

4. We are now left to define the sequencer, "Seq1". The sequencer is an extremely useful device for implementing sequencing logic found in many automated equipment. i-TRiLOGI supports 8 sequencers of 32 steps each. Each sequencer requires a "Step counter" to keep track of the current step sequence.

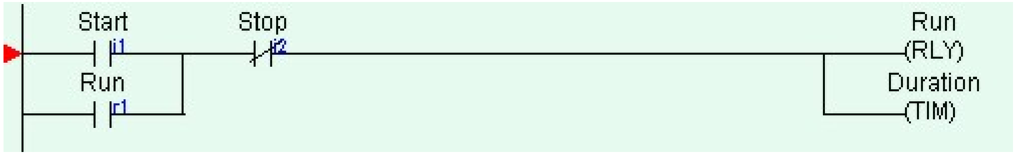
   The first 8 counters in the counter table double as the step counters for the 8 sequencers. These sequencers must be named "Seq1" to "Seq8" if they are to be used, i.e. Counter #1 to be named as "Seq1", Counter #2 as "Seq2", etc. However, any counter not used as sequencer may assume any other name (up to a maximum of 10 characters) if they are used as ordinary counters.

   If you are at the "Timers" table, pressing the right cursor key again will bring up the "Counters" table. Enter the name: "Seq1" at the label column for Counter #1. Press <Enter> and the text entry field will be opened at the "Set Value" column. For now, let's enter a preset value of "4" for "Seq1".

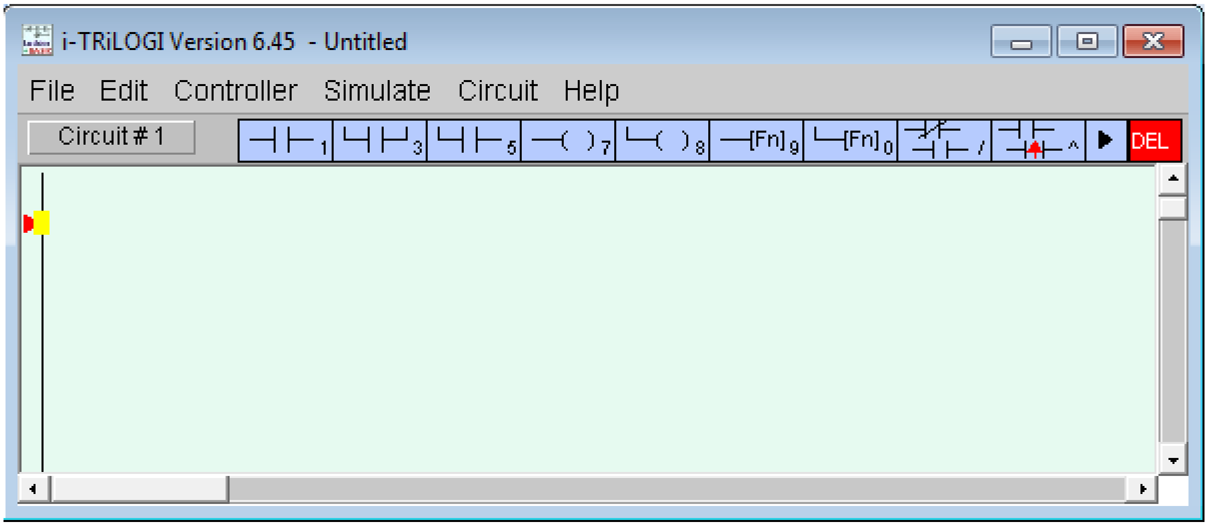
5. We have now completed defining the I/Os, timers and counters. Press the <ESC> key to close the counter or other tables. Note that not all labels need to be defined before programming. You may create the label names any timer during circuit creation by pressing hotkeys <F2>.

6.3 Ladder Logic Programming Tutorial: STEP 3

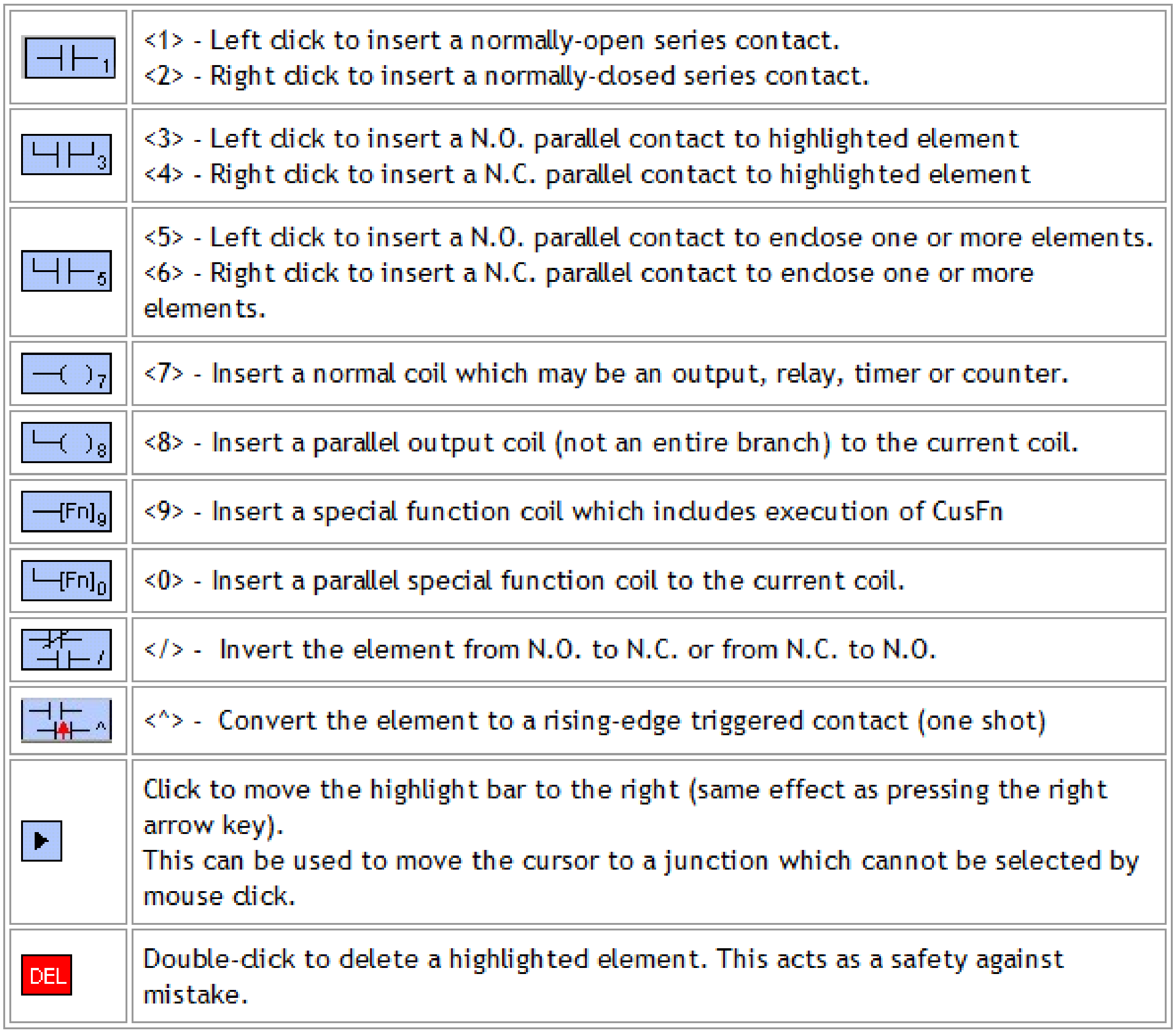
We are ready to create Circuit #1 as shown below:



1. With the circuit pointer (red color triangle) at Circuit #1, press the <Spacebar> to enter the "Ladder Edit" mode. You can also enter the circuit edit mode by double clicking at Circuit #1.



Once you enter the "Ladder Edit" mode, a row of ladder icons appear along the top of the main i-TRiLOGI window just below the pull down menu. The following is a description of each item. A yellow color highlight bar, which you can move to select an element in the ladder circuit, will appear.

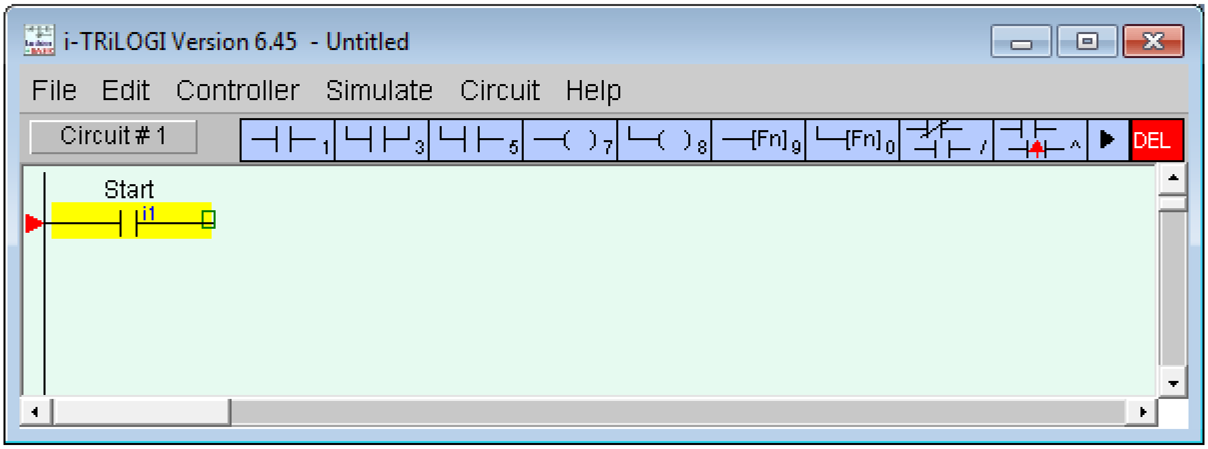


┤├1	<1> - Left click to insert a normally-open series contact. <2> - Right click to insert a normally-closed series contact.
└┤├3	<3> - Left click to insert a N.O. parallel contact to highlighted element <4> - Right click to insert a N.C. parallel contact to highlighted element
└┤├5	<5> - Left click to insert a N.O. parallel contact to enclose one or more elements. <6> - Right click to insert a N.C. parallel contact to enclose one or more elements.
─( )7	<7> - Insert a normal coil which may be an output, relay, timer or counter.
└( )8	<8> - Insert a parallel output coil (not an entire branch) to the current coil.
─[Fn]9	<9> - Insert a special function coil which includes execution of CusFn
└[Fn]0	<0> - Insert a parallel special function coil to the current coil.
/	</> - Invert the element from N.O. to N.C. or from N.C. to N.O.
^	<^> - Convert the element to a rising-edge triggered contact (one shot)
▶	Click to move the highlight bar to the right (same effect as pressing the right arrow key). This can be used to move the cursor to a junction which cannot be selected by mouse click.
DEL	Double-click to delete a highlighted element. This acts as a safety against mistake.

2. Now insert the first element by left-clicking on the ┤├1 icon. The icon will change to a bright yellow color to show you the element type that you are creating. At the same time, an I/O table should appear on the screen with a light beige-color background instead of the normal light blue background. The I/O table now acts like a pop-up menu for you to pick any of the pre-defined label names for this contact.

   Note: In i-TRiLOGI version 6.x, if you pick any undefined I/O you will be prompted to enter the label name and what you entered will automatically be updated in the I/O table.

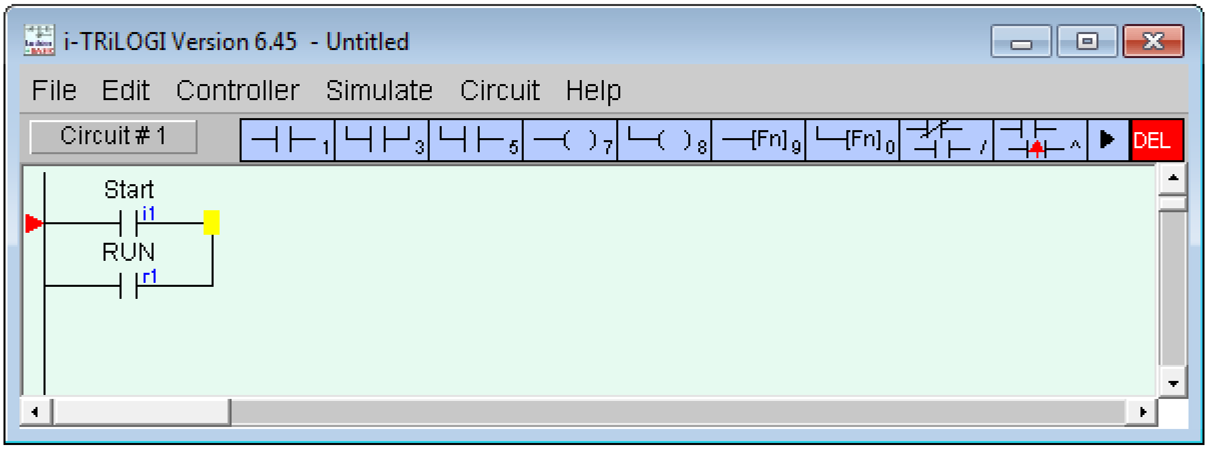
3. The contents in the table are not normally meant to be edited at this moment . Scroll to the "Input" table and click on the label name "Start" and a normally-open contact will be created at Circuit #1.



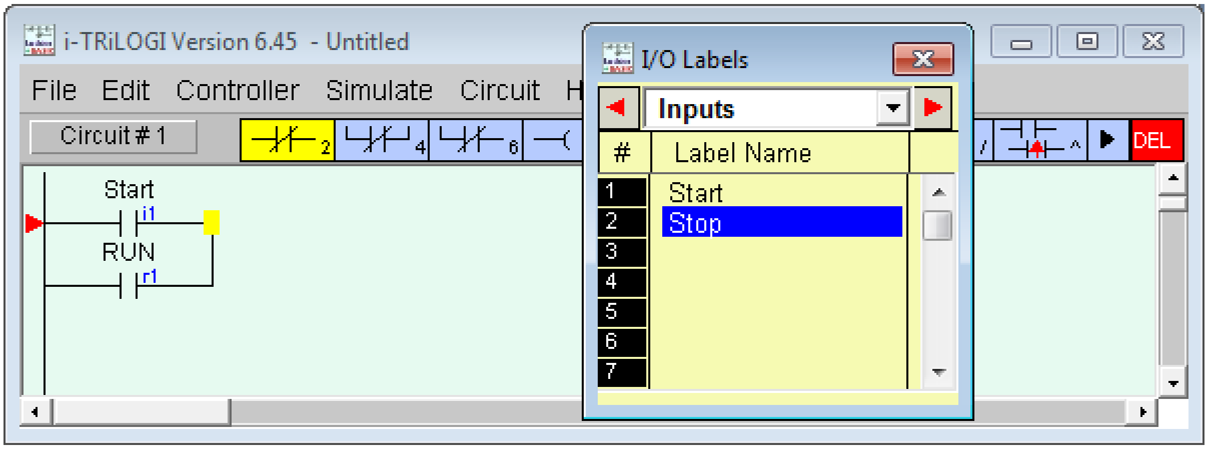
If you observe the highlight bar carefully, you will notice a dark green color square at the right end of the highlight. This indicates the insertion location where a series contact will be attached. You can change the insertion location to the left or the right of the highlight bar by pressing the <SHIFT> key or <TAB> key (Note: <TAB> key only works on JRE 1.3.1 or earlier. It does not work on JRE 1.4.x)

4. Next, create the contact "RUN" which is parallel to the "Start" contact by left-clicking on the ┘├┘├3 icon. The I/O table will appear again. Scroll to the "Relay" table and select the "RUN" relay.

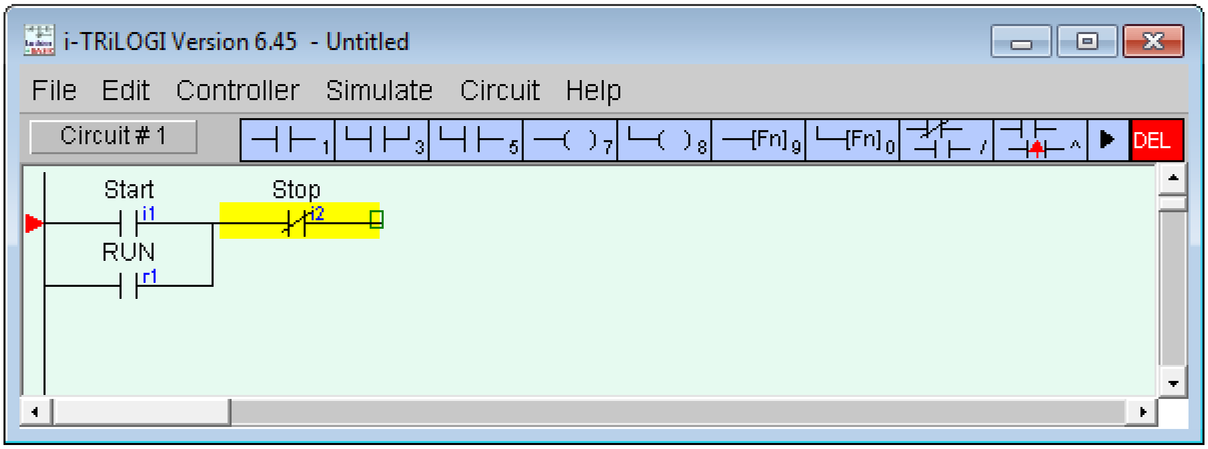
5. To insert the normally-closed "Stop" contact in series with the "Start" and "Run" contacts, you need to move the highlight bar to the junction of the "Start" and "Run" contact. First click on the "Start" contact to select it. Then click on the ▶ icon to move the highlight bar to the junction, as follow:



Note: The highlight bar will move to the junction if you click on the dark green insertion point on the "Start" contact.



6. Next, right-click on the ─┤├─1 icon. It will change into yellow color normally-closed contact as shown in the above diagram. You are now inserting a normally-closed series contact at the location of the highlight bar. Pick the "Stop" label from the "Input" table to add the series contact.



7. We will now connect a relay coil "Run" to the right of the "Stop" contact. Click on the ─( )7 icon to insert the coil. Select "RUN" label from the "Relay" table. Remember that an input can never be used as a coil. Fortunately, i-TRiLOGI is smart enough not to call up the "Inputs" table when you are connecting a coil, to avoid unintentional errors.

Notice that the coil symbol ---(RLY) indicates that this is a relay coil, which is helpful in identifying the function of the coil. i-TRiLOGI automatically places the coil at the extreme right end of the screen and completes the connection with an extended wire.

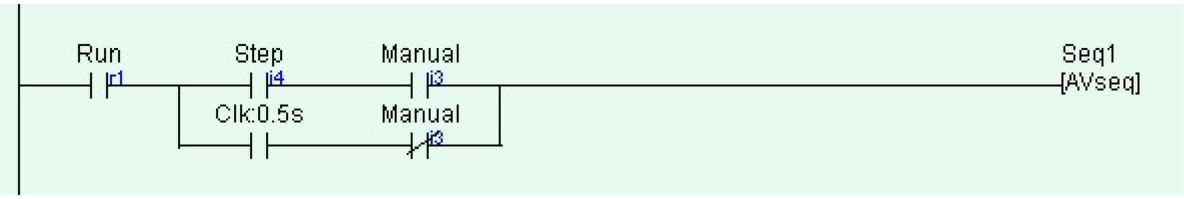
8. Right below the relay coil is a parallel timer coil with label name "Duration". To create this coil, click on the └( )8 icon. This allows you to connect a parallel coil to the existing coil. The "I/O" table will pop up for selection again. Since we want to choose a timer, scroll to the "Timer" table and pick the first timer with the label "Duration" to complete the circuit.

   Press the <Enter> key once to complete Circuit #1

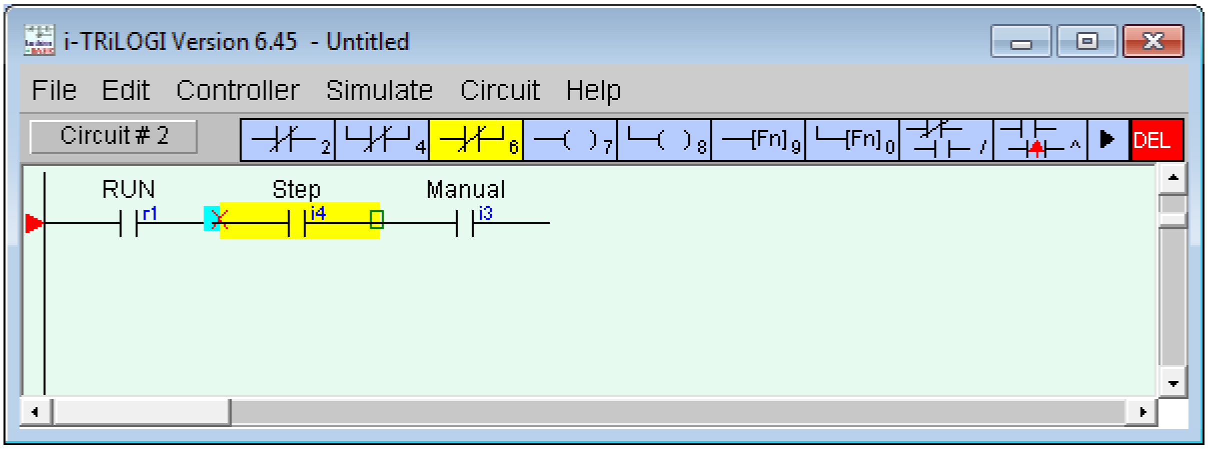
9. Congratulations! You have just successfully created you very own ladder logic circuit. It is that simple! It may be a good time to save your program now by pressing <CTRL-S> key or select "Save" from the "File" menu and give a file name for you new program.

6.4 Ladder Logic Programming Tutorial: STEP 4

We will now create Circuit #2 as shown below.



1. Follow the steps listed in STEP 3 to create the following circuit fragment:

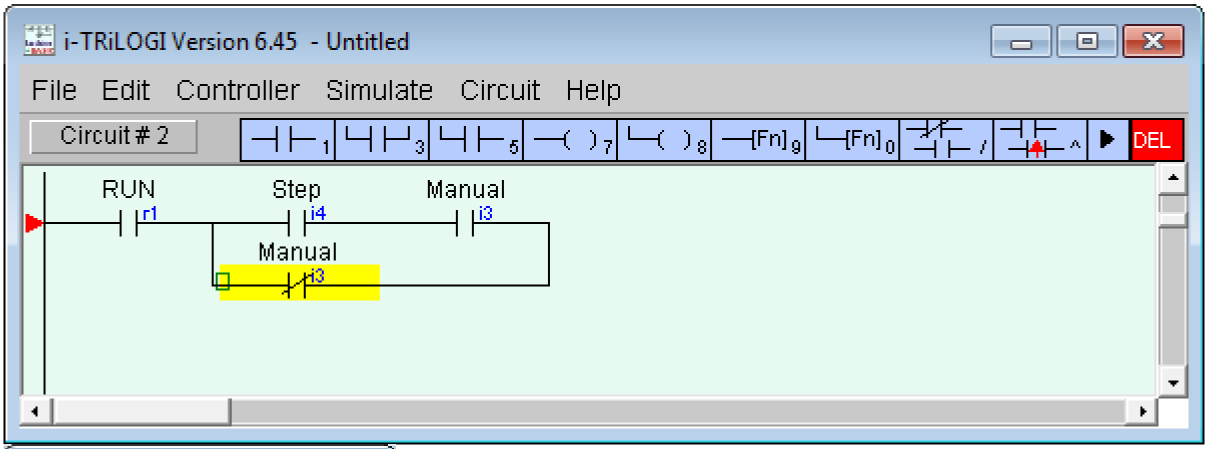


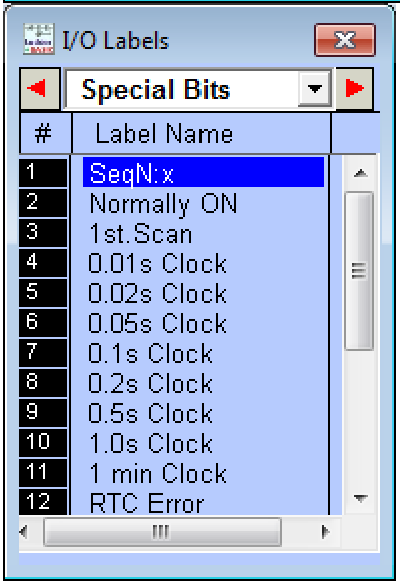
2. We want to enclose the two series contacts "Step" and Manual" with a parallel branch that contains two elements. First, we will create the branch for the N.C. "Manual" contact.

3. Click on the element "Step" to highlight it. Then right-click on the ┤├5 icon to create a N.C. parallel circuit that encloses both the "Step" and the "Manual" contacts. A cross will appear at the left hand end of the "Step" contact, indicating that this is the starting location of the parallel circuit. You should now click on the "Manual" contact to select the ending location for the parallel circuit. The yellow highlight bar will be positioned at "Manual" contact now.

4. You will notice that the ┤├5 icon has now changed into a yellow color N.C. contact ┤/├6 with an opposite connection arm. You should now click on the ┤/├6 symbol to close the parallel branch. (One possible short-cut method is to double-click at the ending location to close the branch).

   As usual an I/O table will be opened for you to select the I/O. For now, select the "Manual" label from the "input" table to create the following circuit:



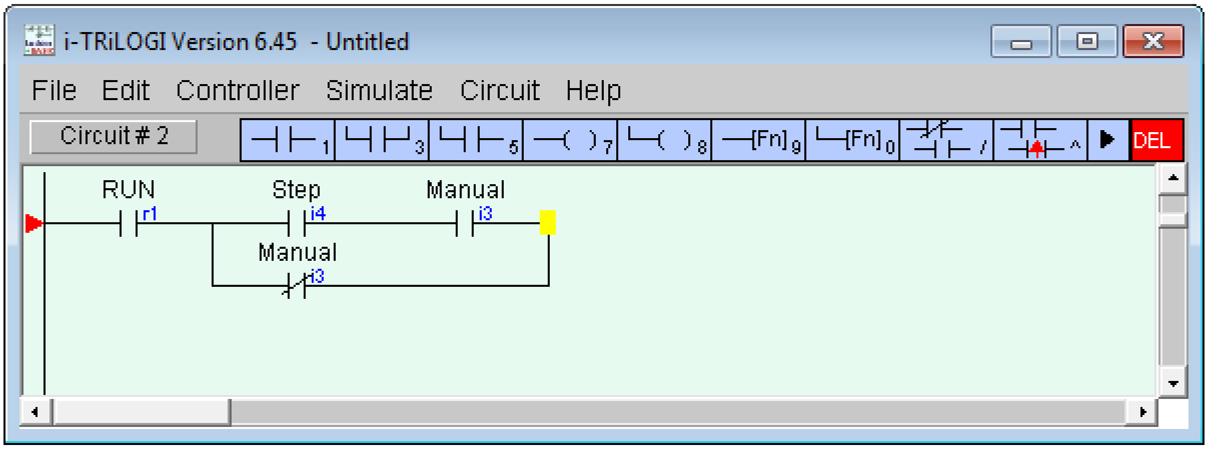


#	Label Name
1	SeqN:x
2	Normally ON
3	1st.Scan
4	0.01s Clock
5	0.02s Clock
6	0.05s Clock
7	0.1s Clock
8	0.2s Clock
9	0.5s Clock
10	1.0s Clock
11	1 min Clock
12	RTC Error

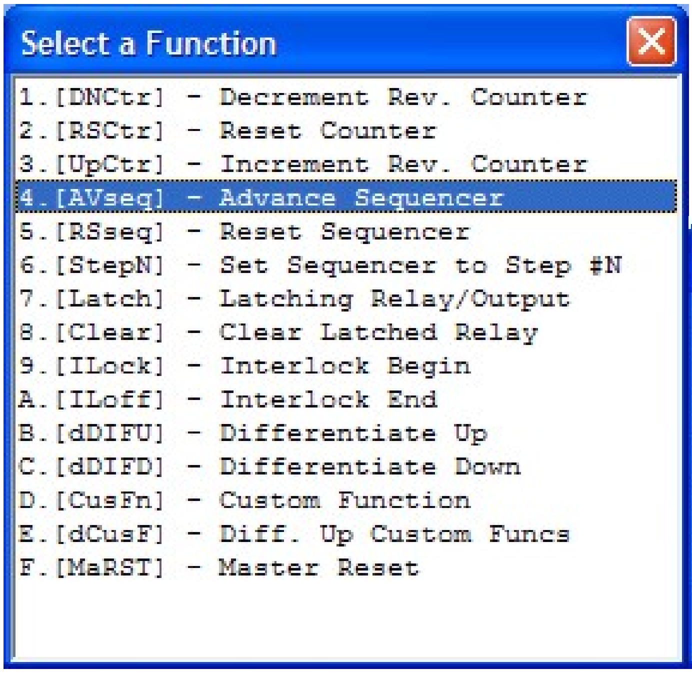
5. Next, we want to insert the special bit "Clk:0.5s" to the left of the "Manual" contact. Press the <SHIFT> key to move the insertion point to the left end of the highlight bar as shown above. Then left-click on the ┤├1 icon to create a normally-open contact. Scroll the I/O table to the "Special Bits" table and select the item: "0.5s Clock". The parallel branch would have been completed by now.

   Note: The "Special Bit" table comprises some clock pulses and some other special purpose bits. These include the eight built-in clock pulses in the system with periods ranging from 0.01s to 1 minute. Built-in clock pulses are useful if you need a time base to create, for example, a "flashing light". A contact such as "Clk:0.1s" will automatically turn itself ON for 0.05s and then OFF for another 0.05s and then ON again, resulting in a series of clock pulses of period = 0.1 second.

6. Next, move the highlight bar to the right end junction of the parallel circuits as follow:

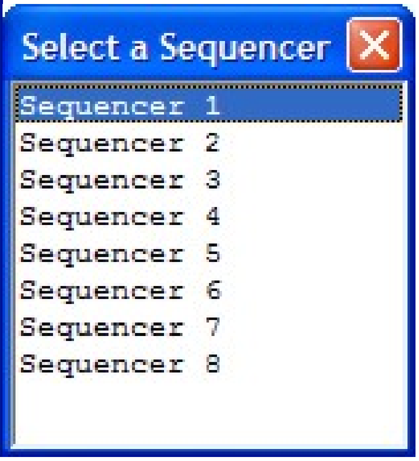


7. Now, click on the —[Fn]9 icon to insert a special function coil. A popup menu will appear for you to select the desired special function. Click on the item "4.[AVseq]-Advance Sequencer" to insert the Advance Sequencer function [AVseq].



Select a Function

1.[DNCtr] - Decrement Rev. Counter
2.[RSCtr] - Reset Counter
3.[UpCtr] - Increment Rev. Counter
4.[AVseq] - Advance Sequencer
5.[RSseq] - Reset Sequencer
6.[StepN] - Set Sequencer to Step #N
7.[Latch] - Latching Relay/Output
8.[Clear] - Clear Latched Relay
9.[ILock] - Interlock Begin
A.[ILoff] - Interlock End
B.[dDIFU] - Differentiate Up
C.[dDIFD] - Differentiate Down
D.[CusFn] - Custom Function
E.[dCusF] - Diff. Up Custom Funcs
F.[MaRST] - Master Reset



Select a Sequencer

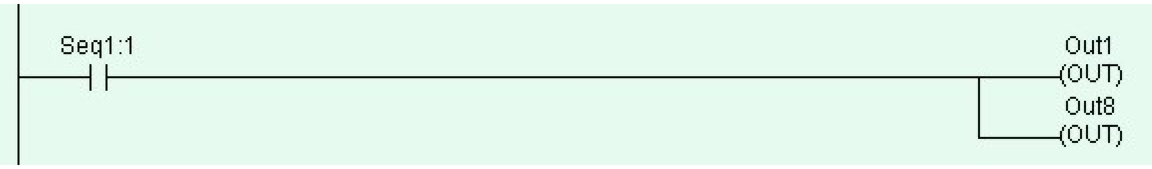
Sequencer 1
Sequencer 2
Sequencer 3
Sequencer 4
Sequencer 5
Sequencer 6
Sequencer 7
Sequencer 8

8. When prompted, select Sequencer 1. This function will increment the step counter of Sequencer #1 each time its execution condition goes from OFF to ON.

   Again, remember to press the <Enter> key to complete Circuit #2

6.5 Ladder Logic Programming Tutorial: STEP 5

1. Circuits #3 to #6 are similar to one another. They make use of the Sequencer to turn on the Outputs 1 to 8 to create a pattern of "running lights" when executed. The label "Seq1:1" of the contact in Circuit #3 represents Step #1 of Sequencer 1. Remember that each sequencer can have up to 32 steps (Step #0 to 31), with each step individually accessible as a contact. A normally-open contact "Seq1:1" will be closed whenever the step counter of Sequencer 1 reaches number 1. Likewise a normally-closed contact "Seq5:20" will be opened when the step counter of Sequencer 5 reaches number 20.



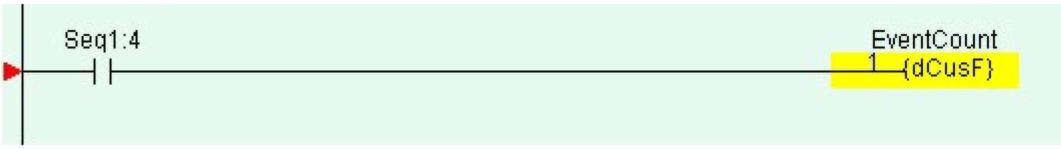
2. To create the normally-open contact "Seq1:1", left-click on the icon. When the I/O table pops up, scroll to the "Special Bit" table and select the

item #1 "SeqN:x". When prompted to select a sequencer choose "Sequencer 1" and another dialog box will open up for you to enter the specific step number for this sequencer. At this point, you should enter the number "1" since we are using Step #1.

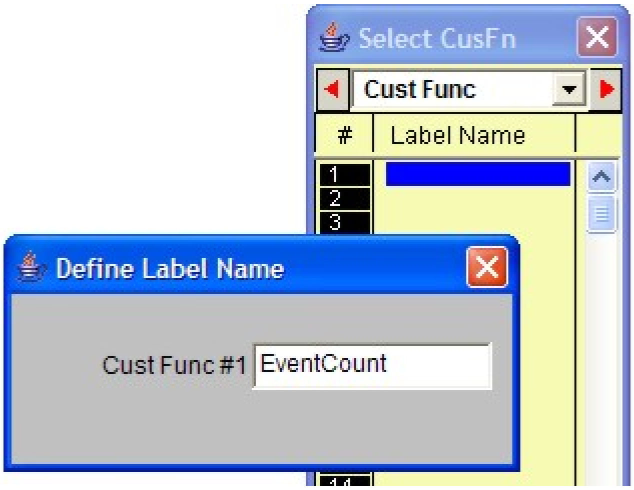
3. We have thus far been creating ladder circuits only by clicking on the ladder icons. A short-cut method of choosing elements to be created without using the mouse does exist. Observe the icon carefully and you will notice a small numeral at the lower right hand corner of each icon which correspond to the shortcut key. You may wish to try this short-cut for the remaining part of Circuit #3. Press the <7> key and the Output table will immediately pop up for selection of a coil. Pick "Out1" from the "Output" table and the "Out1" coil will be connected.

4. Circuits #4, 5 and 6 are very similar to Circuit #3 and you shouldn't have problem creating them. Complete these circuits and we are ready for some interesting simulation exercises. When you have created all the circuits, press <Enter> key or <ESC> key at the last blank circuit to end "Ladder Edit" mode.

5. Next, we want Seq1:4 contact to execute a "Custom Function" (the abbreviation is CusF). A custom function is one that you can create to perform very sophiscated control actions using the TBASIC language. The Custom function can contains multiple TBASIC statements that gives the PLC extremely powerful capabilities unmatched by many ladder-only type of PLCs. The Circuit we wish to create is as follow:



6. First, create the contact "Seq1:4" as per previous circuits. Then left-click on the —[Fn]9 to connect to a Special Function coil. Select from the popup menu the item "E: Diff. Up Custom Function" to create a "Differentiated Up" version of Custom Function (For a full explanation of the meaning of Differentiated Up", please refer to TBASIC Introduction). The Custom function popup menu will appear for you to select the Custom Function number you wish to insert. Up to 256 Custom functions can be defined and you can click on the first custom function, as follows:

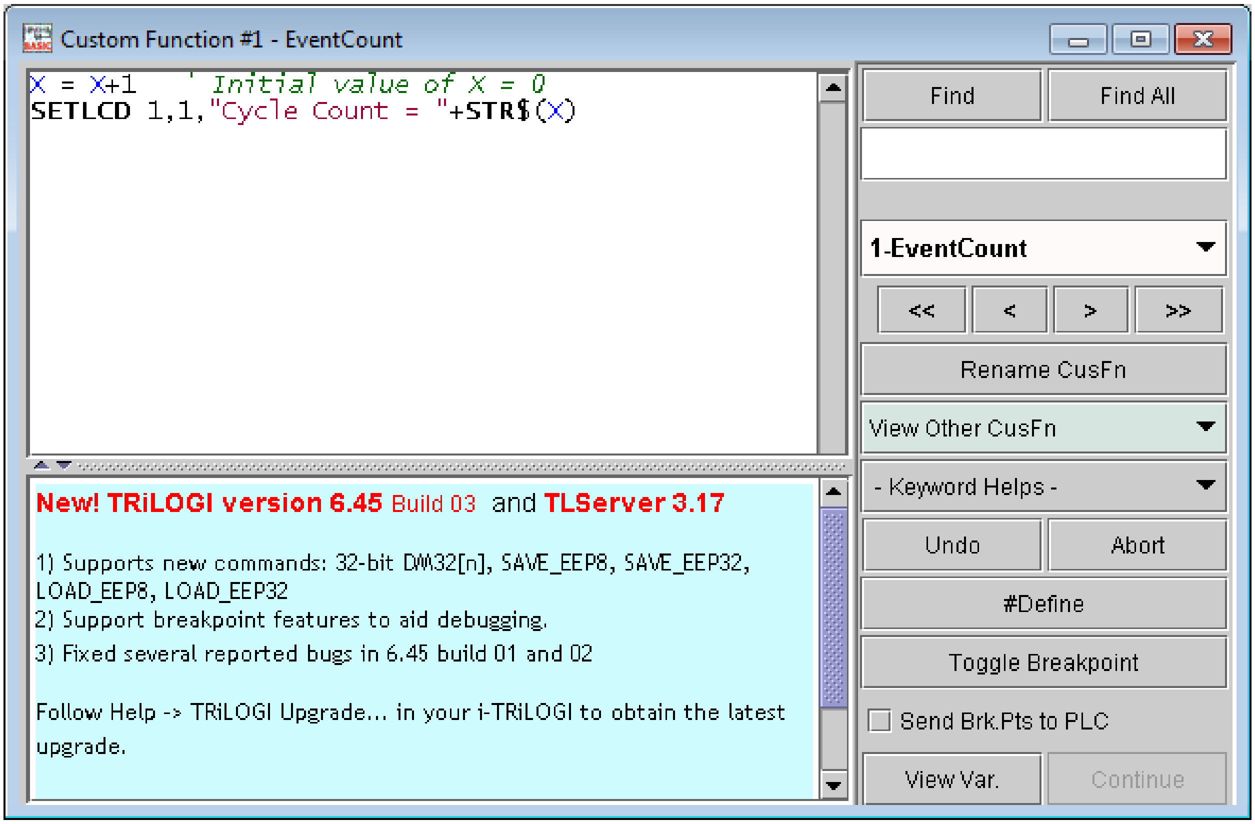


Cust Func #1 EventCount

7. Since the custom function has not been defined with a label name, you will be prompted to give it a label name. The label name given to the CusF must be unique and are subject to the same limitations as that of the I/O label names. Lets enter the label name "EventCount" for function #1.

   Note: Label name for Custom function is optional and if you don't wish to use label name then the program will automatically assign it a name "Fn_#1" for function #1, "Fn_#2" for function #2, and so on.

8. When the Custom Function editor has been opened, enter the following text and then close the editor window by click on the ☒ or by pressing the <ESC> key. If you would like to exit the editor window without saving the changes, click on the Abort button.



```
X = X+1     ' Initial value of X = 0
SETLCD 1,1,"Cycle Count = "+STR$(X)
```

New! TRiLOGI version 6.45 Build 03 and **TLServer 3.17**

1) Supports new commands: 32-bit DM32[n], SAVE_EEP8, SAVE_EEP32, LOAD_EEP8, LOAD_EEP32
2) Support breakpoint features to aid debugging.
3) Fixed several reported bugs in 6.45 build 01 and 02

Follow Help -> TRiLOGI Upgrade... in your i-TRiLOGI to obtain the latest upgrade.

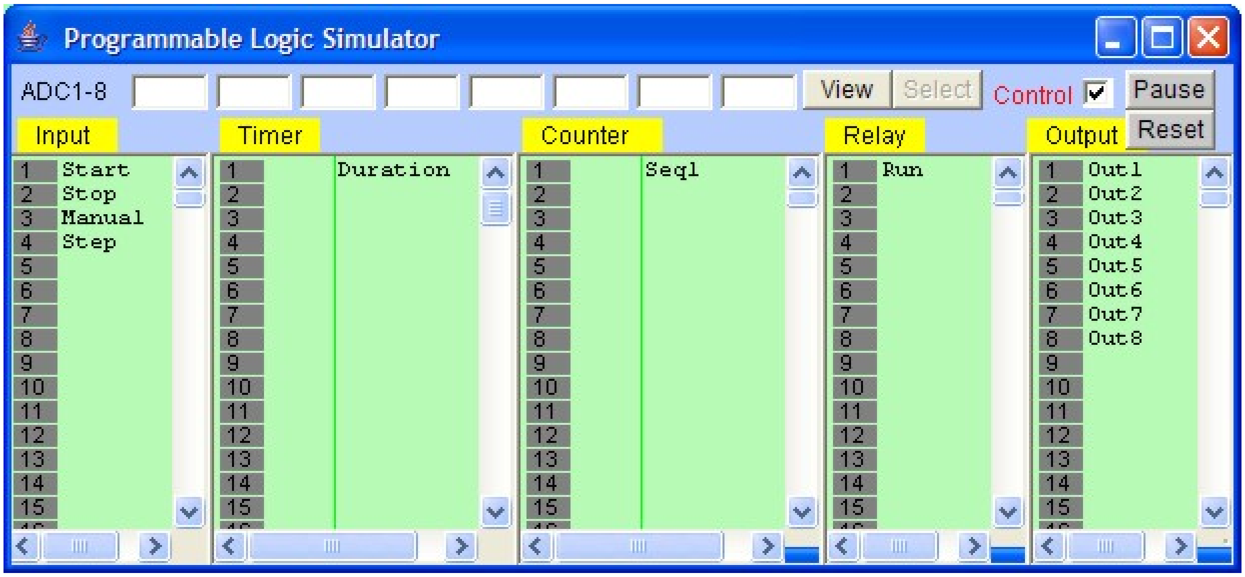
The purpose of this custom function is to keep a cycle count of the number of times the Sequencer has completed the complete sequencing and display the cycle count on the PLC's built-in LCD display (or on the Simulator's "View Variable" screen). This custom function should be run ONLY ONCE whenever the Seq1:4 contact closes, which is why the differentiated version of Custom function must be used in this case ({dCusF}, not {CusFn})

9. We can make our program more comprehensive to other users by utilizing the "Comments" feature of i-TRiLOGI. Open the "Circuit" menu and select "Insert Comment". A comment editor window will be opened up to allow you to add your comments to any part of the circuit. When you are done with your comments, just press <ESC> key or close the comment editor window and the comments you just entered will be inserted between the circuits. Each comment occupies a circuit position and there is no limit to the number of lines a comment circuit may have. (However, if you wish to keep data file compatibility with the old DOS TRiLOGI Version 4.x you should limit the comment to no more than 4 lines per comment and each line should contain no more than 70 characters.)

   A comment circuit may be moved around or deleted just like any other ladder circuits. If you wish to edit the comment, just double-click on it or press the <Spacebar> to open up the comment editor window. You can use the normal text editing keys such as left, right, up, down cursor keys, and <Ctrl-Left>, <Ctrl-Right>, <Del> and <Backspace> keys for editing the comment text.

6.6 Ladder Logic Programming Tutorial: STEP 6

The stage has been set and the show is ready! Having completed the demo program, it is time to test if it works as intended using the built-in real-time programmable controller simulation engine. Open the "Simulate" pull-down menu and activate the command "Run (All I/O reset) - Ctrl+F9". i-TRiLOGI will immediately compile the ladder program and if no error is detected, it will instantly proceed to open up the "Programmable Logic Simulator" screen, as shown below:



Input		Timer		Counter		Relay		Output	
1	Start	1	Duration	1	Seq1	1	Run	1	Out1
2	Stop	2		2		2		2	Out2
3	Manual	3		3		3		3	Out3
4	Step	4		4		4		4	Out4
5		5		5		5		5	Out5
6		6		6		6		6	Out6
7		7		7		7		7	Out7
8		8		8		8		8	Out8
9		9		9		9		9	
10		10		10		10		10	
11		11		11		11		11	
12		12		12		12		12	
13		13		13		13		13	
14		14		14		14		14	
15		15		15		15		15	

1. If you have followed closely all the instructions during the creation of the demo program, you should not encounter any compilation error. However, if you do receive an error message, then please check your circuit against the picture shown in the assignment page, make all necessary corrections and then try again.

2. The simulator screen comprises 5 columns: Input, Timer, Counter/Sequencer, Relay, and Output. With the exception of the Relay table which contains up to 512 elements, and the Timer table which contains up to 128 timers, all other columns contain 256 elements each. Every column has its own vertical scroll bar. You can use the mouse to scroll each column independently to locate the desired I/O.

3. The label names for the inputs, outputs, relays, timers and counters defined earlier in the I/O tables automatically appear in their respective columns. To the left of each label name column is an "LED" lamp column which indicates the ON/OFF state of respective I/O. A red color lamp represents the ON state of an I/O, whereas a dark grey color lamp represents an OFF state. The I/O number is indicated in the middle of the lamp.

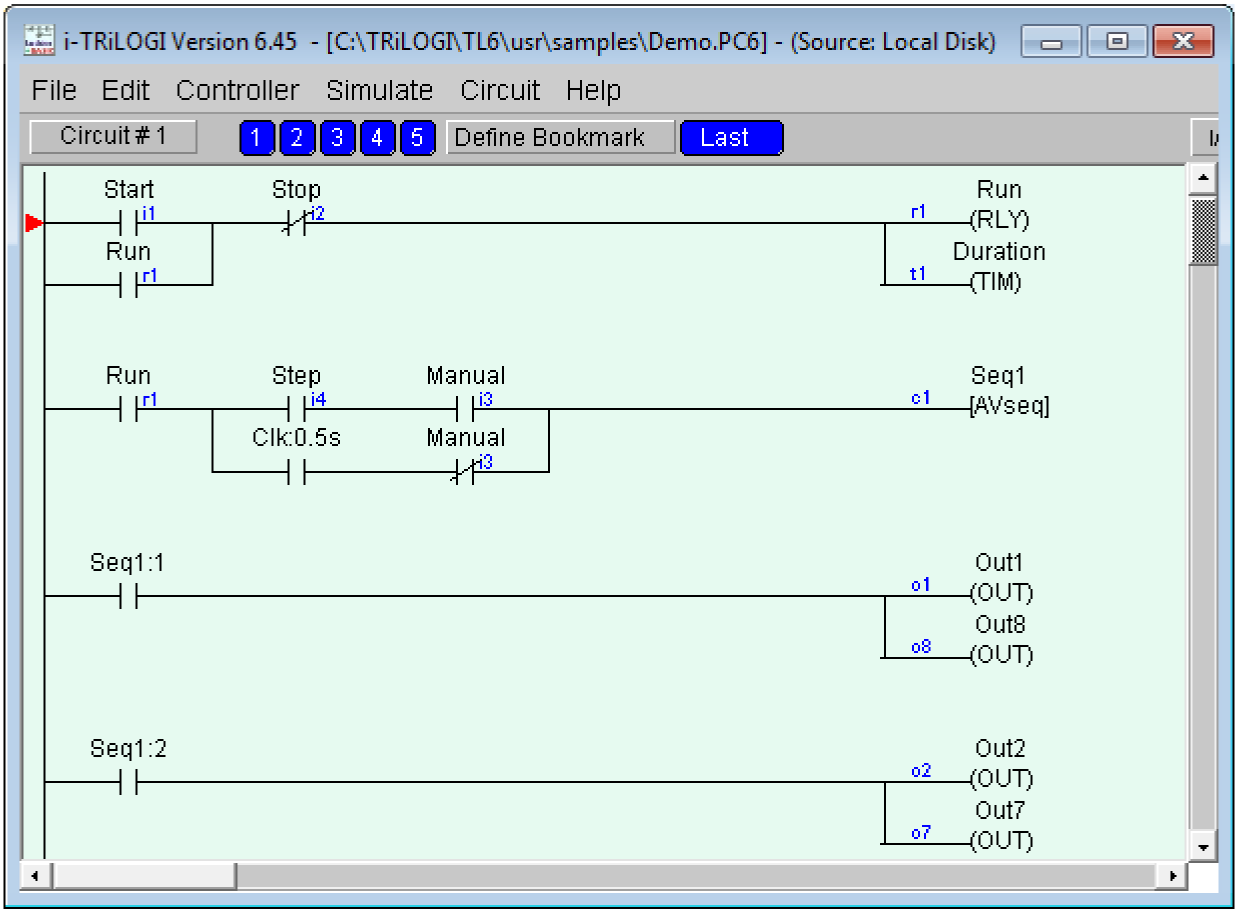
   The simulator requires the use of the mouse to work properly, so it is important to remember the mouse button actions as follows:

Left Mouse Button	Turn ON the I/O when pressed. Turn OFF when button is released.
Right Mouse Button	Toggle the I/O when pressed once. (i.e. OFF becomes ON and ON become OFF)

4. Our ladder program requires us to "push" the "Start" button momentarily. You can simulate this action by moving the mouse pointer to the "Start" label (or the LED lamp) and press the LEFT mouse button once and then release the button. The action starts!

5. At this time, notice that the relay "RUN" is latched ON and the timer "Duration" begins to count down from the value of 1000 every 0.1sec, and the Output #1-#8 are turning ON/OFF sequentially in a "running light" pattern. Sequencer "Seq1" in the "Ctr/Seq" column begins to count upward from 1 to 3 and then overflows to 0 and repeats continuously. For each step of the Sequencer, the corresponding Output will be turned ON. Our demo program will show a running light pattern starting from Outputs 1 & 8, then 2 & 7, 3 & 6 and 4 & 5 and then back to 1 & 8, 2 & 7.....

6. Now you should verify that the logic works as intended by observing the ladder diagram. You should notice that the "Run" labels in all circuits are highlighted as shown below:



7. The logic states of any I/O can be displayed on the ladder diagram directly. An Input, Output, Relay, Timer or Counter contact that is turned ON will have its label name highlighted in the ladder diagram. This feature helps greatly in debugging and understanding the logical relationship between each I/O. For example, from the above figure, we can see clearly that the "Self-latching" circuit for relay "Run" works as intended: when we first turn ON the "Start" input, "Run" will be energized and its contact which is parallel to "Start" will hold itself in the ON state, even if we subsequently turn OFF the "Start" input by releasing the button.

8. The timer coil "Duration", being connected in parallel to "Run" relay, will also be energized. However, its contact will only be closed after 100 seconds (when its present value counted to 0). To break the latched On "Run" relay, we must energize the "Stop" input momentarily to break the "power" flow. Try it now.

9. Let's restart the system by turning ON the "Start" input momentarily again. Next, we want to turn ON the "Manual" input. Move the mouse pointer to the "Manual" input and then press the right mouse button. "Manual" input will be

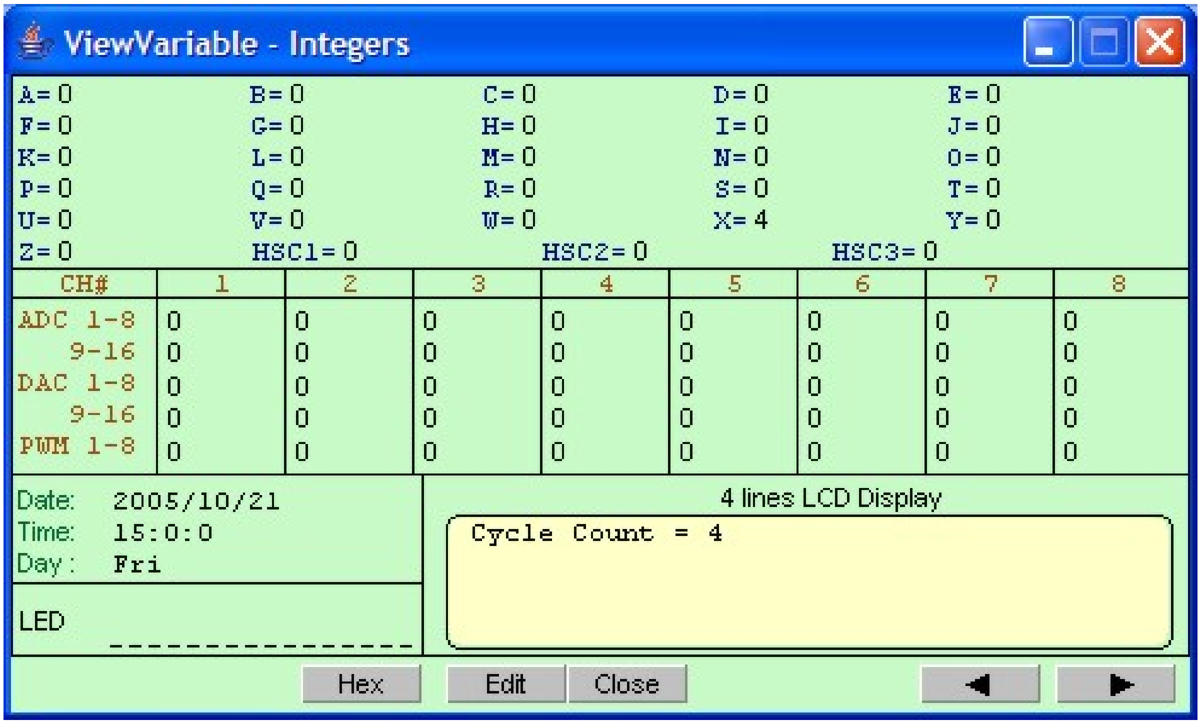
"stuck at "ON" state even after you have released the right mouse button. Click on "Manual" button using the right mouse button again and it will be turned to OFF.

10. When "Manual" input has been turned ON, the running lights should stop. This is because the normally-closed contact of the "Manual" input in Circuit #2 is now turned OFF and the 0.5s clock pulse could not trigger the [AVseq] function anymore.

11. If you now left-click on the "Step" input, the running lights will move one step at a time in response to your mouse click. Observe the Seq1:x contact with respect to the counter value of Seq1 and the logic of this circuit become very clear instantly.

12. Observe that the timer "Duration" continues to count down every 0.1 second, and when it reaches 0, the "Duration" output coil label will be highlighted. You can use this timer to stop the program by connecting a N.C. "Duration" contact to Circuit #1. This is left as an exercise for you!

13. You can also observe the execution of the Custom Function #1 by clicking on the "View" button on the "Programmable Logic Simulator" screen and the following window will appear:



A= 0	B= 0	C= 0	D= 0	E= 0
F= 0	G= 0	H= 0	I= 0	J= 0
K= 0	L= 0	M= 0	N= 0	O= 0
P= 0	Q= 0	R= 0	S= 0	T= 0
U= 0	V= 0	W= 0	X= 4	Y= 0
Z= 0	HSC1= 0	HSC2= 0	HSC3= 0	

CH#	1	2	3	4	5	6	7	8
ADC 1-8	0	0	0	0	0	0	0	0
9-16	0	0	0	0	0	0	0	0
DAC 1-8	0	0	0	0	0	0	0	0
9-16	0	0	0	0	0	0	0	0
PWM 1-8	0	0	0	0	0	0	0	0

Date: 2005/10/21
Time: 15:0:0
Day : Fri

LED ----------------

4 lines LCD Display

```
Cycle Count = 4
```

14. This screen shows the values of TBASIC variables A to Z and many other data. It also provides a simulated 4 lines LCD display that show case the action of the SETLCD statement contained in Custom Function #1. As you can see, the value of variable X used in the Custom function is also shown in the variable section. The value of X is converted by the STR$(X) statement into a string and displayed after the text string "Cycle Count = " at line #1, column #1 of the LCD area.

Summary

We have completed this hands-on session and have successfully created a simple ladder +BASIC program. We have also performed real time simulation to test the program's functionality. By now you would probably have a good appreciation of i-TRiLOGI's superb capability and ease of use and are ready to include i-TRiLOGI as an integral part of your programming needs.

Chapter 7 i-TRiLOGI Ladder Logic Editor

7 I-TRILOGI LADDER LOGIC EDITOR

i-TRiLOGI's ladder logic editor window lies between the main menu bar along the top of the screen and the help message line along the bottom of the screen. The cursor will appear in the window whenever you are in the logic editor. The ladder logic editor comprises two modes: the browse mode and the circuit editing mode. We shall explain the operation of both modes.

7.1 The Browse Mode

You are normally in the browser mode when you start up the program. The browse mode allows you to manipulate the whole ladder logic circuit as a single entity: you can view any circuit, make copies of it, move it to another location or delete it entirely. Each complete ladder logic "circuit" is given a circuit number. You should see a small red color marker showing you the currently selected circuit. The circuit number of the selected circuit is shown on the upper status line as "Circuit # xxx ".

7.1.1 Mouse Actions

Since i-TRiLOGI Version 6 runs under windowing environment, all usual mouse action applies. You can grab the vertical scroll bar to scroll to your desired circuit and click on it to select it. Double click on a circuit enters the Circuit Editing Mode which will be described later.

7.1.2 Keyboard Actions

The functions of various keys in the browse mode are explained below:

1. <Spacebar> - Allows you to enter circuit editing mode for the currently selected circuit. If the selected circuit is a comment circuit, the comment editor will be opened automatically.

2. <F1> - Activates the context-sensitive help function to display on-line help.

3. <F2> - Opens the I/O Table to create the I/O Label Names

4. <F3> - Turns ON/OFF display of the I/O type and the physical I/O number for ladder logic contacts on the screen (The physical I/O number is now linked to the labelname). All ladder logic contact symbols are normally identified by their label names. However, you can also display an optional small

literal to indicate the I/O types and number. e.g. i1=input 1, o12=output 12, r25= relay 25, t1= timer 1 and c5=counter 5.

5. <F5> - Refreshes the display. If for some reason the screen is garbled by incomplete circuit display, you can just press the <F5> key to redraw the screen.

6. <F7> - Opens any custom function. If the currently selected circuit contains a custom function, then it will be opened for editing. Otherwise i-TRiLOGI will ask you to select a custom function # from a menu.

7. <F8> - Compiles the i-TRiLOGI program to show the compilation statistics.

8. <F9> - Runs the simulator without resetting any I/O.

9. <Ctrl-F9> - Resets all I/Os and then runs the simulator.

10. <Ctrl-F8> - Resets all I/Os except inputs and then runs the simulator.

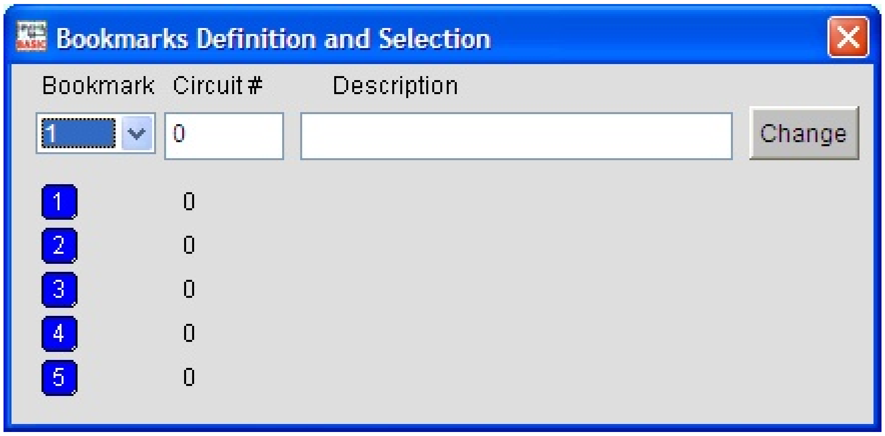
11. <Up>/<Dn><PgUp> and <PgDn> keys - Use the up/down cursor keys to move the marker to other circuits and the "Circuit #" display at the upper status line will simultaneously reflect the change. If you attempt to venture beyond the screen, the logic editor screen will scroll. The <PgUp> and <PgDn> keys can be used to scroll one page at a time.

7.1.3 Navigating Circuits in Browse Mode

Although most of the time you will probably be navigating the ladder circuit diagram using the scroll bar or cursor keys to find the circuit that you are looking for, there exist several short cuts to enable you to quickly jump to specific circuits in the program. E.g. If you select "Goto" from the "Edit" menu, you will be prompted by a “Goto Circuit” window to enter the circuit number to jump to immediately. You can also call up the “Goto Circuit” window by pressing <Ctrl-G> key anytime.

If you have a large ladder program with multiple rungs of circuits, then you will find the “Bookmark” feature very useful for quickly jumping from one circuit to another. The bookmarks appear as blue button along the top of the i-TRiLOGI editor windows: 1 2 3 4 5 You can define up to five bookmarks by clicking on Define Bookmark, each bookmark corresponds to a specific circuit #. By default all 5 bookmark points to circuit #1. Once you have defined the circuit # for a bookmark, then when you click on the bookmark button the editor instantly jumps to the pre-defined circuit number.

When you click on Define Bookmark, the following window will appear



Bookmarks Definition and Selection

Bookmark	Circuit#
1	0
2	0
3	0
4	0
5	0

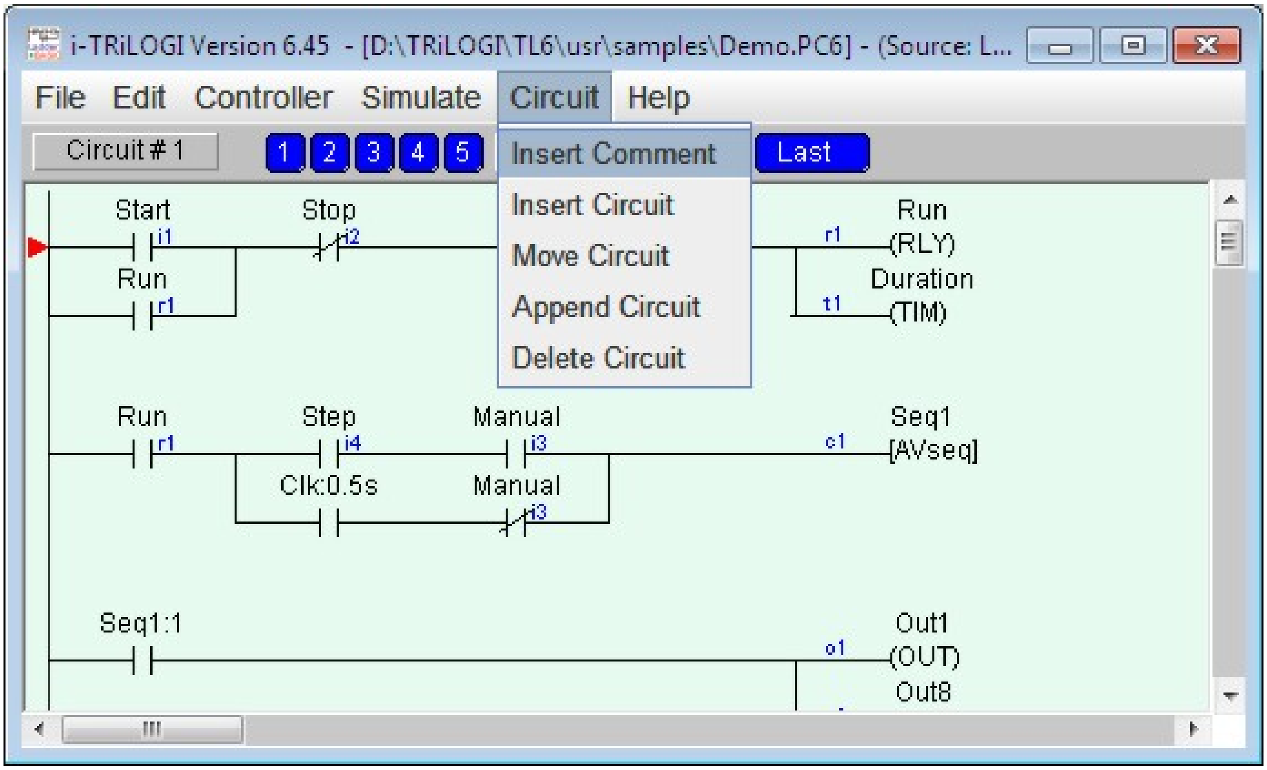
You can select the bookmark No. either from the choice box or by clicking on the blue bookmarks: 1 and 2. Then enter or edit the circuit # to associate with the selected bookmark. You can optionally enter/edit a description for the bookmark to help you recall the purpose of that section of ladder circuit which the bookmark is associated with. Then click the Change button to update the circuit # and description at the corresponding bookmark location.

In i-Trilogi version 6.2 and later the bookmark you have created will be displayed in the corresponding ladder circuits. New programs created will set all the bookmarks to a default value of 0 so that no bookmarks are displayed until they have been manually created.

In i-Trilogi version 6.2 and later there is a feature that allows you to immediately go the last circuit that was selected. This is implemented by clicking on the Last button that is located beside the bookmark buttons, which are described above. Whether you move to a new circuit by using the "Goto Circuit" command or a bookmark or just by clicking on a circuit with the mouse, the "Last" button will remember your last circuit position. This also allows you to easily switch back and forth between two circuits an unlimited number of times because if you don't select a new circuit after pressing the "Last" key, you will go back to the circuit you came from.

Note: The bookmark definitions are saved along with the program body into the ".PC6" file, which means that the bookmark that you defined for a particular program can be recalled later when you open its ".PC6" file.

7.2 The Circuit Menu



7.2.1 Insert Comments

Comments are specific remarks used by a programmer to explain various characteristics of a program segment and are ignored by the compiler. i-TRiLOGI Version 6 allows comments to be freely inserted between circuits. Execute this command and the Comment Editor will be opened. The comment editor allows you to enter any text you like that best describe the working of the circuit. All standard text editing keys, including cut and paste are applicable to the Comment Editor. When you have finished editing the comments, click "OK" to accept the changes and to close the comment window or click abort to cancel the changes and close the comment window. Alternatively, you can press the <ESC> key to accept the changes and close the comment window.

Once a comment has been created, it is assigned a circuit number and is treated like any other circuits. You can edit it by pressing the <spacebar> when you are in Browse mode, alternatively, you can move it around, copy it to another destination or delete it entirely using commands in the "Circuit" menu.

7.2.2 Insert Circuit

This command enables you to insert a new circuit just before the currently selected circuit. The current circuit number will be increased by one while the new circuit will assume the current circuit number. You will be placed in the circuit editing mode for immediate circuit creation.

7.2.3 Move Circuit

You can rearrange the order of the circuits by using this command. Select the circuit you wish to move and execute the "Move Circuit" command, then select a destination circuit location and press <Enter>. The selected circuit will be moved to the new location before the destination circuit.

Note that if you wish to move a block of circuits to a new location, you may find it more productive to use the "Cut Circuit" and "Paste Circuit" commands in the "Edit" menu.

7.2.4 Append Circuit

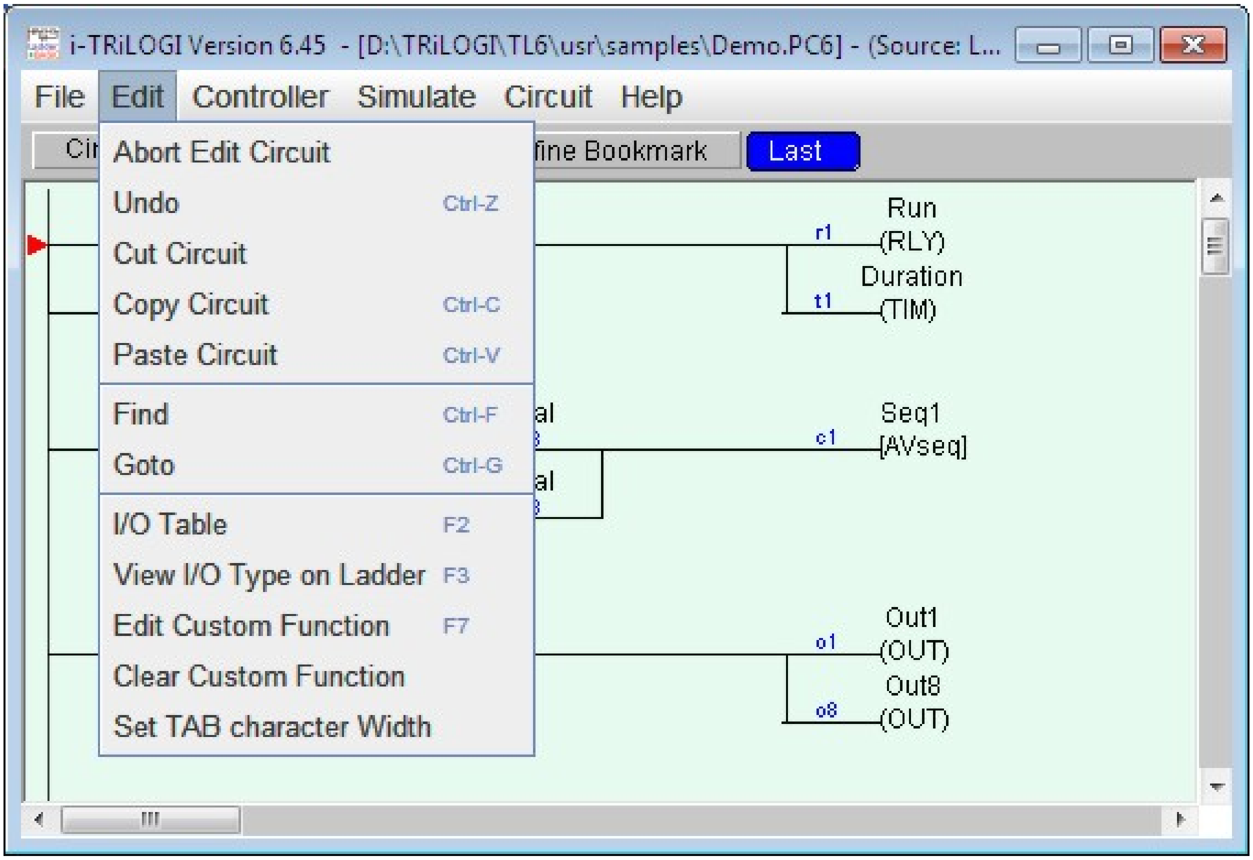
Execute this to add a new circuit to the ladder logic program. This new addition will be positioned immediately after the last circuit in the entire program.

7.2.5 Delete Circuit

This command allows you to delete the one or more circuits. You will be prompted to enter the range of circuits that you wish to delete.

Please note that you can't UNDO a delete circuit operation.

7.3 The Edit Menu



7.3.1 Cut Circuit

You can remove a number of circuits from the current ladder program and store them temporarily in the clipboard for pasting into another part of the present ladder program or into another file altogether. In other words, it lets you move a block of circuits from one part of the ladder program to another part or into another file.

Please note that you can't UNDO a Cut Circuit operation. However, if you do make a mistake you can always paste it back in its original position.

7.3.2 Copy Circuit (Ctrl-C)

You can copy a block of circuits from the current ladder program and store them into the clipboard for pasting into another part of the present ladder program or into another ladder program file altogether.

7.3.3 Paste Circuit (Ctrl-V)

When you execute this command, the block of ladder circuit which you "Cut" or "Copy" into the clipboard will be pasted just before the currently selected circuit. The current circuit number will be adjusted to reflect the change.

7.3.4 Find (Ctrl-F)

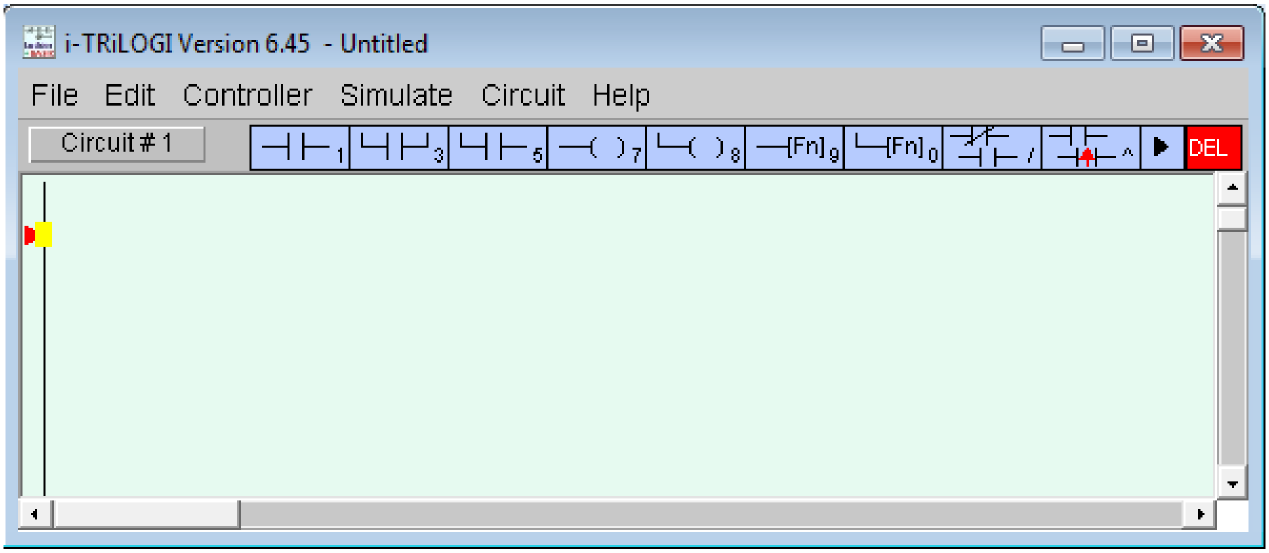
The Find command allows you to quickly locate a ladder logic circuit that contains a particular label name. The Find command can also be used to search for a keyword in a TBASIC program. When this command is executed, you will be further prompted to select the option of searching for a ladder logic label or a text in a Custom Function.

7.3.5 Goto (Ctrl-G)

Use this command to move towards a specific circuit number. The "Goto" command is particularly useful if your program contains many circuits, and it is inconvenient to search for a particular circuit using the mouse or the cursor keys.

7.4 The Circuit Editing Mode

i-TRiLOGI comes with a smart editor which allows you to insert or delete a single element within a circuit easily. The editor interprets your circuit immediately upon entry and prevents you from creating illegal circuit connections. The functions of various keys in the circuit editing mode are detailed below. You know that you are in the circuit editing mode when a row of ladder logic icons appears along the upper status line next to the circuit number and a yellow color highlight bar appears and you can move it to select an element in the ladder circuit, as shown below:

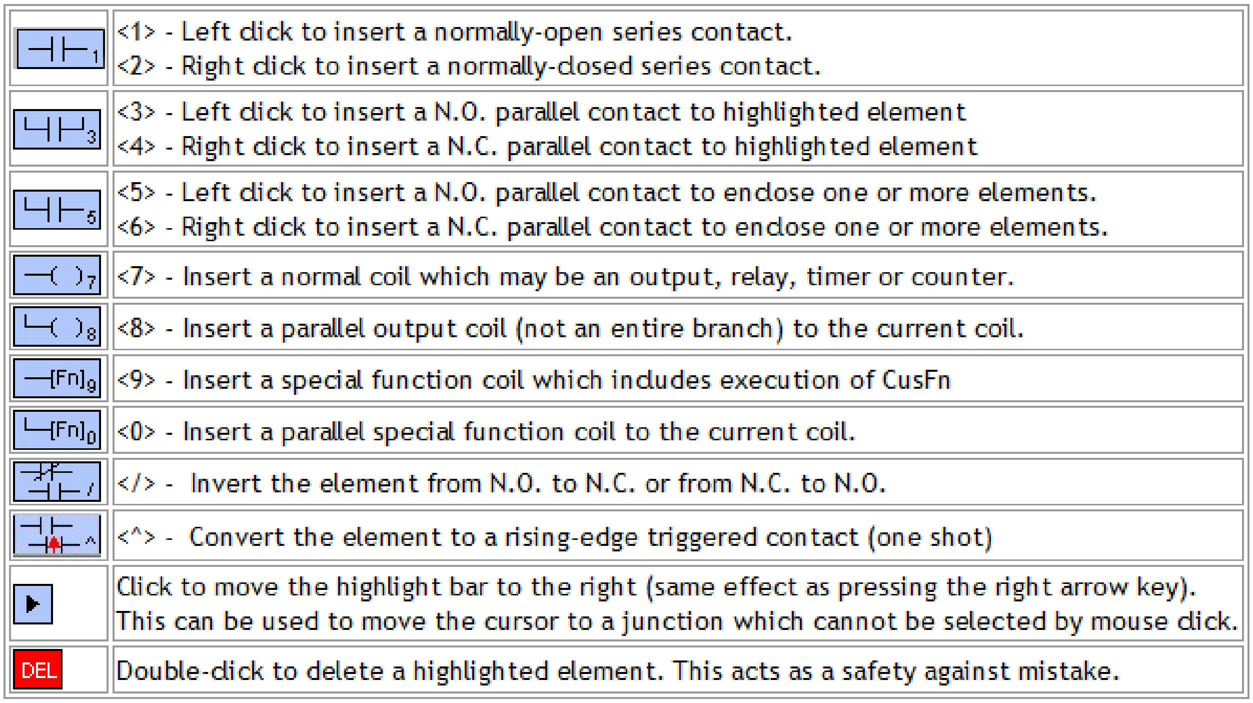

i-TRiLOGI Version 6.45 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 1
DEL

7.4.1 Mouse Actions

Left Click - When you click on an element using a the left mouse button, the element is selected and highlighted by the yellow color highlight bar.

Right Click - When you click on an element using the right mouse button, you are allowed to directly edit the label name of the element. This can be a convenient feature if you need to change one or two characters in the name only. However, if the element is a custom function [dCusFn], or [CusFn], then the custom function editor will be opened for you to edit the function directly.

Insert Ladder Element - You create the ladder circuit element simply by moving the mouse pointer to the icon and pressing either the left or the right mouse button to insert a ladder logic element to the currently highlighted element. The following is a description of the functions of each icon. A yellow color highlight bar will appear which you can move to select an element in the ladder circuit.



⊣⊢1	<1> - Left click to insert a normally-open series contact. <2> - Right click to insert a normally-closed series contact.
┤├3	<3> - Left click to insert a N.O. parallel contact to highlighted element <4> - Right click to insert a N.C. parallel contact to highlighted element
┤├5	<5> - Left click to insert a N.O. parallel contact to enclose one or more elements. <6> - Right click to insert a N.C. parallel contact to enclose one or more elements.
─( )7	<7> - Insert a normal coil which may be an output, relay, timer or counter.
└( )8	<8> - Insert a parallel output coil (not an entire branch) to the current coil.
─[Fn]9	<9> - Insert a special function coil which includes execution of CusFn
└[Fn]0	<0> - Insert a parallel special function coil to the current coil.
/	</> - Invert the element from N.O. to N.C. or from N.C. to N.O.
^	<^> - Convert the element to a rising-edge triggered contact (one shot)
▶	Click to move the highlight bar to the right (same effect as pressing the right arrow key). This can be used to move the cursor to a junction which cannot be selected by mouse click.
DEL	Double-click to delete a highlighted element. This acts as a safety against mistake.

When you click on an icon, ⊣⊢1 for example, the icon will change to a bright yellow color to show you the element type that you are creating. At the same time, an I/O table should appear on the screen with a light beige-color background. The I/O table acts like a pop-up menu for you to pick any of the pre-defined label name for this contact. This saves you a lot of typing and at the same time eliminates typo errors that could result in a compilation failure. You should spend a few minutes to follow the "Ladder Logic Programming Tutorial" on the steps needed to create a ladder program.

As mentioned previously, the ladder editor is intelligent and will only accept an action that can result in the creation of a correct ladder element. Otherwise it will simply beep and ignore the command.

UNDO Circuit Editing – If you have wrongly inserted or deleted an element and wish to undo the mistake, you can either select "Undo" from the "Edit" menu or press <Ctrl-Z> key to undo the last step. The undo buffer stores the last 10 editing steps. You can also choose to abort all the operations on the current circuit by selecting "Abort Edit Circuit" to abort all changes made to the current circuit.

7.4.2 Create Ladder Circuit Using The Keyboard

Users of existing TRiLOGI version 3.x or 4.x who are familiar with creating ladder programs using the keyboard will be delighted to know that they can still create their ladder programs using the keyboard. The keyboard actions are described below:

Left/Right/Up/Down cursor keys

The cursor keys are for moving the highlight bar from one element to another in their four respective directions. You can only move in a direction which will end up with an element.

<ESC>

Press <ESC> key to end the circuit editing mode and return to the browse mode of the logic editor.

<Enter>

When you are done with editing the current circuit, hit <Enter> to proceed to the next circuit.

<SHIFT> (or <TAB>)

If you observe the highlight bar carefully, you will notice a dark green color square at the right end of the highlight. This indicates the insertion location where a series contact will be attached. You can change the insertion location to the left or the right of the highlight bar by pressing the <SHIFT> key. (Note: prior to Version 5.32 only the <TAB> key was used for this purpose, but <TAB> key does not work on JRE 1.4.x so we added the <SHIFT> key to achieve this action).

The position of the insertion point has no effect when you connect a parallel contact to the highlighted element. The left terminal of the element will always be connected to the left side of the parallel branch.

<0> to <9> , </> & <E> keys

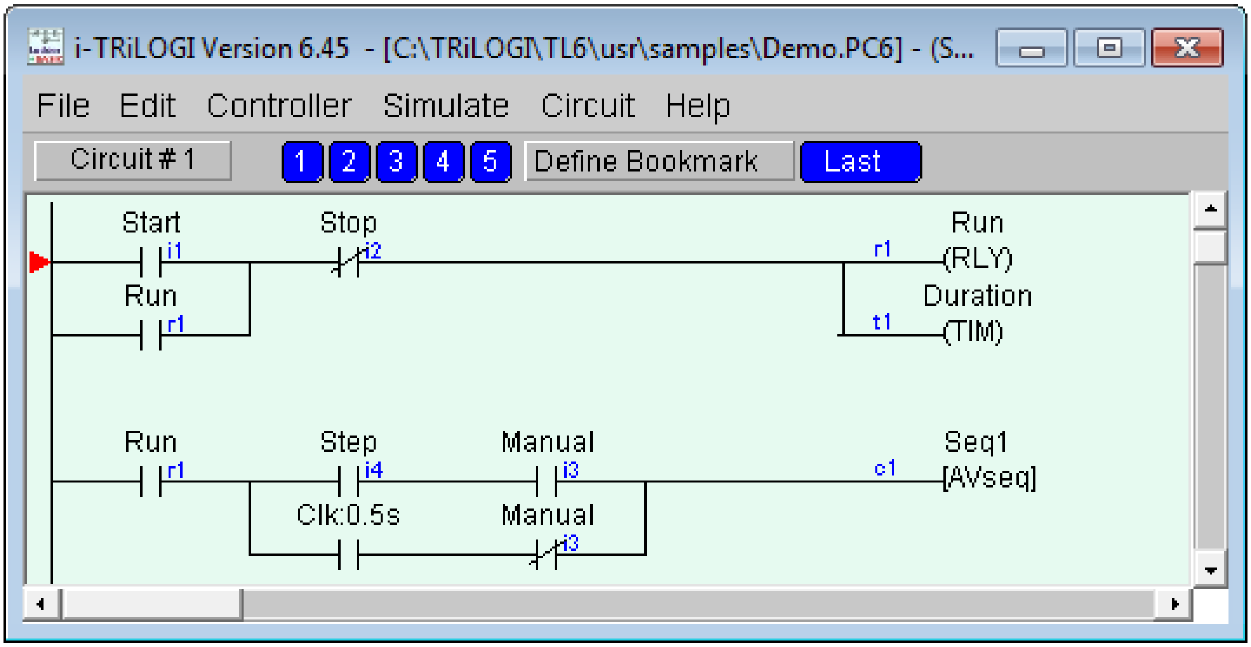
Pressing the key <0> to <9> and </> is equivalent to clicking on the icon shown in the table. The equivalent keyboard number is shown as a small numeral at the lower right corner of the icon. The </> key is the quickest way of converting a normally-open contact to a normally-closed one (and vice versa).

Pressing the <E> key when a contact or coil is selected allows you to edit the label name directly. Note that it is the user's responsibility to ensure that the label is valid.

Chapter 8 TRiLOGI Main Menu Reference

8 TRILOGI MAIN MENU REFERENCE

Both TRiLOGI application and applet programs have nearly identical look and feel (as shown below), with the exception that the applet can't save to or load from local drive.



The main body of the program window is for displaying and editing your ladder logic program. A ladder logic program is made up of many ladder "rungs". In TRiLOGI we call each ladder rung a "circuit" with an associated "circuit number". The currently selected circuit is marked by a little red triangle pointed to the circuit's intersection with the left vertical line (a.k.a. the "power rail" in ladder logic terminology)

The circuit number of the selected circuit is displayed on a button located just above the top left corner of the ladder editor window. If you happen to click on this button, a dialog box will popup that prompts you to enter the circuit number you wish to go to and the editor will bring you there immediately. The five blue buttons 1 2 3 4 5 next to the Circuit # button are called "Bookmarks" and their usage has been covered in details in Chapter 6.

8.1 File Menu

The File menu provides commands for the opening/saving of i-TRiLOGI files either on the local harddisk or on the TLServer's storage space.

8.1.1 New <Ctrl+N>

Activate this command when you want to create a new ladder logic program. All current ladder circuits and custom functions will be cleared from the screen and the default filename is "Untitled.pc6".

8.1.2 Save <Ctrl+S>

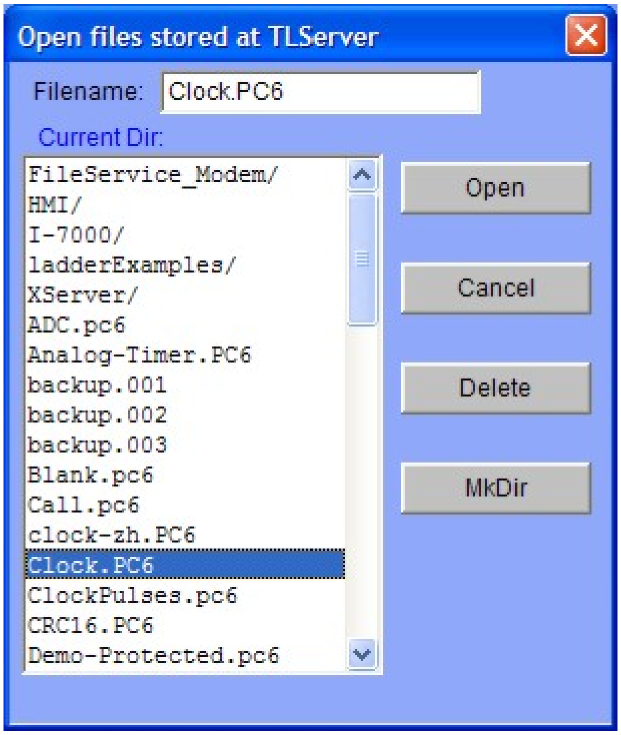
This command saves the whole ladder logic program, all I/O tables and all the custom functions to the disk. The current file will be saved to the same source from which it was opened from, i.e. If a file has been previously opened from the TLServer via the network, this command will save the file back to the TLServer. Likewise, a file opened from the local harddisk will be saved automatically to the local harddisk.

8.1.3 Open (TLServer 3) - <Ctrl+O>

This command is for loading a i-TRiLOGI program file "xxx.PC6" from the TLServer version 3.x or later. Due to the need to use Unicode in the network stream to handle Unicode-based ".PC6" file, this function on TL6 can only work with TLServer 3.x and above. Since TLServer 2.x does not support Unicode network stream this function will not work with TLServer 2.x. (If you attempts to execute "Open (TLServer 3)" and put in the IP address of a running TLServer 2.x or earlier, TL6 will complain that the TLServer is not running or unsuccessful connection to TLServer.)

When selected, you will be prompted to enter the Username and the Password to gain access to the TLServer. (the same Username and Password must have already been defined in TLServer for this to work). If you are running i-TRiLOGI as a local application instead of as an applet, you may be required to enter the "IP Address: port" of TLServer in order to connect to TLServer. (Note that last entry of IP Address: port is saved to the TL6 configuration file and will be loaded when the TL6 application re-starts).

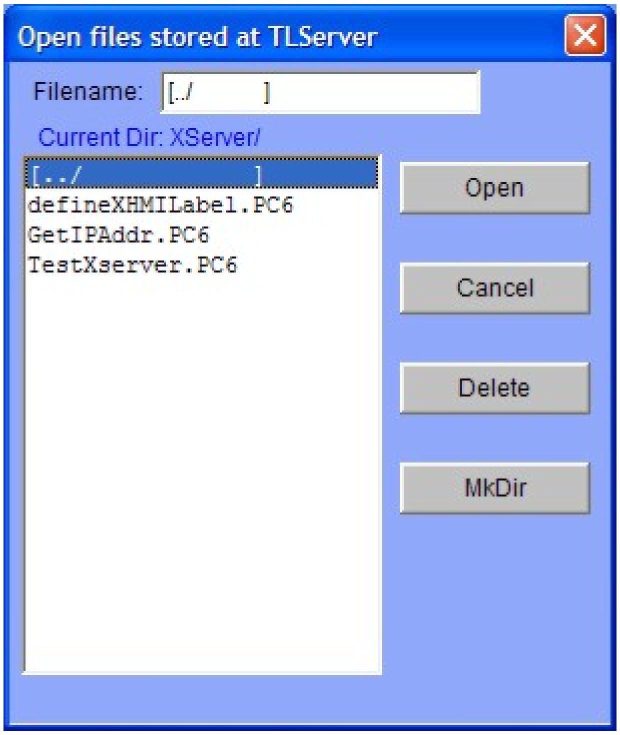
Each user has his own exclusive directory for storing his i-TRiLOGI files. Once authenticated, a network file dialog will be opened for you to select a file, delete a file or create a subdirectory, as shown below:


Open files stored at TLServer

Filename: Clock.PC6

Current Dir:

FileService_Modem/
HMI/
I-7000/
ladderExamples/
XServer/
ADC.pc6
Analog-Timer.PC6
backup.001
backup.002
backup.003
Blank.pc6
Call.pc6
clock-zh.PC6
Clock.PC6
ClockPulses.pc6
CRC16.PC6
Demo-Protected.pc6


Open files stored at TLServer

Filename: [../]

Current Dir: XServer/

[../]
defineXHMILabel.PC6
GetIPAddr.PC6
TestXserver.PC6

Simply double-click on the desired file or select the file you wish to open and click the "Open" button to open the i-TRiLOGI file.

Sub-directory: The MKDir button allows you to create a sub-directory on the server to organize your files. Subdirectory names always end with a "/" character. If you open a subdirectory its contents will be displayed in the file window. To return to the parent directory from a sub-directory, you simply double-click on the [../] symbol.

8.1.4 Save As (TLServer 3)

Use this command if you wish to save the currently edited i-TRiLOGI file to the TLServer 3.x using a different filename. You will be prompted to enter the Username/Password (and the IP address if it is a TL6 application) to gain access to TLServer. Once authenticated, the network file dialog similar to that described in "Open (TLServer 3)" will be opened for you to enter a file name or select a filename to overwrite. The default file extension for file saving is ".PC6" which will save the file in Unicode format to the TLServer user folder. However, you may also save the file with an extension ".PC5", in which case the TLServer 3.x will then save the i-TRiLOGI file in ASCII format. Files saved with ".PC5" extension may be opened by older TL5 applications.

Important Notes: Any non-ASCII characters used in the program file will be lost when the file is saved in ".PC5" format and it is not recoverable. So if you use non-English I/O labels your program will most likely fail to compile when it is saved as a ".PC5" file, since the I/O labels are converted to ASCII and are partially trancated. Thus, it is important to keep the original copy of your .PC6 program so that you can work on it to resolve the unicode conversion issues.

8.1.5 Open (Local Drive) / Save As (Local Drive)

For TL6 Application (not Applet) you can open or save a file from/to the local harddisk. You will be presented with the typical file dialog provided by your O/S. This command however is not available to the TL6 Applet since an applet does not have the right to access local hard disk resources.

The default file extension for opening a i-TRiLOGI file is ".PC6". This signals to the TL6 application that the file is in Unicode format and TL6 will then load it accordingly. However, TL6 is also able to read ".PC5" files created by older TL5 applications. TL6 assumes that all ".PC5" files are stored in ASCII format and will load it accordingly. To display ".PC5" files in the file dialog, simply enter the string "*.PC5" in the "File Name:" text field and press <Enter>. The window will now display only files with "*.PC5" extension. You can then navigate to the folder that contains the ".PC5" files to pick the file you want to open.

The default file extension for saving a i-TRiLOGI file in TL6 is ".PC6". This signals to the TL6 application to save the data in Unicode format. Unicode is required to store international characters which cannot be properly saved in the ASCII format used by TL5. However, you may still save the currently opened i-TRiLOGI file in TL6 to a file with an extension ".PC5", which will then instructs TL6 application to save the file in ASCII format. Files saved with ".PC5" extension may then be opened using older TL5 applications.

<u>Import Notes:</u>

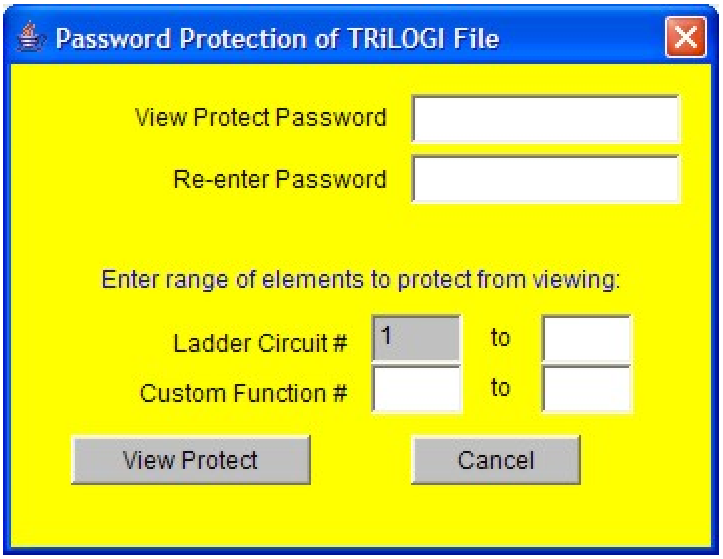
1. Since ".PC6" files are saved in Unicode while ".PC5" files are saved in ASCII code format, the two files type are NOT interchangeable. Therefore, you must not simply change a file name extension from ".PC5" to ".PC6" via Windows File manager and then attempt to open it from TL6. Doing so will cause TL6 to complain that it is not a valid i-TRiLOGI file.

2. Any Unicode characters used in the program that cannot be represent by 8-bit ASCII code will be lost when the file is saved in ".PC5" format and it is not recoverable. So if you use non-English I/O labels your program will most likely fail to compile when it is saved as a ".PC5" file, since the I/O labels will be converted to ASCII and partially truncated. Thus, it is important to keep the

original copy of your .PC6 program so that you can work on it to resolve the Unicode conversion issues.

8.1.6 View Protect / Un-Protect

This feature allows you to prevent others from viewing a pre-defined range of ladder logic and custom functions. When you select "View Protect" command, you will be asked to enter an unlocking password, a range of ladder logic starting from circuit #1, as well as a range of custom functions whose content you wish to prevent others from viewing, as follow:



The moment you click on the "View Protect" button, you can no longer view or print the protected range of ladder circuits and custom functions. When you save the view-protected program, it will be saved in an encrypted format and it cannot be opened using older version of i-TRiLOGI software. However, you will still be able to add new ladder circuits and custom functions to this program as well as modify those unprotected ladder circuits and custom functions. Of course, you will still be able to compile, simulate and transfer the protected program to the PLC as usual.

A view-protected file can be unprotected by selecting the "Un-protect" command from the "File" menu and supplying the unlocking password. Note that this unlocking password is strictly for un-locking the viewing restriction and it has nothing to do with other username and/or password required for interacting with the PLC.

This View Protection command is extremely useful for OEMs who wish to allow end users of their equipment to modify or append to the PLC's program for ease of interfacing to other equipment, but without revealing the core content of the PLC program to the end user. Besides being able to protect the OEM's intellectual property, it will also help to prevent the end users from mistakenly modifying the core program which can lead to disastrous result.

8.1.7 Write Compiled Code to Disk

This new feature is added since i-TRiLOGI version 5.32. You can now write the compiled program code to a disk file so that you can send the compiled code (with a ".CO5" extension) to your end customer to upload to the PLC using a standalone "TL5 Uploader" program. This allows you to protect your source program file content, while giving the end users of your equipment the ability to upgrade the PLC program. The end users DO NOT need to install the i-TRiLOGI or the Java JRE in order to use the TL5 Uploader program. so that makes it more flexible for you to distribute self-upgrade for the end users.

The TL5 Uploader program can be obtained by emailing to "sales@tri-plc.com" and installed by first unzipping the "Install TL5 Uploader.zip" file found in your i-TRiLOGI Version 6.x CD-ROM or and then run the "Setup.exe" program. Please note that the "TL5 Uploader" is a copyrighted program and Triangle Research International is the copyright owner of this program. However, Triangle Research International authorize the licensed users of the i-TRiLOGI version 5.xx software to freely distribute the "Install TL5 Uploader.zip" program to their end users at no charge.

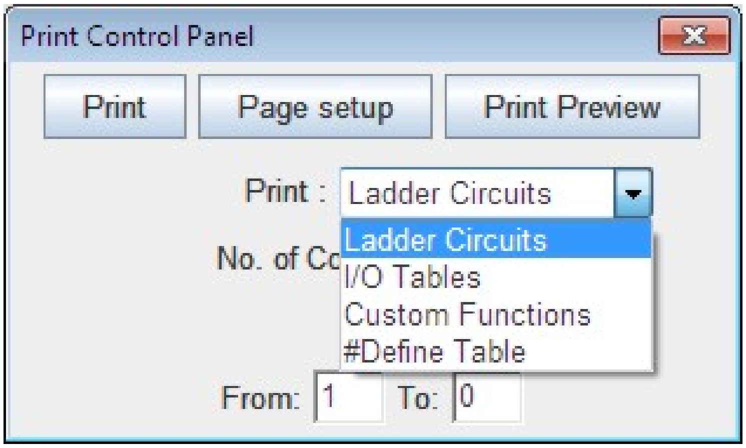
Note:

1. Although i-TRiLOGI version 6.x supports Unicode for its user-interface, the compiled code produced by TL6 is identical to that of TL5 and therefore the compiled file is still saved in ASCII format. To maintain backward compatibility the "Write Compiled Code to Disk" function still produces ".CO5" file that can be uploaded using the TL5 Uploader" program.

2. Beginning from i-TRiLOGI version 6.41, you can also use the i-TRiLOGI software to transfer the .CO5 file to the PLC. This allows licensed i-TRiLOGI users to exchange compiled .CO5 file without revealing the source code.

8.1.8 Print

You may use all the printing resources supported by your O/S to print a selectable range of the ladder diagram, the I/O Tables or the custom functions. When executed the following "Print Control Panel" will appear:



To print, first select the item from the choice box and define the range you wish to print and then click on the "Print" button. For "Ladder Circuits", the range indicates the circuit numbers. For "I/O Tables", the range indicates the I/O number (up to 256) and for "Custom Functions", the range is the function number.

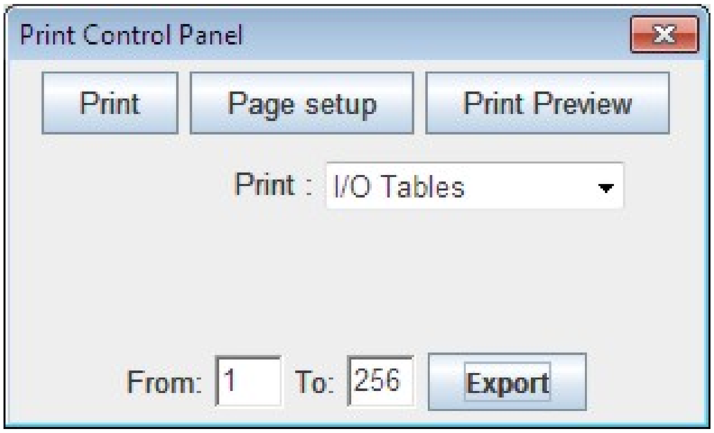
You can use the "Print preview" button to check the pagination of the printing on screen. You can select paper size and print orientation. etc. by clicking the "Page setup" button. Empty custom functions will be automatically skipped to save paper. When you select to print the "Ladder Circuits" a special "No. of Element" textbox appears. This textbox is for you to enter the maximum number of series element that can be printed on the paper width. Changing this number affects the scaling of the ladder diagram when printed. The smallest number is 5 and largest number is 13. Use a smaller number if you wish to have a larger printout. However, please note that if your ladder program contains circuits with more elements than that indicated by the "No. of Element" parameter, the "out-of-page" part of those ladder circuits will not be printed.

Wider I/O Table

Previously, the I/O Table label names could only be 10 characters, but i-Trilogi 6.2 and up allows label names of up to 16 characters (see the IOTable section for more details). It is now possible to print an I/O Table with label names up to 16 characters and still fit all of the information within the page width. The only difference is that Relays #257 to 512 will be printed on a second page to provide more room.

Export the I/O Table

Now you can export the I/O Table to a ".csv" file that can be opened by any program that has ".csv" compatibility. If you use Microsoft Excel, you can open the ".csv" file and the I/O table contents will automatically be displayed in an Excel spreadsheet. This provides limitless possibilites for printing the contents of the I/O Table and for integrating them into other documents. To export the I/O Table, just choose the "I/O Tables" option from the Print Control Panel and click on the Export button, as shown in the Print Control Panel below. You will then be prompted to save the I/O Table as a ".csv" file.



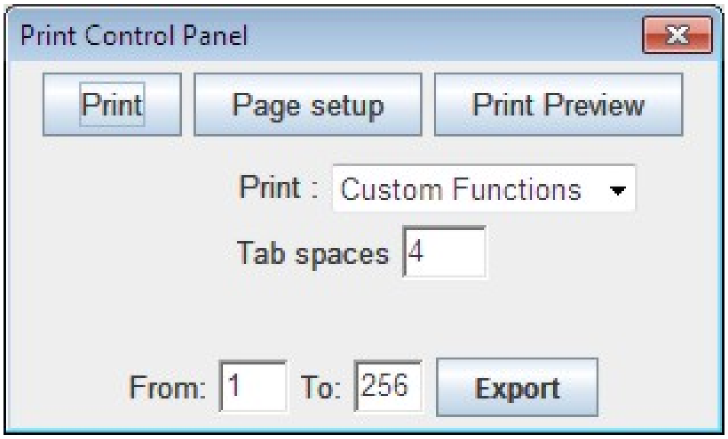
Print : I/O Tables

From: 1 To: 256

NOTE: You can select the range of I/Os you want to export to a ".csv" file, just like you can select the range of I/Os you want to print.

Export Custom Functions

Now you can export custom functions to a ".txt" file that can be opened any text editing program. It is best to use a text editor that interprets carriage returns as new-line characters as well so that each custom function is displayed as it would be in the i-Trilogi Custom Function Editor instead of as one long line that is very difficult to read. A good editor to use is Wordpad or Microsoft Word. To export Custom Functions, just choose the "Custom Functions" option from the Print Control Panel and click on the Export button, as shown in the Print Control Panel below. You will then be prompted to save the Custom Functions as a ".txt" file.



Print : Custom Functions
Tab spaces 4
From: 1 To: 256

NOTE: You can select the range of Custom Functions you want to export to a "txt" file, just like you can select the range of Functions you want to print.

Export #Define Table

The new #Define Table feature introduced since version 6.43 can be exported to a ".txt" file that can be opened any text editing program or MS Excel program. To export Custom Functions, just choose the "#Define " option from the Print Control Panel and click on the Export button.

8.1.9 Exit

Execute this command to exit orderly from the i-TRiLOGI program. You will be prompted to save the current file if the contents have been edited and the changes have not yet been saved.

8.2 Edit Menu

8.2.1 Abort Edit Circuit

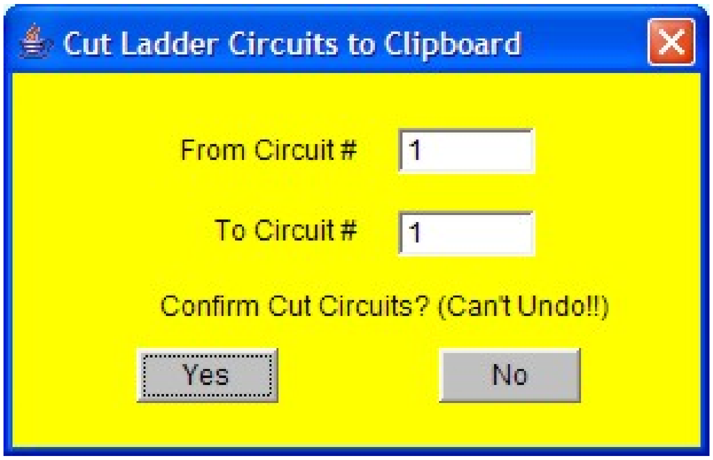
Changes made to the current ladder circuit can be aborted if you execute this command before pressing <Enter> to accept changes made to the current circuit. If changes have already been accepted by pressing the <Enter> key, then this command will have no effect. This command is useful if you wish to completely abandon changes you have made to a circuit without going through all the undo steps.

8.2.2 Undo <Ctrl+Z>

Undo the last changes made to a ladder circuit. i-TRiLOGI automatically stores the last 10 edited steps so you could execute undo several times to restore the circuit back to its original shape.

8.2.3 Cut Circuit - <Ctrl+X>

You can remove a number of circuits from the current ladder program and store them temporarily in the clipboard for pasting into another part of this ladder program or into another file altogther. In other words, it lets you move a block of circuits from one part of the ladder program to another part or into another file. Once you execute the "Cut Circuit" command, a prompt box as shown below will appear. You have to specify the range of the circuits you wish to cut and press the "Yes" button to remove them from the ladder program.



From Circuit # 1

To Circuit # 1

Confirm Cut Circuits? (Can't Undo!!)

Please note that you can't UNDO a Cut Circuit operation.

8.2.4 Copy Circuit (Ctrl+C)

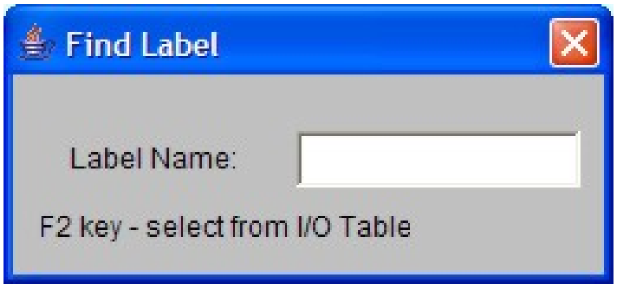
You can copy a block of circuits from the current ladder program and store them into the clipboard for pasting into another part of this ladder program or into another ladder program file altogther. The range dialog box similar to "Cut Circuit" will appear for you to enter the range of circuit to copy.

8.2.5 Paste Circuit <Ctrl+V>

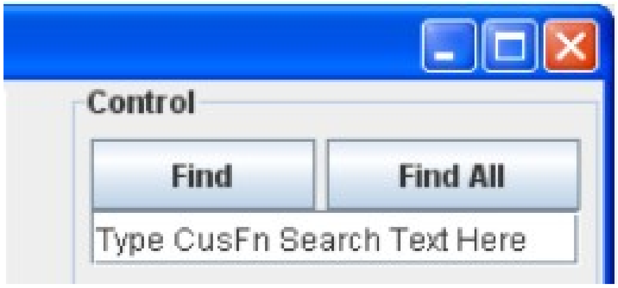
When you execute this command, the block of ladder circuit which you "Cut" or "Copy" into the clipboard will be pasted just before the currently selected circuit. The current circuit number will be adjusted to reflect the change.

8.2.6 Find <Ctrl+F>

The Find command allows you to quickly locate a ladder logic circuit that contains a particular label name. This is useful for searching for the activity of a particular I/O in the program. The Find command can also be used to search for a keyword in a TBASIC program. When this command is executed you will be further prompted to select the options of either searching for a ladder logic label or finding a text in a Custom Function.



F2 key - select from I/O Table

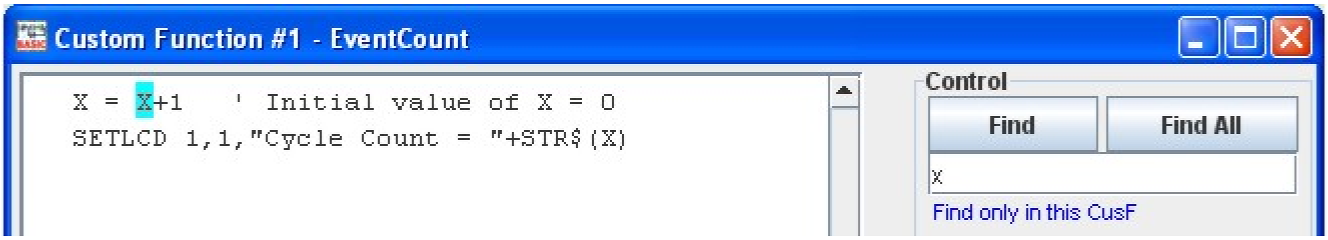


Type CusFn Search Text Here

Find Ladder element: You can enter into the the text field a string that partially or fully matches the label name you wish to locate. You can also press the <F2> key to open up the I/O table and pick the label name from the I/O table.

Find Text in CusFn: It is now possible to search for text within a custom function and when the option "Find Text in CusFn" is selected, the custom function editor window will open automatically to the first custom function. You can either search for text within the current custom function (local search) or in all of the custom functions in your program (global search).

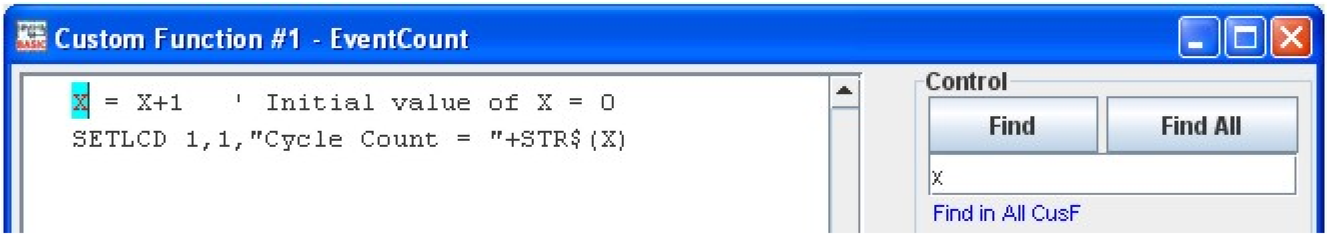
To do a local search, simply type the text in the command line below the "Find" and "Find All" buttons and then click the "Find" button. If the text is found in the current custom function, it will be highlighted in the text editor as shown below. Also, the text "Find only in this CusF" will be displayed below the command line in the search area, indicating a local search. Each time the "Find" button is clicked, the next instance of the search text will be highlighted until the text cant be found anymore. At this point the message in the search area will change to "Text Not Found" and the next time "Find" is clicked, the first result will be highlighted again.



```
X = X+1    ' Initial value of X = 0
SETLCD 1,1,"Cycle Count = "+STR$(X)
```

x

To do a global search, simply type the text in the same command line and click the "Find All" button. If the text is found in any custom function within the program, it will be highlighted in the text editor as shown below. Also, the text "Find in all CusF" will be displayed below the command line in the search area, indicating a global search. Each time the "Find All" button is clicked, the next instance of the search text will be highlighted until the text cant be found anymore (in any custom function). At this point the first result will be highlighted again. If no text matches the search text, the message in the search area will change to "Text Not Found".



```
X = X+1    ' Initial value of X = 0
SETLCD 1,1,"Cycle Count = "+STR$(X)
```

x

NOTE: Highlighting of text found in the editor during a user initiated search or during compilation error tracking now works with JRE (Java Runtime Environment) 1.5 and 1.6

8.2.7 Goto <Ctrl+G>

Use this command to move towards a specific circuit number. The "Goto" command is particularly useful if your program contains many circuits, and it is inconvenient to search for a particular circuit using the mouse or the cursor keys.

8.2.8 I/O Table <F2>

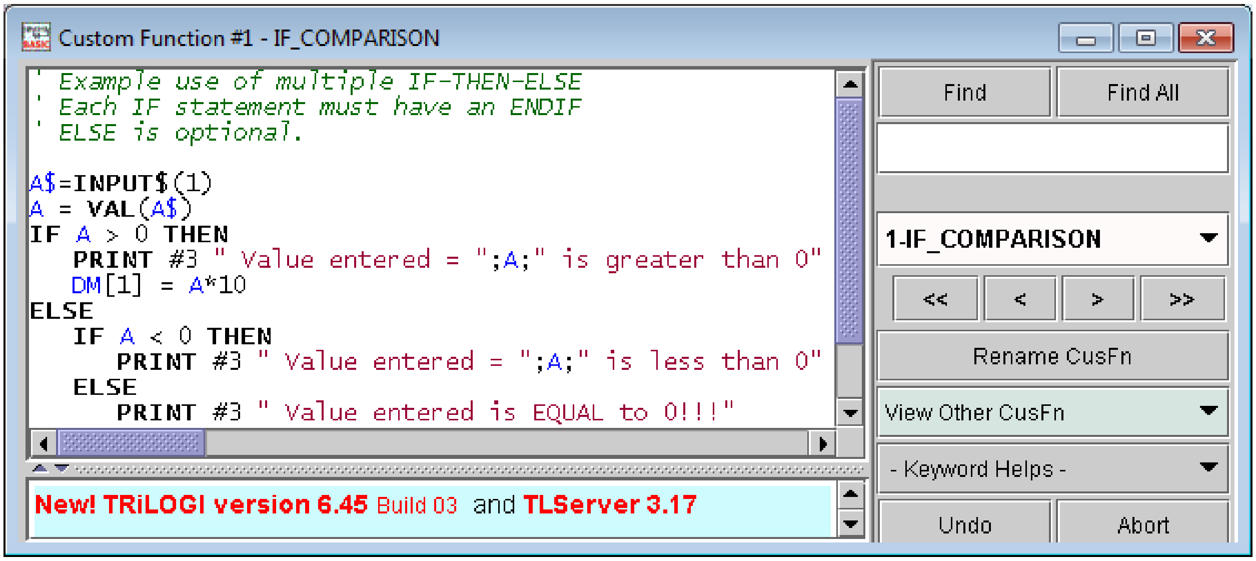
Open up the I/O Table for defining label names for the PLC's I/O. For detailed explanation of I/O tables, please click on the following link: I/O Definition Table

8.2.9 View I/O Type on Ladder <F3>

Toggle between display or no display of the I/O type and physical I/O number for ladder logic contacts on the screen. All ladder logic contact symbols are normally identified by their label names. However, you can also choose to display an optional small literal to indicate the I/O types and physical I/O number, which is now linked to the I/O labelname. e.g. i1=input 1, o12=output 12, r25= relay 25, t1= timer 1 and c5=counter 5. When i-TRiLOGI first starts, the display is enabled but you have the option of turning it off if you find it distracting.

8.2.10 Edit Custom Function <F7>

Opens up the Custom Function Editor window for you to enter the TBASIC program. You will be required to select the custom function number or a label name from the CusFn table (which is part of the I/O Table) . Each i-TRiLOGI file can contain a maximum of 256 custom functions. At any one time only one custom function will be opened for editing. The custom function number and the required label name will be displayed on the Title of the Custom Function editor window:



Custom Function #1 - IF_COMPARISON

```
' Example use of multiple IF-THEN-ELSE
' Each IF statement must have an ENDIF
' ELSE is optional.

A$=INPUT$(1)
A = VAL(A$)
IF A > 0 THEN
   PRINT #3 " Value entered = ";A;" is greater than 0"
   DM[1] = A*10
ELSE
   IF A < 0 THEN
      PRINT #3 " Value entered = ";A;" is less than 0"
   ELSE
      PRINT #3 " Value entered is EQUAL to 0!!!"
```

New! TRiLOGI version 6.45 Build 03 and TLServer 3.17

NOTE: This is a new and improved custom function editor. For a full description of its features, see the TBasic Introduction page and go to the 2. Custom Function Editor section.

8.2.11 Clear Custom Functions

This command allows you to select a range of custom functions whose content you want completely cleared. You will be prompted to select the range of custom functions to erase. Note that this action is not undoable.

8.2.12 Set TAB Width

You can change the number of spaces the editor used to display the <TAB> character that you insert to format the code hierarchically.

8.3 Controller Menu

All commands in this menu are for communication with the PLCs via the TLServer or the Ethernet XServer. Hence the TLServer must be actively running and connected to the PLC(s) via its serial port before the commands here can be successfully executed.

Note that TLServer can be running on the same computer that i-TRiLOGI is running on (using localhost IP 127.0.0.1) , or on another computer in the same local area network, or anywhere in the world with an Internet connection. The experience is identical regardless of where the TLServer (and hence the PLC) is situated.

Note: In order to maintain compatibility with the "Ethernet XServer" device which is not Unicode aware, the TL6 Application still uses standard ASCII encoding when communicating with the PLCs. (This is unlike opening or saving i-TRiLOGI program files to the TLServer, that would require Unicode network stream as mentioned in the "File" menu help page). As a result, the TL6 application can work with TLServer version 3.x as well as older TLServer 1.x and 2.x. when it is only for the purpose of programming/controlling and monitoring the PLC and not to read/write i-TRiLOGI file.

If there is no existing connection made to the TLServer, then execution of any command in this menu will always bring up the password dialog for you to enter the Username/Password as well as the IP Address:port of the TLServer. You must be positively authenticated before you are able to log-in to the TLServer. See Log In to TLServer for detailed explanation of the Username/Password Dialog box. Once you have log-in to the TLServer, see explanation of each function below:

8.3.1 Select Controller <Ctrl-I>

The only editable field is the ID field. You have to enter the ID address in hexadecimal notation (00 to FF). This command allows you to select another PLC that is connected to the same TLServer but with a different ID for on-line monitoring or program transfer.

8.3.2 Connect/Disconnect to Server

Use this command to log-in to the TLServer only if you have no intention to perform other controller commands. You may find that you seldom need to use this command since running the On-Line Monitoring or Program Transfer commands will also let you log-in to the TLServer if you have not yet done so. However, once you are connected, this command changes into "Disconnect from Server" and this is the only way for you to log out of the currently connected TLServer so that you can use the Username/Password dialog box to log-in as a different user, or to log-in to another TLServer of a different IP Address/port number.

8.3.3 On-Line Monitoring <Ctrl+M>

Refer to the On-Line Monitoring section for details.

8.3.4 Program Transfer to PLC <Ctrl+T>

This command is only available if your log-in username is assigned the access level of a "Programmer". If you log-in as a "User" or "Visitor", this command is disabled from the Controller's menu. (It will be enabled again after you disconnect from the server)

You can use this command to transfer your i-TRILOGI ladder+TBASIC program into the PLC. You will be prompted to confirm your action to prevent accidentally affecting the target PLC. The ladder program must be compiled successfully before the program transfer process can take place. The progress of the transfer process will be clearly shown on the program transfer dialog box.

NOTE: When doing a program transfer, it is now possible to abort a transfer even after the transfer has started. However, the PLC will be left in PAUSE state if a program transfer has been aborted to prevent execution of an incomplete program.

NOTE: The Program Transfer window is now wider to accommodate file names with longer strings and/or in other languages.

IMPORTANT: (For Nano-10, FMD88-10, and FMD1616-10 PLCs) If your program is using any timers or counters, you must click '**Reset**' (or ‘Reboot’) when the program transfer is complete so that the set values are properly stored to FLASH memory in the PLC. If you click on 'Close' instead, the set values would not be retained and would be lost when PLC power is turned off or cycled unless the program executes a 'RESET' or 'SETSYSTEM 252,0' command first. This is necessary whether or not any FRAM-RTC module is installed.

Refer to the Program Transfer section for procedure details

8.3.5 Transfer .CO5 File To PLC

This new command is added from i-TRiLOGI version 6.41 onwards. It lets you transfer a file that has been saved previously to the hard disk using the "File -> Write Compiled Code To Disk" function under the "File" Menu.

Using this function you can send compiled code to other users to be transferred into the PLC using a licensed copy of i-TRiLOGI v6.41 and above, but without revealing your source code. To ensure that the file is not corrupted when you email to others, try to compress it using a Zip or Rar software.

8.3.6 Open Matching Source File

You can use this command to query the connected PLC for the filename of the last i-TRiLOGI program transferred to it and it will attempt to match it to a file stored in the log-in user's directory at TLServer or on the PC's hard disk. If the file is found, it will be opened. Otherwise it will report that the file is not found. Note that this command only opens the source file based on file name matching. It does not verify whether the file has been modified. It is the user's responsibility to ensure that the file stored in the server is the same one that has been compiled and transferred to the PLC.

If you have created a new file (i.e. the file name is "Untitled") and then attempt to perform on-line monitoring, this command will be called automatically to try to open a file that matches the PLC. The command is also invoked when you select a PLC with a different ID either from the "Controller" menu or from within the "Full-Screen Monitoring" window.

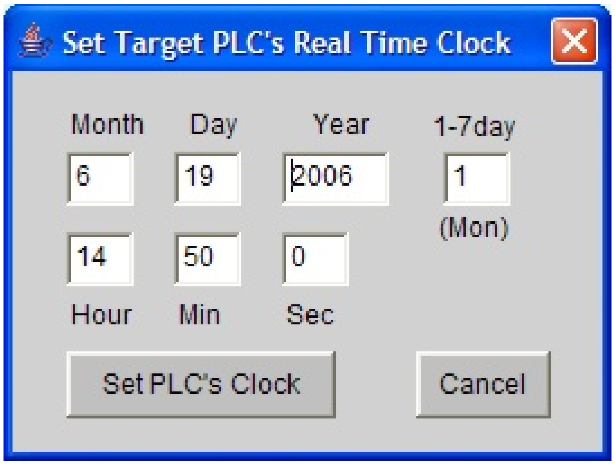
Note to Unicode users: If you created your file name using Unicode instead of plain ASCII, the Unicode filename will not be saved into the PLC since TRi PLCs currently do not support Unicode in its internal memory. In such case you cannot use this command to open the matching source file. You would have use the "File -> Open (Local Drive)" command to manually open the file for monitoring purpose.

8.3.7 Get PLC's Hardware Info

You can find out the PLC's firmware version number, the maximum of input, outputs, relays, timers and counters supported on this PLC as well as the total amount of program memory available. The same info will be displayed when you try to transfer a program to the PLC.

8.3.8 Set PLC's Real Time Clock

The PLC's real time clock (RTC, which includes both date and time) can be set quickly using this command. When you execute this command, a dialog box which contains the year, month, day, hour, min, sec and day of week are displayed for you to enter the value. The dialog box is initially filled with value taken from the client's computer's own calendar and clock. You can change any of the field to the desired values and then click on the "Set PLC's Clock" button:



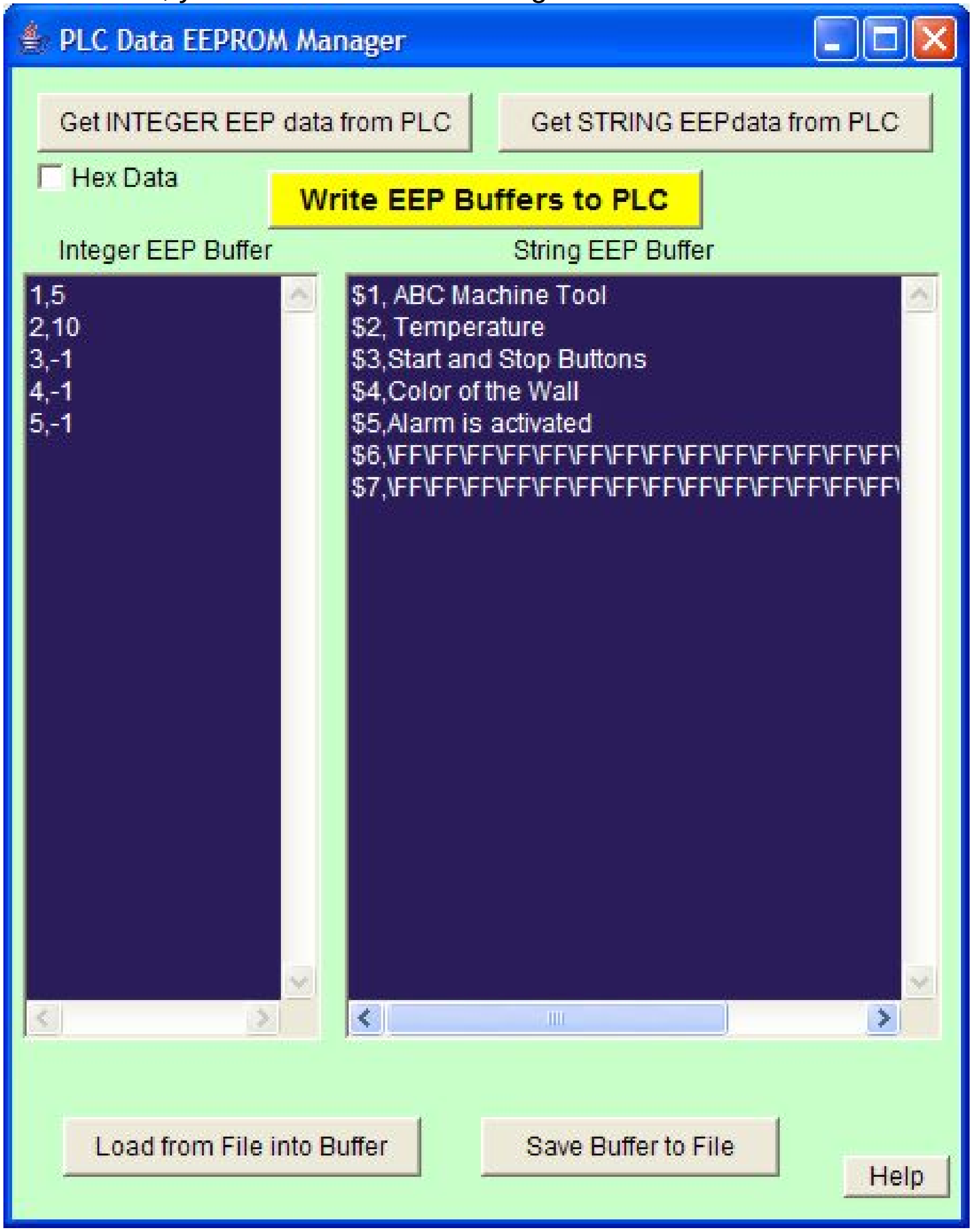
Month: 6
Day: 19
Year: 2006
1-7day: 1 (Mon)
Hour: 14
Min: 50
Sec: 0

The dialog box will be closed after the i-TRiLOGI has transferred all the data to the PLC. You should use on-line monitoring to verify that the data has indeed been properly written into the PLC.

Note that the "Year" field is restricted to only between 1996 and 2096, "Month" is between 1 and 12, "Day" is between 1 and 31, "Hour" is between 0 and 23, "Min" and "Sec" are between 0 and 59. If you enter an illegal value i-TRiLOGI will beep and the cursor will be put at the offending text field. Correct the mistake and then click on the "Set PLC's Clock" button again to transfer the values to the PLC.

8.3.9 EEPROM Manager

This command allows you to read/write the EEPROM data content from and to a Nano-10, FMD series or F-series PLC (as well as the legacy M-series PLC with firmware r47 and above). Note that the data memory type may not actually be EEPROM (could be Ferro-Magnetic RAM (FRAM) or FLASH), but the EEPROM manager will work the same way. The EEPROM manager also allows you to save or load the data to and from the PC's hard disk. When the "EEPROM Manager" is selected, you will see the following screen:



```
1,5
2,10
3,-1
4,-1
5,-1
```

```
$1, ABC Machine Tool
$2, Temperature
$3,Start and Stop Buttons
$4,Color of the Wall
$5,Alarm is activated
$6,\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\
$7,\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\FF\
```

There are two buffer areas: The left dark blue text area is a buffer to hold 16-bit Integer EEPROM data, the right text area is to hold the String EEPROM data. You can manually enter data in the buffer areas that are to be written to the data EEPROM, or if you retrieve data EEPROM (integer or string) from the PLC the data will be captured into these two buffer areas. The data must be entered in "comma-delimited" format as described below:

1. Integer EEP Buffer Format

Integer EEPROM data should be entered in the format [address], [data] and a line break. E.g. to store decimal data 12345 (equivalent to 3039 in hex) into address 10, enter the following text into the Integer EEP Buffer area.

10, 12345 or, in hexadecimal data format:
10, &H3039

2. String EEP Buffer Format

String EEPROM data should be entered as $[address], [text string]. E.g. to store the string "Test Message 10" into string address 5, enter the following text into the String EEP Buffer area:

$5,Test Message10
$8,Rm Temp=25\DFC

If you need to store a non printable character or ASCII character which is > 127 in value into the string EEP buffer, you can enter it as an "escape sequence" which is in the format of "\XX". The backslash character "\" denotes that this is an escape sequence, and XX is the hexadecimal value of the character. E.g. character A can be written as "\41" and character Z can be written as "\5A". In the second example above, the \DF character is ASCII 223 which on an LCD216 or LCD420 will be displayed as a 'degree' symbol.

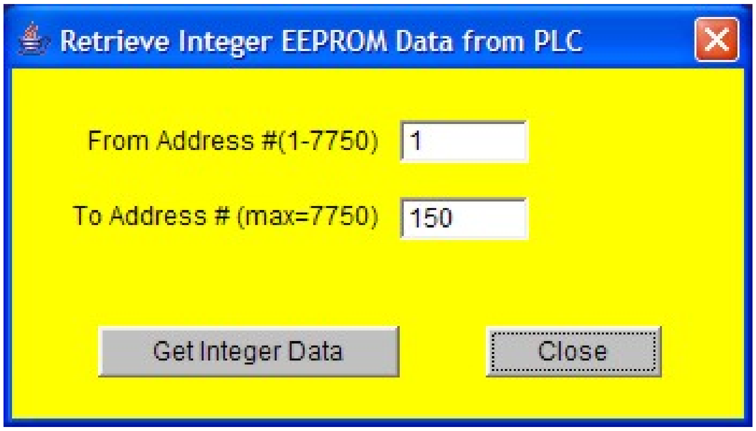
Note:

1. Normally when an EEPROM is in erased state, all it memory bits contain binary '1' and when you read them from the EEPROM as string, they will be returned as "\FF".

2. To prevent disrupting communication with the PLC, the strings to be written to EEP must not contain any of the following characters: ASCII 0 ("\00"), ASCII 10 ("\0A") and ASCII 13 ("\0D").

Click on this button to write both the integer and string EEP buffer contents to the PLC's EEPROM. i-TRiLOGI will prompt you to confirm the write action and it will also prompt you to login to a TLServer if it is not already connected to one.

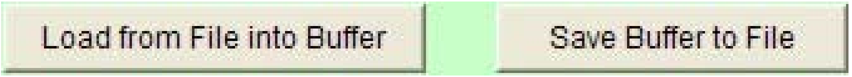
You can click on one of these two buttons to read a range of data EEPROM as 16-bit integer or as an ASCII string into their respective EEP Buffer. The program will attempt to login to a TLServer if it is not already connected. It will check the maximum EEPROM addresses available on the target PLC and then prompt you to enter the range of integer data you wish to read (up to the maximum available):


Retrieve Integer EEPROM Data from PLC

From Address #(1-7750) 1

To Address # (max=7750) 150

The data read from the PLC are stored into the EEP buffer in comma-delimited format as described above. Integer EEPROM data can also be captured in hexadecimal format if you check the ☐ Hex Data check box. You can edit or append to the captured data in the buffer and then write them back to the data EEPROM area or save as a hard disk file.



You can save both the Integer and String EEP Buffers content to a single text file with the default extension ".csv". The data are stored in comma-delimited format described above with Integer data being save first, then followed by the string data. The comma-delimited format makes it very easy for a spreadsheet program such as MS-Excel to import the data into the spreadsheet for further processing. You can also use MS-Excel to generate data that are to be written to the EEPROM and then save the file as "CSV" file, which can then be imported by EEPROM Manager using the "Load from File into Buffer" button.

EEPROM Manager Applicatons

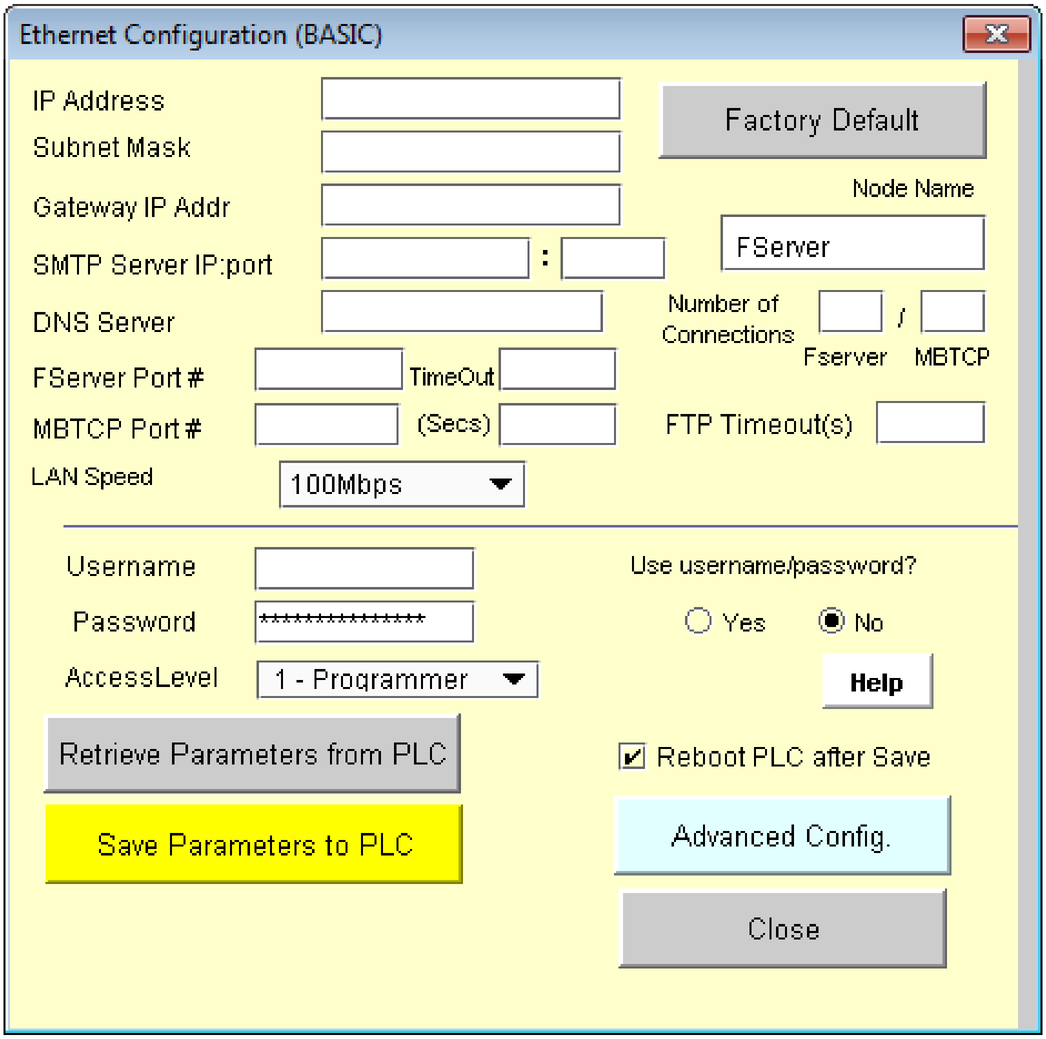
a) Using EEPROM Manager, it is now possible to pre-load the PLC's data EEPROM (either as 16-bit integer or ASCII strings) with pre-defined content so that they can be used by the PLC program. This is very convenient for implementing a lookup table or to define a large number of ASCII text string constants to be used for LCD display or communication without taking up too much program memory.

b) If you use the PLC's data EEPROM for data-logging, it is now possible to retrieved the saved data from the PLC and write it to the hard disk as a comma-delimited text file, which can then be imported into a spreadsheet or database program for further actions.

c) If you ever need to replace a PLC that logs data to its data EEPROM area, you now can also use the EEPROM Manager to retrieve all the data EEPROM content from one PLC and transfer them to the replacement PLC so that the replacement PLC will work identically to the original PLC.

8.3.10 Ethernet & ADC Configuration

This tool, which is located in the "Controller" Menu, allows you to configure the Ethernet Port and ADC/RTC calibration settings on a TRi's PLC with built-in Ethernet port, such as the F-Series PLCs. When the “Ethernet & ADC Calibration” is selected, you will see the following screen:



Node Name: FServer

LAN Speed: 100Mbps

Password: ****************

AccessLevel: 1 - Programmer

Use username/password? ○ Yes ● No

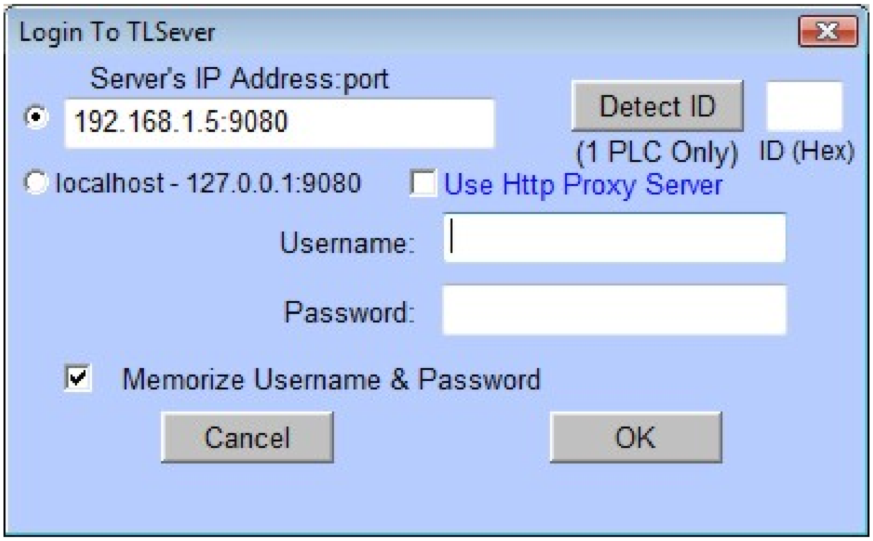
☑ Reboot PLC after Save

Note: Default settings shown here.

The configuration tool communicates with the PLC via the same user interface as all other communication functions under the "Controller" menu. This means that the configuration tool will be communicating with the PLC using TCP/IP protocol. Since only PLCs with built-in Ethernet can utilize this function, it means that most likely the PLC would be connected to a router or switch with a preconfigured static IP address. If your PC is also on the same network then you simply login to the PLC server using the static IP address of the PLC. Of course if the PLC's Ethernet port is already mapped to a public IP address then you can also remotely configure the PLC via the Internet.

However, if you do not know or have lost the IP address of the PLC, then it is still possible to configure the PLC's Ethernet port by connecting the serial ports on the PLC to the PC. However, in this case you will need to run the TLServer on your PC and configure the serial port to work with the PLC's serial port. The configuration tool will connect to the PLC indirectly via the TLServer software. Click here for more details about TLServer.

Before you make any changes to the PLC's Ethernet configuration parameters, it is a good idea to retrieve the current settings. You can do this by clicking on the “Retrieve Parameters from PLC” button and answer "Yes" when prompted. If your i-TRiLOGI is not yet connected to a TLServer software or directly to the PLC built-in hostlink command server (known as F-Server in the F-series PLCs), then you will see the login popup windows as follow:



Login To TLSever

Server's IP Address:port

● 192.168.1.5:9080

○ localhost - 127.0.0.1:9080

☐ Use Http Proxy Server

Username:

Password:

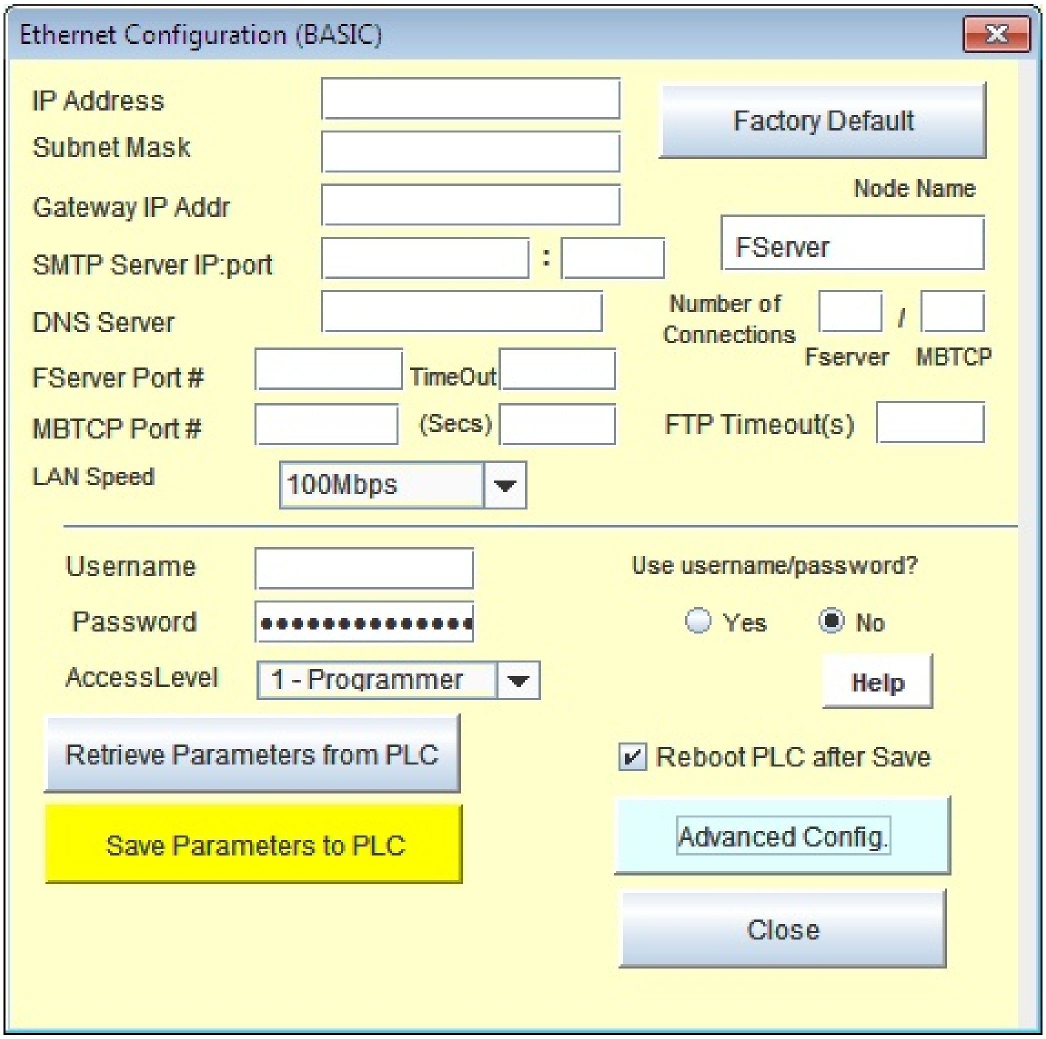
☑ Memorize Username & Password

If you are connecting to the PLC via TLServer that runs on the same PC as i-TRiLOGI, then simply select the "localhost-127.0.0.1:9080" to connect to TLServer.

If you are connecting to the PLC's host link command server, the IP address of the PLCs' and the port number should be entered in the "Server's IP Address:Port" field and select the corresponding radio button. The default settings for the Ethernet port are IP: 192.168.1.5 and port: 9080 with no username and password required (same as shown in the screenshot above). Once you have entered the correct login information, click on the "Detect ID" button and wait for the PLCs ID to populate in ID box (01 is the default ID). Then click "OK" and the configuraton tool will be populated with the current Ethernet settings.

After you have retrieved the existing parameters from the PLC, you will see that various fields in the configuration software screen are filled up. These fields and the remaining unpopulated fields can be configured as described in the rest of this document.

8.3.10.1 BASIC CONFIGURATION



Node Name: FServer

LAN Speed: 100Mbps

Password: ●●●●●●●●●●●●●●

AccessLevel: 1 - Programmer

Use username/password? ○ Yes ● No

☑ Reboot PLC after Save

1. <u>IP Address</u>

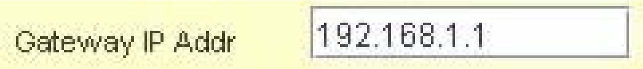
IP Address: 192.168.1.5

One of the most important parameters that you must define here is the “IP Address” field (as shown above). By default, every F-series PLC is shipped with the static IP address: “192.168.1.5”. You will need to assign the PLC’s with an IP address that is unique on your network and yet is accessible from your PC. If you are on a company network, then you must consult your company’s system administrator to assign you a useable IP address.

However, if your PC is connected to a small local area network, then most likely you will have a LAN IP address of 192.168.XXX.YYY. For proper networking, you should set the PLC's IP address to "192.168.XXX.ZZZ". i.e. The first three numbers should match each other. The fourth number (ZZZ) should be an IP address that is not used by any other devices on your network.

Note that majority of small LANs are built using a network router that assigns dynamic IP addresses (called DHCP server) to the PC. You should enter the administrator page of the router and define the range of DHCP for use by the PCs and then you may assign the PLC with any IP address that is outside of the DHCP range. E.g. If you define the DHCP address range to be 192.168.1.100 to 192.168.1.150, then you may assign the PLC with any IP address between 192.168.1.2 to 192.168.99 (Usually the router itself would have the IP address 192.168.1.1, and the address 192.168.1.255 is reserved so these two addresses are not available) and also between 192.168.151 to 192.168.1.254 (again making sure that 192.168.1.254 is not already used by your router).

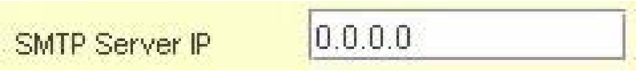
2. GateWay IP Addr



Gateway IP Addr 192.168.1.1

The Gateway IP address (as shown above) lets the F-series PLC communicate with other LAN segments or connect to the Internet. The gateway address is usually the local IP address of the router where the PLC is connected. For small local networks with no plan for connection to the Internet, the Gateway IP Address is not needed and can be set to 0.0.0.0. But if you plan to use the Fserver's email capability then you must fill in the correct Gateway IP Address. Ask your system administrator if you have any question about this.

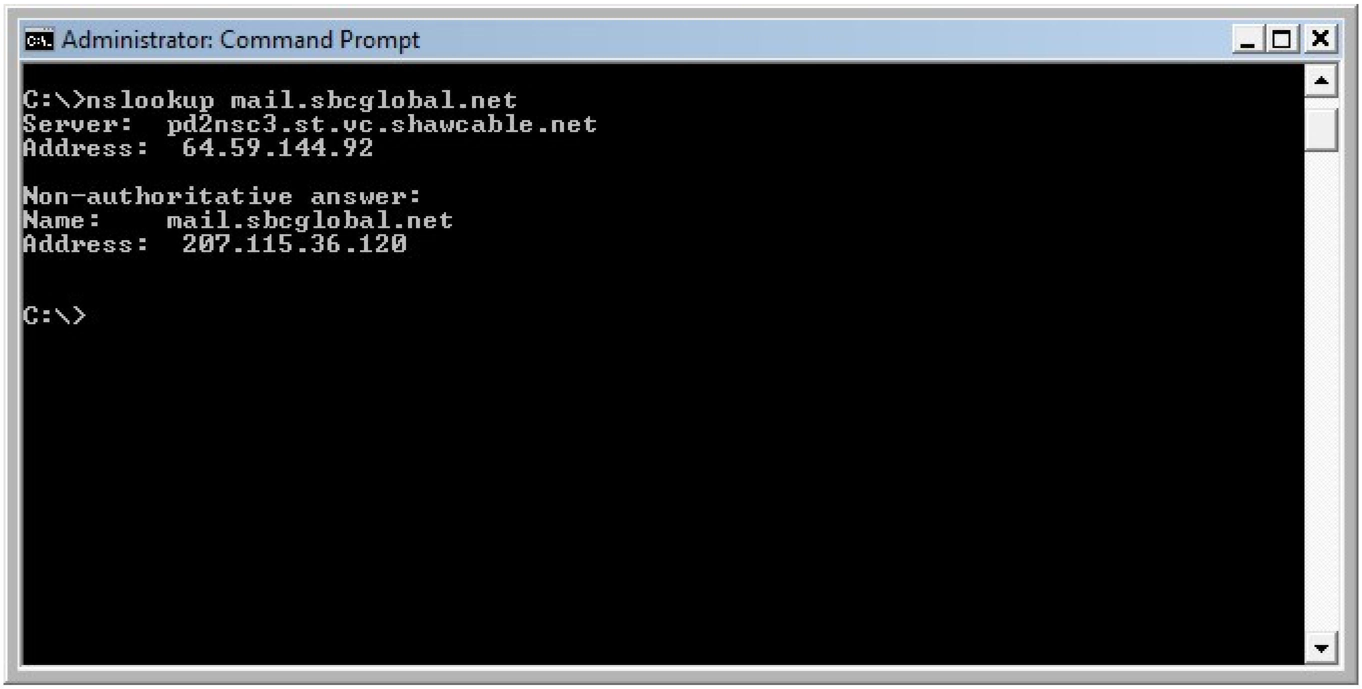
3. SMTP Server IP Address



SMTP Server IP 0.0.0.0

The SMTP (Simple Mail Transport Protocol) Server field (as shown above) lets you define the IP address of the email server that the PLC can use to send out emails from user's program (for F-series PLC users, please refer to Chapter 2.4.2 of the User Manual for more details on how to program the PLC to send emails). This is the same SMTP server that your normal email client software such as Thunderbird or MS Outlook uses to send out email. You can ask your Internet Service Provider (ISP) for the IP address of their SMTP server. The ISP usually provides the SMTP server in domain name form (such as "mail.sbcglobal.net"), but you should also be able to request the numerical IP address of the SMTP server from the ISP.

For Windows XP or Vista users, you can resolve the IP address as follows: First, launch the "Command Prompt" window. Then enter the command nslookup <smtpserver name>" to get the IP address. An example is shown below where the IP address of mail.sbcglobal.net is resolved to the IP address: "207.115.36.120":

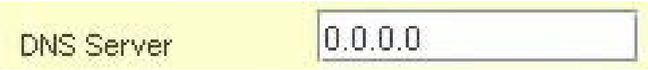


```
C:\>nslookup mail.sbcglobal.net
Server:  pd2nsc3.st.vc.shawcable.net
Address:  64.59.144.92

Non-authoritative answer:
Name:    mail.sbcglobal.net
Address:  207.115.36.120


C:\>
```

Windows users may also search the Internet for a free "host.exe" tool that lets you resolve the IP address from a given domain name (one host.exe tool that we found to work was downloaded from http://pigtail.net/LRP/dig/). For example, executing the command line: "host mail.sbcglobal.net" will resolve its IP address. (Of course you can only use this smtp server provided your ISP is SBC, almost no SMTP server will relay emails from a client that is not one of its own subscribers).

If you do not plan to use the FServer to send out emails yet, then you can leave the default SMTP Server IP Address = 0.0.0.0. You can change the settings anytime later when you need it.

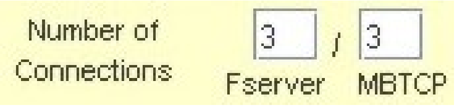
4. <u>DNS Server IP Address</u>



DNS Server: 0.0.0.0

DNS (Domain Name Server) allows the FServer to contact a remote location by means of domain name instead of IP Address. The DNS takes in the given domain name (such as yahoo.com) and returns the IP address of the target server. You only need to fill in the DNS IP Address if you are asking the FServer to contact a remote server by domain name, instead of by IP Address. However, currently this feature is not supported on the FServer and hence it may be left as the default value of 0.0.0.0.

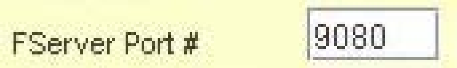
5. No. of Connections (FServer/ Modbus TCP)



Number of Connections 3 / 3
Fserver MBTCP

The F-series CPU assigns sufficient memory to support up to a maximum of 6 simultaneous TCP/IP connections to the FServer and Modbus/TCP server. By default, each server is assigned a maximum of 3 connections each. However, to improve flexibility, you can re-assign the mix of maximum connections between the two servers as long as the no. of Modbus/TCP connections does not exceed 5 and the no. of FServer connections does not exceed 4. This means that you can define 1 to 4 FServer connections and 2 to 5 Modbus/TCP connections. When you change the number in one box the other box will change automatically so that the total number of possible connections remains at 6.

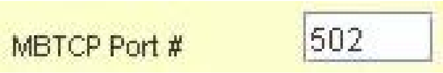
6. FServer Port No.



FServer Port # 9080

The Port number is a 16-bit integer (range 0 to 65535) that needs to be specified on top of the IP address when accessing the FServer from across the network. The default value is 9080, which is the same default value used by the TLServer and i-TRiLOGI client software. Please see the i-TRiLOGI programmer's manual for an explanation of the use of the port number. One reason why you may want to change the port number is to use the "port forwarding" capability of an NAT router so that different F-series PLCs may be accessible from the Internet using the same public IP address of the router but with different port numbers.

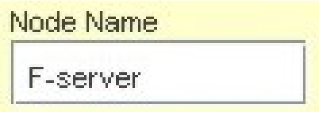
7. Modbus/TCP Secondary Port No.



MBTCP Port # 502

According to MODBUS.ORG specifications, all Modbus/TCP servers must listen on port #502. However, Modbus.org also permits the device to be assigned a different secondary port number. As such, the Modbus/TCP server will always listen on port #502 for all of its connections by default. Should you choose to define a secondary port number, then the Modbus/TCP server will only listen on port 502 on one connection while the additional connections (1 to a maximum of 4) would be listening on the secondary port. You may specify any port number between 1024 and 65535 (except for the port number already used by the FServer) to be the secondary port number. Please see the i-TRiLOGI programmer's manual for an explanation of the use of the port number. One reason why you may want to change the port number is to use the "port forwarding" capability of an NAT router so that different F-series PLCs may be accessible from the Internet using the same public IP address of the router but with different port numbers.

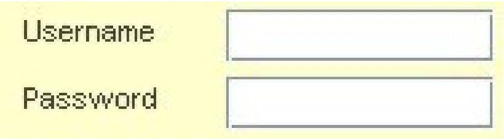
8. Node Name



Node Name
F-server

You can assign up to 16 ASCII characters (any character) in naming a PLC. The node name is currently not used by the network router so it is merely a convenient name for user to identify a PLC.

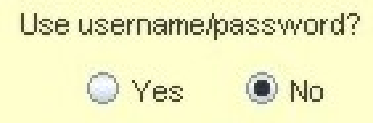
9. Username and Password (FServer only)



Username
Password

You can use the username and password feature to prevent unauthorized access to the FServer. It adopts the same proprietary encryption scheme used in the TLServer and i-TRiLOGI software to encrypt the password transmission. However, unlike the TLServer that allows you to define unlimited number of usernames and passwords, the FServer only permits a single username and password and this is limited to a length of 16 characters.

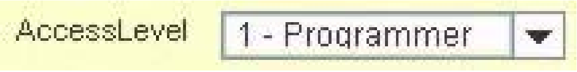
10. Use Username/Password (Yes/No)?



Use username/password?
○ Yes ● No

In applications where there is no danger of unauthorized access to the PLC via FServer, you can elect not to use the username/password. With the "No" option selected, the i-TRiLOGI client or Java Applet can log-in to the FServer using whatever username and password since FServer will bypass the username and password authentication and allow the client to log in.

11. Access Level



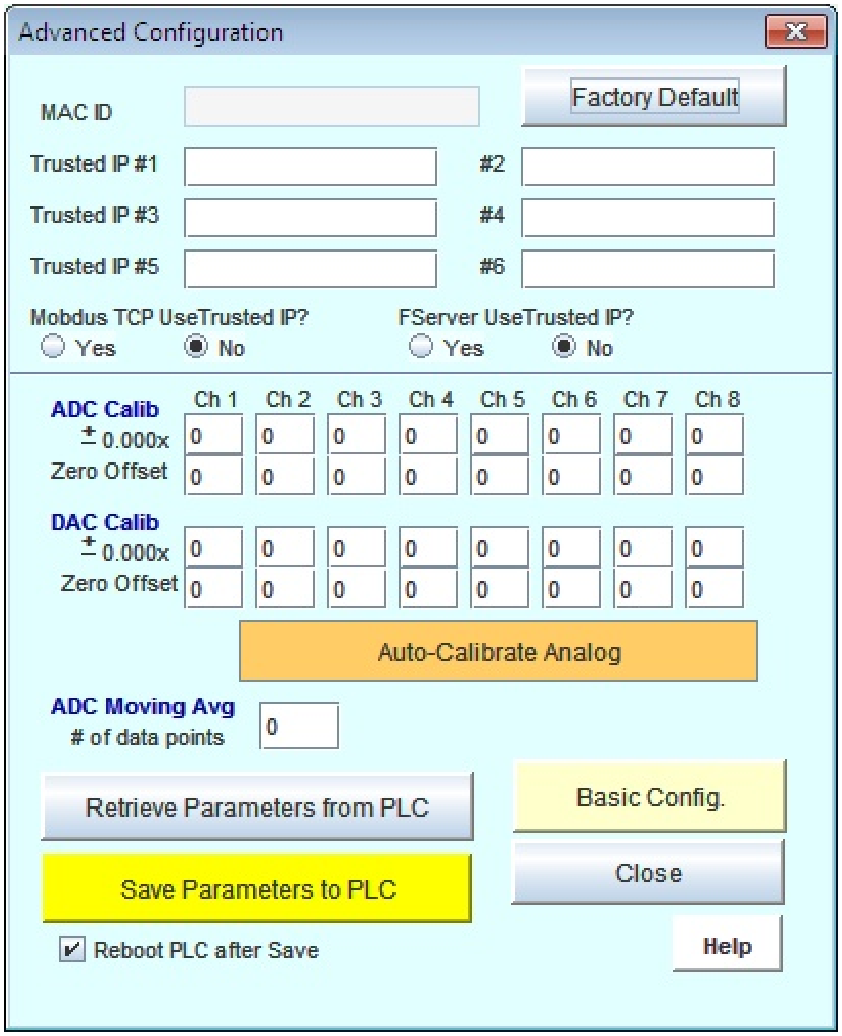
AccessLevel 1 - Programmer

You can define the access level that the i-TRiLOGI client is permitted to operate under on the PLC. Three access levels are currently defined: 1 for Programmer, 2 for User and 3 for Guest. Please see the TLServer User Setup for the definition of the access levels.

12. Advanced Configuration

The Advanced Configuration button lets you configure other more advanced (beyond the basic Ethernet configuration), but less often used features of the PLC. This includes definition of the “Trusted IP” addresses as well as calibrations of the PLCs Analog I/Os and Real Time Clock (RTC).

8.3.10.2 ADVANCED CONFIGURATION



MAC ID
Trusted IP #1 | #2
Trusted IP #3 | #4
Trusted IP #5 | #6

Mobdus TCP UseTrusted IP? ○ Yes ● No
FServer UseTrusted IP? ○ Yes ● No

	Ch 1	Ch 2	Ch 3	Ch 4	Ch 5	Ch 6	Ch 7	Ch 8
ADC Calib ± 0.000x	0	0	0	0	0	0	0	0
Zero Offset	0	0	0	0	0	0	0	0
DAC Calib ± 0.000x	0	0	0	0	0	0	0	0
Zero Offset	0	0	0	0	0	0	0	0

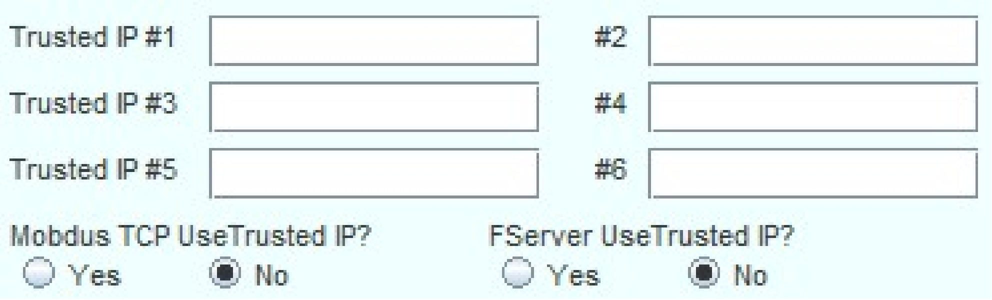
ADC Moving Avg # of data points: 0

☑ Reboot PLC after Save

1. <u>MAC ID</u>

A read-only field that contains the 6 byte (48-bit) Ethernet MAC ID.

2. <u>Trusted IP - MODBUS/TCP Access Security</u>



Trusted IP #1 | #2
Trusted IP #3 | #4
Trusted IP #5 | #6

Mobdus TCP UseTrusted IP? ○ Yes ● No
FServer UseTrusted IP? ○ Yes ● No

If an F-series PLC is to be accessible only on the local area network, then the direct connections offered by MODBUS/TCP provide simplicity without time-consuming login sequences. However, if the MODBUS/TCP port is to be exposed to the public Internet, then you ought to consider the security issues associated with MODBUS/TCP connections. Since a MODBUS/TCP connection does not require a username/password login sequence (unlike the FServer login), the only way to protect against unauthorized access is through the "Trusted IP" addresses defined using the F-Series Ethernet Configuration software.

The first thing you should do is to click on the "Retrieve Parameters from F-PLC" so that you can capture a copy of the current configuration in the PLC and you can then modify selectively. You can define a list of up to 6 "Trusted IP" addresses in this panel. To enable the Modbus/TCP Trusted IP, click on the "Yes" button next to the "Modbus/TCP Use Trusted IP". Note: The FServer can also be enabled to only allow connections from devices that match one of the "Trusted IP" defined in this panel. This is on top of the username/password login sequence that can be enabled/disabled from the Basic Configuration screen. In other words, you can choose either security method to access the FServer or implement both security methods at the same time.

After you have defined the list of trusted IP addresses and checked the "Use Trusted IP" radio button, click on the "Save Parameters to FServer" to save your data to the PLC's nonvolatile memory.

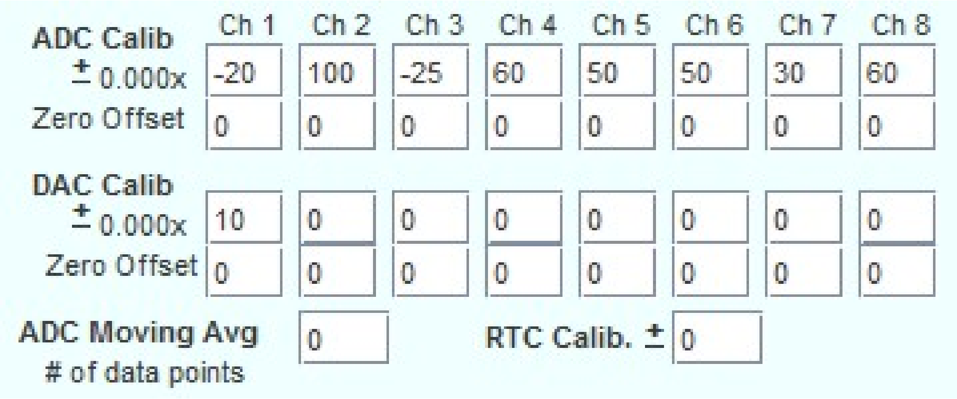
When "MODBUS/TCP Use Trusted IP" is enabled, it means that only TCP/IP packets that come from a client whose IP address matches one of the "Trusted IP" would be allowed connection to the MODBUS/TCP server.

3. Calibration of ADC and DAC & Moving Average Definition

The ADC and DAC are factory-calibrated such that a voltage of half of fullscale voltage (5.00V on F2424, and 10.00V on F1616-BA) should return a value of 2048 when read by the ADC(n) function. Likewise, an output voltage of half of fullscale voltage should be present at the DAC pin when you SETDAC to 2048. However, if there is a need to re-calibrate the ADC and DAC, you can follow the procedure outlined below.

To perform calibration of an ADC channel, you need to supply a precise DC reference voltage to the ADC channel, and then check the analog readings obtained via the ADC(n) function and compare it to the expected value. If there is an error, you can apply a correction factor to it.

The bottom half of the "Advanced Configuration" screen contains ADC and DAC calibration constants that you can enter and transfer to the PLC, as shown below:



	Ch 1	Ch 2	Ch 3	Ch 4	Ch 5	Ch 6	Ch 7	Ch 8
ADC Calib ± 0.000x	-20	100	-25	60	50	50	30	60
Zero Offset	0	0	0	0	0	0	0	0
DAC Calib ± 0.000x	10	0	0	0	0	0	0	0
Zero Offset	0	0	0	0	0	0	0	0

ADC Moving Avg # of data points: 0

RTC Calib. ±: 0

4. ADC Calib.

These fields are used to apply a multiplication factor to the value returned by ADC function. The multiplication factor = (1+ x/10000).

Example 1:

If you apply 2.500V to ADC #1 on a PLC with 5.000V full scale, you would expect the value returned by ADC(1) to be 2048. But the actual average reading centers around 2060.

Proportional Error = 2060/2048 = 1.005859

Multiplication factor required to correct this error = 1/1.005859 = 0.9942 = (1 – 58/10000) => x = -58

You should therefore enter a value of 58 into the “ADC Calib” field for Ch #1 and save it to the PLC. After the PLC has rebooted, the CPU would apply the multiplication factor of 0.9942 to the readings it received, which would correct the reading to: 2060 x 0.9942 = 2048.

Example 2:

If you apply 8.000V to ADC #8 on a PLC with 10.000V full scale, you would expect the ADC(8) function to return a value of 8.000/10.000 x 4096 = 3277. However, your program returned a value of 3230 from ADC(8).

Proportional Error = (3230)/3277 = 0.985658

Multiplication factor required to correct this error = 1/0.985658 = 1.0146 = (1+ 146/10000) => x = +146

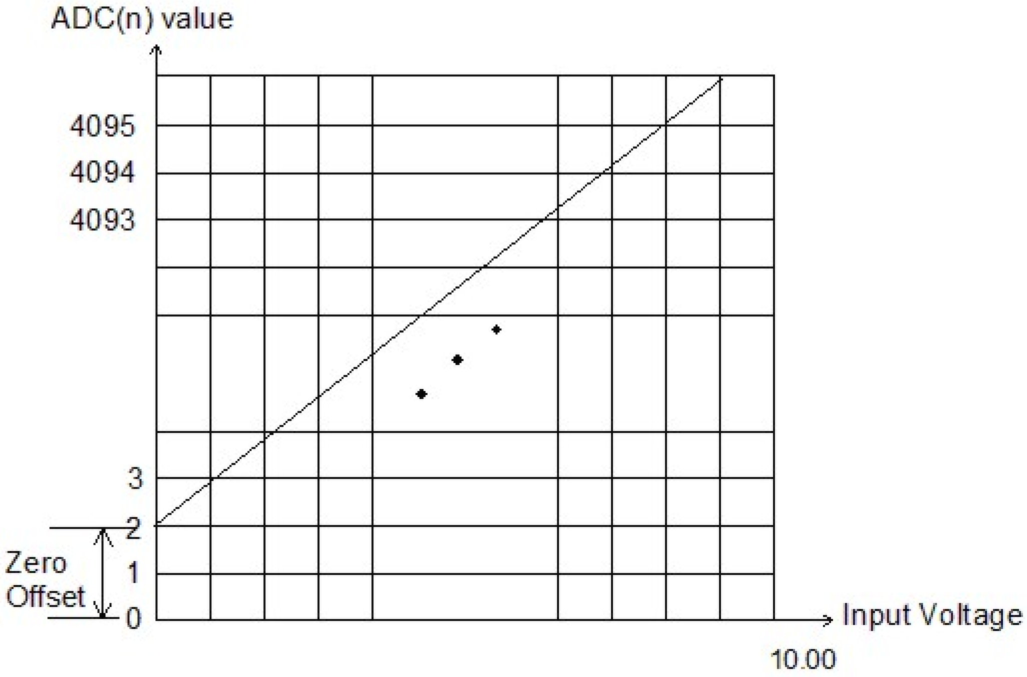
To compensate for this error, enter a value of 146 in the "ADC Calib." for Ch 8 and save it to the PLC. After the PLC has rebooted, the CPU would apply the multiplication factor of 1.0146 to the readings it received, which would correct the reading to: 3230 x 1.0146 = 3277.

Notes:

a) We have created an MS-Excel spreadsheet file "AnalogCalibration.xls" to facilitate the computations of the correction factor, X, used in the ADC and DAC Calibration. This file can be downloaded from the following web page: http://www.tri-plc.com/appnotes/F-series/AnalogCalibration.xls

b) Changes to the ADC calibration data only take effect after the PLC has been cold-booted. You can either power cycle the PLC or simply check the "Reboot PLC After Save" checkbox and the PLC will re-boot after you have transferred the parameters to it.

3. ADC Zero Offset



The zero offset error can be corrected by entering a value into the "ADC Zero Offset" field. Any number between –100 and 100 can be entered here. The ADC(n) function would add the zero-offset value that you entered here to the measured value and return the total sum to the calling routine.

4. DAC Calib.

A value x entered in each of these fields represents the multiplication factor (1 + x/10000) that the PLC will apply to the "value" parameter, which is executed by the command "SETDAC n, value" before actually writing to the DAC output. This allows the user to apply a correction factor to the DAC output if there is a problem with the DAC voltage.

Example 1:

If you execute "SETDAC 1,2048", you would expect the DAC #1 to output 5.000V on a 10.000V full scale DAC, but instead you only get 4.960V.

Proportional Error = 4.960/5.000 = 0.992

Multiplication factor required to correct the DAC output = 1/0.992 = 1.0081 = (1+81/10000)

You can therefore enter the value "81" into the "DAC Calib." field corresponding to DAC #1 and save the parameters to the PLC. After the PLC has rebooted, when you execute the statement "SETDAC 1,2048", the CPU would apply the multiplication factor of 1.0081 to the actual digital value it sends to the DAC output. The actual DAC output =4.960V x 1.0081 = 5.000V.

Example 2:

Same as Example 1 but you measure 5.030V when you expect 5.000V

Proportional Error = 5.030/5.000 = 1.006

Multiplication factor required to correct the DAC output = 1/1.006 = 0.9940 = (1 – 0.0060)

You should therefore enter the value –60 into the "DAC Calib" field corresponding to DAC #1 and save it to the PLC. After the PLC has rebooted, when you execute the statement "SETDAC 1,2048", the PLC will apply the multiplication factor of 0.9940 to 2048 before it writes to the DAC hardware.

The actual DAC output = 5.030V x 0.994 = 5.000V

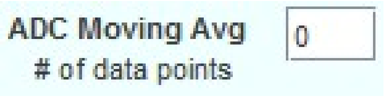
5. DAC Zero Offset

If you plot the line graph for the output voltage versus the DAC set value, the line should normally pass through the origin. But if there were any zero offset error, then the line would be above or below the origin.

The DAC output on the F-series PLC should not have any zero offset error and you normally should just leave these fields set to "0".

However, if for any reason there is a need to apply a zero offset error correction, you can enter a value between –100 to +100 into the "DAC Zero Offset" field. The CPU would add the zero-offset value you enter into this field to the "X" in the "SETDAC n, x" statement and only send the sum to the actual DAC hardware.

6. A/D Moving Avg

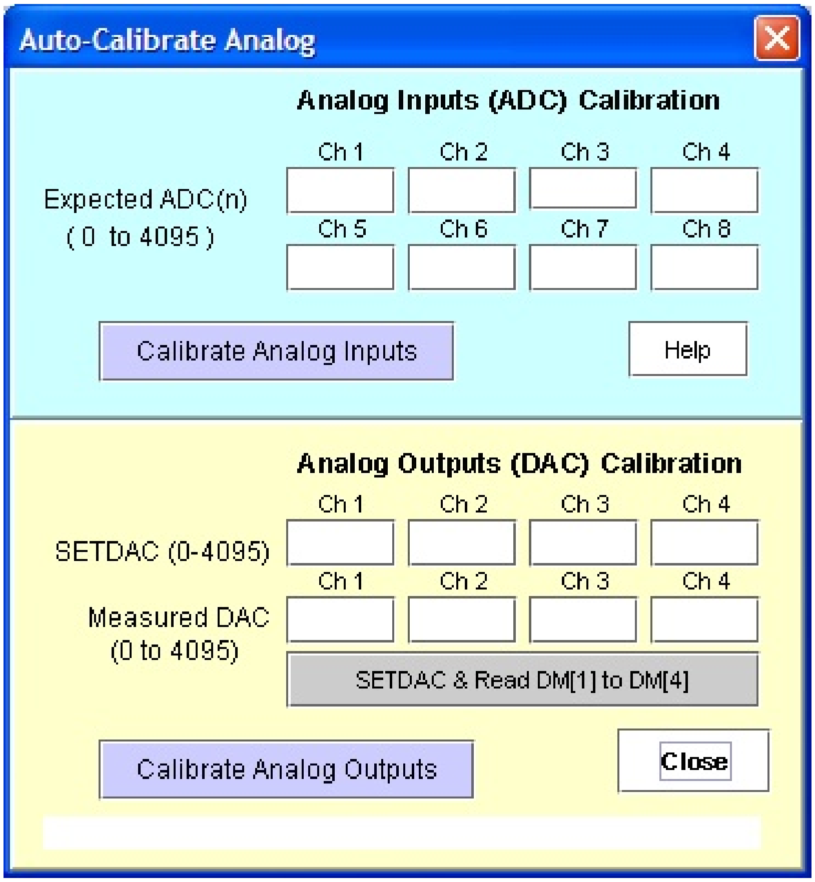


ADC Moving Avg
# of data points 0

This field lets you define the number of points of moving average that the F1616-BA CPU firmware uses to compute the value returned by the ADC(n) function (for F-series PLC users, please refer to Chapter 5.2.5 of the User Manual for an explanation of moving average).

A larger number of moving Average points has the positive effect of filtering out large noise spikes seen at the analog input, but the disadvantage is that the PLC would be slower in noticing a sudden step change at the analog input. If you specify a moving average of 1 point, that means no moving average will be used and the ADC(n) function will return the most recently sampled data at the analog input #n.

8.3.11 Auto Analog Calibration



Available since version 6.43, this new Analog calibration tool allows OEMs to perform quick calibration of the PLC's analog inputs and outputs.

This new tool is normally located under the "Controller" Menu. Alternatively, this tool may also be opened from the "Ethernet & ADC Configuration Tool -> Advanced Config." by clicking on the button labelled "Auto-Calibrate Analog".

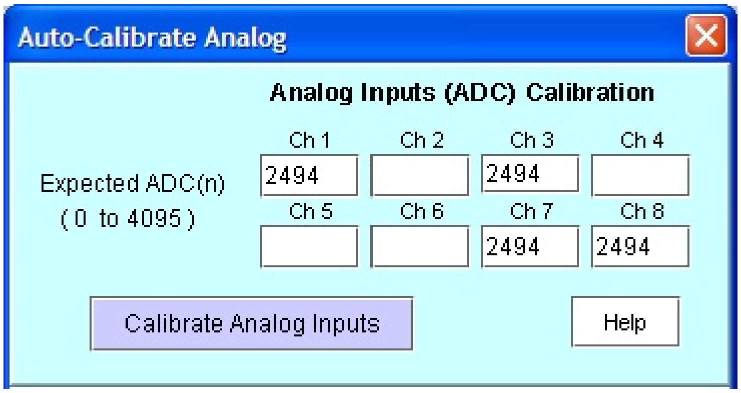
The Auto Calibrate Analog I/O screen is split into two parts. The upper portion is meant for calibration of the Analog inputs. The lower portion is for calibration of the PLC's analog outputs. We will explain each part separately in the next few sections as you may only need to calibrate what you need to use.

However, for those not too familiar with how to communicate with the PLC via i-TRiLOGI, go to section *4. Communicating with the PLC* below that describes the connection procedure.

1. Analog Inputs (ADC) Calibration

In order to calibrate the analog inputs, you need to have one or more stable and quiet reference voltage source(s) and a precision voltmeter. First, connect the voltage source to all the PLC's analog inputs that you wish to calibrate. For best precision, select a voltage source that falls somewhere in the middle of the range of analog values that you are concerned with. For example, if the temperature sensor that you normally measure with would return a voltage between 2.3V and 4.5V. Then it is best to calibrate using a voltage source around 3V. After connecting the voltage source to the analog inputs, use your precision voltmeter to measure the actual voltage and convert it to a reading that you expect the ADC(n) to return.

E.g. if your voltage source measures 3.045V and your analog input range is 0-5V, then the expected reading that ADC(n) should return corresponding to 3.045V should be = 3.045/5.000 x 4096 = 2494. Lets say If you are only calibrating analog input #1, #3 , #7 and #8 only and the rest are not used, then enter the value into the "Expected ADC(n)" fields for each channel as shown below. Leave any unused channel blank.



Expected ADC(n) (0 to 4095)

Ch 1	Ch 2	Ch 3	Ch 4
2494		2494	

Ch 5	Ch 6	Ch 7	Ch 8
		2494	2494

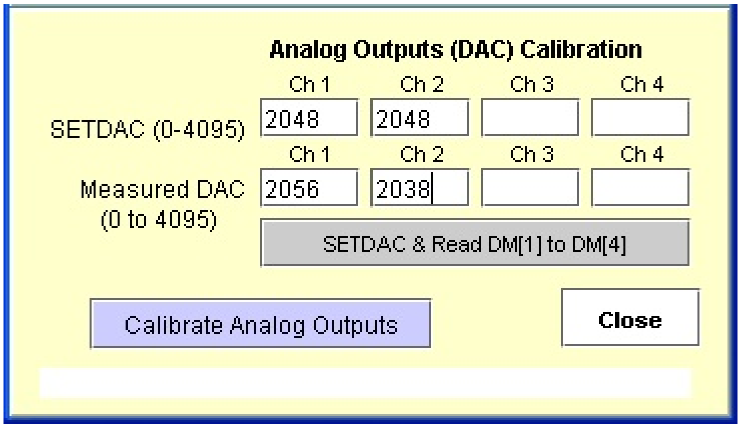
When you have finished entering the expected ADC data, then click on the button "Calibrate Analog Inputs" to start calibration.

The calibration program will first clear the old calibration data in the PLC and reboot it once. After that it will read 50 samples from the PLC's analog inputs and compare it against the expected ADC(n) value. The average readings of the 50 samples is then used to compute the calibration parameters and the program will store the calibration parameters into the PLC and reboot it again. The analog inputs are thus completed in a single step.

You may also use different voltage references to calibrate different analog channels. Simply leave the channel(s) that you do not want to calibrate with the currently selected voltage source BLANK. (Note: The calibration tool will NOT calibrate any channels whose field are left blank):

2. Analog Outputs (DAC) Calibration

In order to calibrate an analog output, you need to set the DAC to a certain value and then use a measuring instrument to measure the resulting output voltage (or current, if an AN20MA-2 interface board is used). The difference between the expected DAC output and measured DAC output is used to compute the calibration parameters for the PLC's analog output.



Analog Outputs (DAC) Calibration

	Ch 1	Ch 2	Ch 3	Ch 4
SETDAC (0-4095)	2048	2048		
Measured DAC (0 to 4095)	2056	2038		

For example, if you wish to calibrate analog output #1 and #2 and using an analog output value of 2048, you first must clear the old analog calibration paramters (using the Ethernet & ADC Configuration Tool -> Advanced Config), then write a program into your PLC to execute the following two statements:

```
SETDAC 1, 2048
SETDAC 2, 2048
```

Your PLC will then output an uncalibrated analog output voltage on the DAC which you can then measure with a precision voltmeter. If your analog outputs are configured for the 0 to 5V range, then ideally at 2048 you expect the DAC to output a voltage = 2048/4096*5.000 = 2.500V.

However, assuming that your precision voltmeter measures a voltage of 2.510V at DAC #1 and 2.488V at DAC #2, you will then need to convert the two measured readings to the range of 0-4096 and then enter into the "Measured DAC" fields. For DAC #1, the measured reading = 2.510/5 x 4096 = 2056 and for DAC #2, the measured reading = 2.488/5 x 4096 = 2038. Enter these two values into the table as shown in the above diagram.

After you have entered all the above parameters, you can then click on the "Calibrate Analog Outputs" button to start calibration of the analog outputs. The program will compute the calibration parameters based on the "SETDAC" and the "Measured DAC", write them into the PLC and then reboot it to complete the calibration.

3. Automating Analog Output Calibration

The procedure described in the last section may be a little bit tedious to work with since it involves several manual steps and require measuring the analog output readings and convert them into the 0-4096 range. It however illustrates the principle used by the calibration routine.

In order to automate the calibration of the analog outputs, an additional button SETDAC & Read DM[1] to DM[4] is included to assist with the task.

When you click on this button, i-TRiLOGI will perform the following tasks:

1) It clears all the old analog output calibration parameters for any analog output channel that has a non-blank entry into the SETDAC field.
2) It reboots the PLC so that the analog output from the PLC will be freed from any old calibration data.
3) It sets the corresponding analog output channel using the data entered into the "SETDAC" field.
4) It waits for one second and then read the DM[1] to DM[4] data from the PLC and store into the "Measured DAC" fields for DAC #1 to #4.

Therefore, if your PLC's analog inputs are already calibrated previously, you can make use of your calibrated Analog inputs to measure your uncalibrated analog outputs and store the measured readings into DM[1] to DM[4] to be used by your calibration program!

What you do is to connect the PLC's DAC #1 to its own ADC #1, DAC #2 to ADC #2 etc, and then write a PLC program to read the ADC data, take an average readings to even out white noise, then write the average readings into DM[1] to DM[4]. The SETDAC & Read DM[1] to DM[4] button can then retrieve the measured DAC readings from the DM[1] and DM[2] in a single step to perform the calibration.

Alternatively, if you have a precision measuring instrument that is capable of talking to your PLC (e.g. another PLC or an instrument that can communicate with the PLC via Modbus protocol), then the precision instrument can measure the DAC output of the PLC and write the measured readings into the DM[1] to DM[4] of the PLC, which can also be used by the SETDAC & Read DM[1] to DM[4] button to read the measured readings easily.

Once the reading has been obtained, just click on the Calibrate Analog Outputs button to complete the calibration.

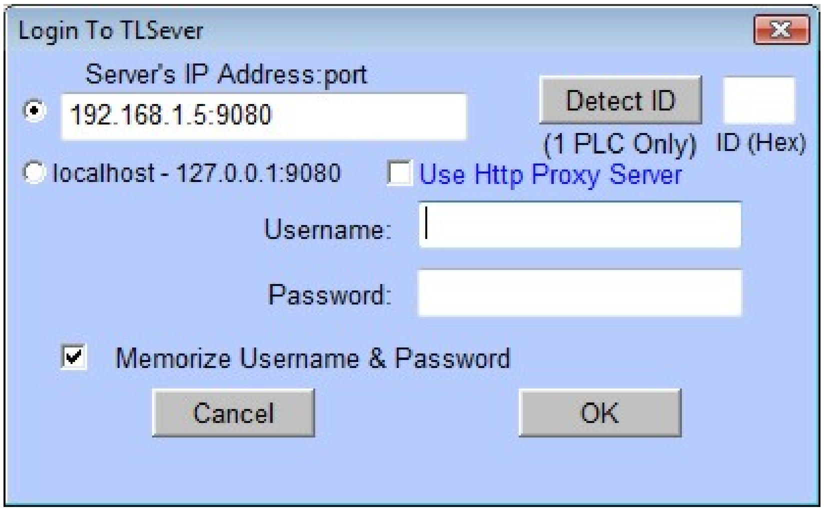
We have therefore reduced the analog output calibration to a simple two-step process which can be easily handled by any production operator during production of an OEM equipment.

7. Communicating with the PLC

This program communicates with the PLC via the same user interface as all other communication functions under the "Controller" menu. This means that the calibration tool will be communicating with the PLC using the TCP/IP protocol. If the PLC has a built-in Ethernet port then it should be connected to an Ethernet router or switch with a preconfigured private static IP address. If the PC is also on the same network then you simply login to the PLC server using the private IP address of the PLC.

However, if you do not know or have lost the IP address of the PLC, then it is still possible to calibrate the PLC's by connecting the serial ports on the PLC to the PC. In this case, you will need to run the TLServer on your PC and configure the PC's serial port to work with the PLC's serial port. The configuration tool will connect to the PLC indirectly via the TLServer software. Click here for more details about TLServer.

If your i-TRiLOGI is not yet connected to a TLServer software or directly to the PLC built-in hostlink command server (known as F-Server in the Nano, FMD or F-series PLCs), then when you try to calibrate the PLC you will see the login popup windows as follow:



● Server's IP Address:port: 192.168.1.5:9080
○ localhost - 127.0.0.1:9080
☐ Use Http Proxy Server
Username:
Password:
☑ Memorize Username & Password

If you are connecting to the PLC via TLServer that runs on the same PC as i-TRiLOGI, then simply select the "localhost-127.0.0.1:9080" to connect to TLServer.

If you are connecting to the PLC's F-server, the IP address of the PLCs' and the port number should be entered in the "Server's IP Address:Port" field and select the corresponding radio button. The default settings for the Ethernet port are IP: 192.168.1.5 and port: 9080 with no username and password required (same as shown in the screenshot above). Once you have entered the correct login information, click on the "Detect ID" button and wait for the PLCs ID to populate in ID box (01 is the default ID). Then click "OK" and you are now connected to the PLC for any calibration work.

8.4 Simulate Menu

i-TRiLOGI allows you to perform almost 100% simulation of your PLC's program off-line on your PC. This is a great tool for testing a program quickly before a machine has been manufactured. It is also a wonderful tool for all new PLC programmers to practice their ladder logic programming skill without the need to purchase a PLC test station.

i-TRiLOGI automatically compiles a ladder program before activating the simulator. If an error is found during compilation, the error will be highlighted where it occurs and the type of error is clearly reported so that you can make a quick correction.

8.4.1 Run (All I/O Reset) <Ctrl+F9>

This should be the option to use when you first begin to test your i-TRiLOGI program. When executed, all I/O bits (including inputs) are cleared to OFF state, all integer data are set to 0 and all string data are set to empty string. Then the "Programmable Logic Simulator" window will open for you to conduct the simulation test of your i-TRiLOGI ladder program.

8.4.2 Run (reset Except i/p) <Ctrl+F8>

Very often you may wish to keep the input settings "as is" when you reset the simulator. This situation is quite realistic in the sense that when a PLC is powered-on, some of its inputs may already be in the ON state. (e.g. sensors may detect the end positions of a cylinder rod, etc). This command allows you to preserve the logic states of all "Inputs" while resetting all other data. Note that the <Ctrl-F8> key also works in the "Simulator" screen so that at any time you can reset the simulator without affecting the logic states of the inputs.

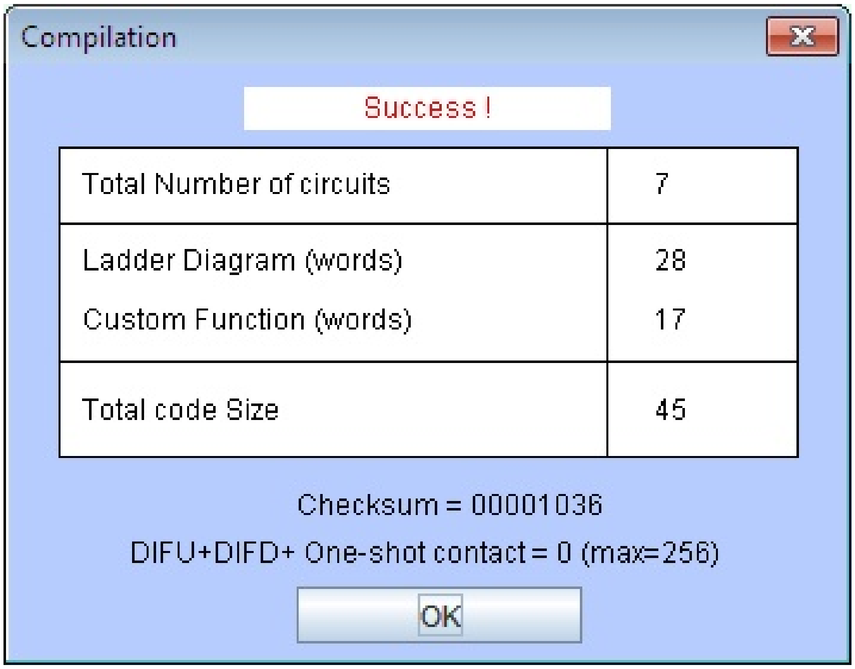
8.4.3 Continue Run (no reset) <F9>

Use this command to continue simulating the program since you last closed the simulator. All previous data are kept intact and will be used by the simulator to continue execution of the ladder program. If you have made changes to the ladder program or custom functions, the whole program will be recompiled before running.

Note that first scan pulse (1st.Scan) will not be activated when this command is executed since this is supposedly a continuation from the previous simulation run. This command can be useful if you have discovered a simple bug in your software during simulation, you can fix it immediately and test the effect of the change on the simulator instantly without restarting the entire simulation session from the beginning again.

8.4.4 Compile Only <F8>

Allows you to compile the i-TRiLOGI file only in order to view the compilation statistics:



Success !

Total Number of circuits	7
Ladder Diagram (words)	28
Custom Function (words)	17
Total code Size	45

Checksum = 00001036

DIFU+DIFD+ One-shot contact = 0 (max=256)

8.4.5 Reset All I/Os <Ctrl-R>

Clears all I/Os in the simulation engine without invoking the simulator. Since all I/Os whose logic states are turned ON in the simulator will also be shown as highlight on the ladder diagram, this offers a way to clear the I/Os if it hinders your viewing of the ladder program.

8.5 Circuit Menu

8.5.1 Insert Comments

Comments are specific remarks used by a programmer to explain various characteristics of a program segment and are ignored by the compiler. TL6 allows comments to be freely inserted between circuits. Execute this command and the Comment Editor will be opened. The comment editor allows you to enter any text you like that best describe the working of the circuit. All standard text editing keys, including cut and paste are applicable to the Comment Editor.

In early versions of i-Trilogi, you would press the <ESC> key or click the ☒ button to close the comment window and automatically save the changes when you had finished editing it. i-Trilogi 6.2 and higher allows you to save and close the comment the same way as before or by clicking the OK button. It is also possible to exit the comment and discard any changes made since it was last opened by clicking on the new Abort button.

Once a comment has been created, it is assigned a circuit number and is treated like any other circuits. You can edit it by pressing the <spacebar> when you are in Browse mode, alternatively, you can move it around, copy it to another destination or delete it entirely using commands in the "Circuit" menu.

Note: Since TL6 is Unicode-aware, you can now enter comments text in any international language and as long as the file is saved with a ".PC6" the comments will be properly saved into the data file and will be properly reloaded when the file is next open.

8.5.2 Insert Circuits

This command enables you to insert a new circuit just before the currently selected circuit. The current circuit number will be increased by one while the new circuit will assume the current circuit number. You will be placed in the circuit editing mode for immediate circuit creation.

8.5.3 Move Circuit

You can rearrange the order of the circuits by using this command. Select the circuit you wish to move and execute the "Move Circuit" command, then select a destination circuit location and press <Enter>. The selected circuit will be moved to the new location before the destination circuit.

Note that if you wish to move a block of circuits to a new location, you may find it more productive to use the "Cut Circuit" and "Paste Circuit" commands in the "Edit" menu.

8.5.4 Append Circuit

Execute this to add a new circuit to the ladder logic program. This new addition will be positioned immediately after the last circuit in the entire program.

8.5.5 Delete Circuit

This command allows you to delete the one or more circuits. You will be prompted to enter the range of circuits that you wish to delete. Please note that you can't UNDO a delete circuit operation.

8.6 Help Menu

All contents in this manual are available for instant reference on the computer where the TRiLOGI program is running. This includes running the “applet” version of TRiLOGI program on a remote browser because the relevant help files will be retrieved from the TLServer automatically.

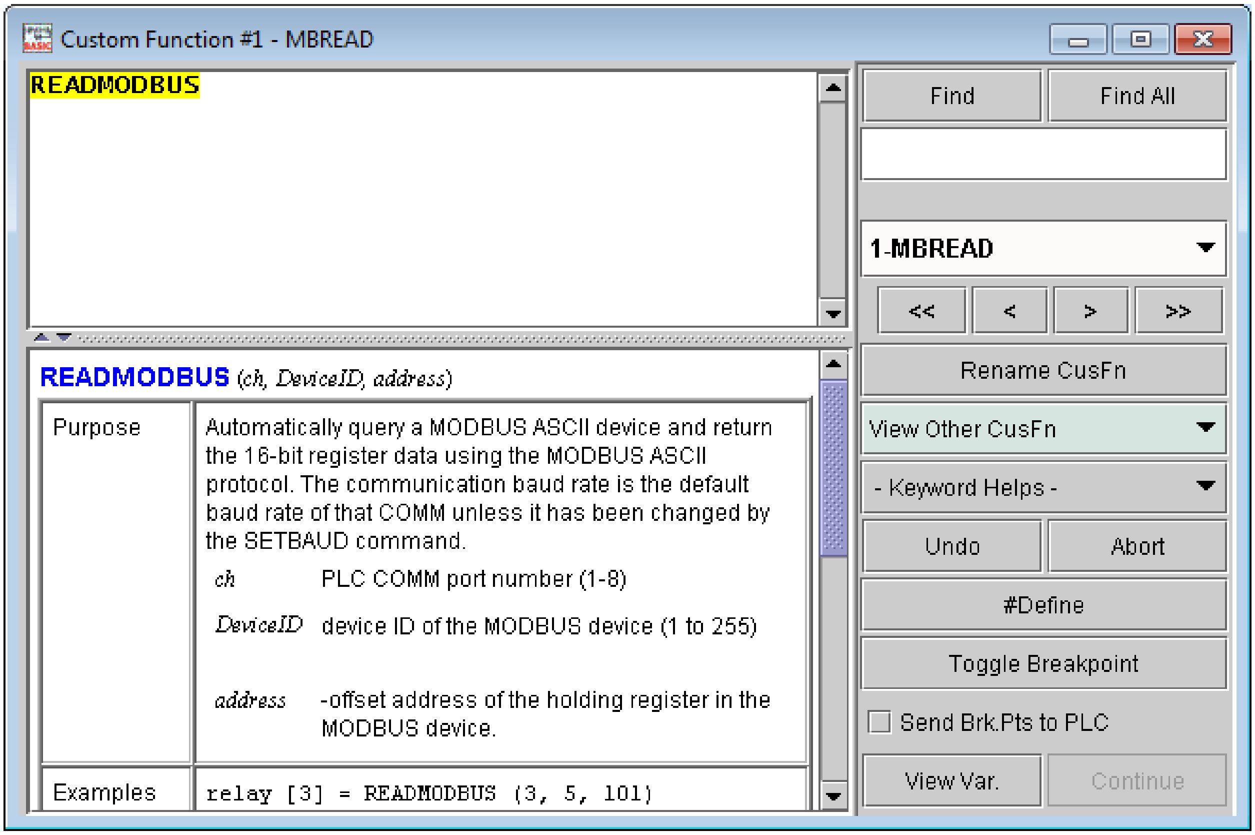
You can call up the help files anytime by pressing the <F1> key. You can also select the “Content” item from this “Help” menu to bring up the content page of the entire on-line TRiLOGI help files. On the content page you can find the links to the Ladder Logic Editor and the entire TBASIC language reference.

There is a “TRiLOGI Upgrade” link in the Help menu that allows you to obtain the latest Internet TRiLOGI upgrade setup program from the Internet. Simply follow the instruction to enter the website using the supplied username and password to obtain upgrade.

On TRiLOGI version 6.1, there is also a special command called “Chinese LCD Code Converter” on the Help menu. This is to help generate the escape sequences for special LCD model that can display Chinese characters. Its function will be covered in the Chinese LCD installation and user guide and is not needed by non-Chinese LCD users.

Instant Help for TBASIC Keywords

One convenient feature implemented in TRiLOGI Version 6.4 and higher is the ease of getting help for the syntax of a known TBASIC keyword. E.g., if you want to find the syntax for the keyword “READMODBUS”, instead of navigating through the help file links, you can simply select the “READMODBUS” keyword in the custom function editor and you will be immediately presented with the help content for the “READMODBUS” command as illustrated in the following screen shot.



```
READMODBUS
```

READMODBUS (*ch, DeviceID, address*)

Purpose	Automatically query a MODBUS ASCII device and return the 16-bit register data using the MODBUS ASCII protocol. The communication baud rate is the default baud rate of that COMM unless it has been changed by the SETBAUD command. *ch* PLC COMM port number (1-8) *DeviceID* device ID of the MODBUS device (1 to 255) *address* -offset address of the holding register in the MODBUS device.
Examples	`relay [3] = READMODBUS (3, 5, 101)`

Chapter 9 Ladder Logic Language Reference

9 LADDER LOGIC LANGUAGE REFERENCE

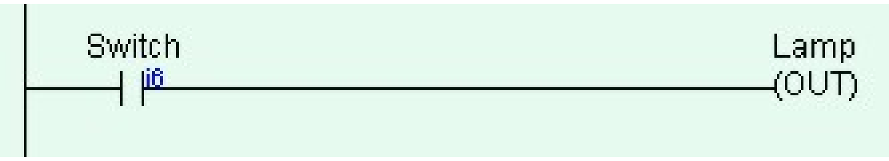
9.1 Ladder Logic Fundamentals: Contacts, Coils, Timers and Counters

9.1.1 Contacts

Ladder logic programs mimic the electrical circuit diagrams used for wiring control systems in the electrical industry. The basic purpose of an electrical control system is to determine whether a load should be turned ON or turned OFF, under what circumstances and when it should happen. To understand a ladder program, just remember the concept of current flow - a load is turned ON when the current can flow to it and is turned OFF when the current could not flow to it.

The fundamental element of a ladder diagram is a "Contact". A contact has only two states: open or closed. An open contact breaks the current flow whereas a closed contact allows current to flow through it to the next element. The simplest contact is an On/OFF switch which requires external force (e.g. the human hand) to activate it. Limit switches are those small switches that are placed at certain location so that when a mechanical device moves towards it, the contact will be closed and when the device moves away from it, the contact will be open.

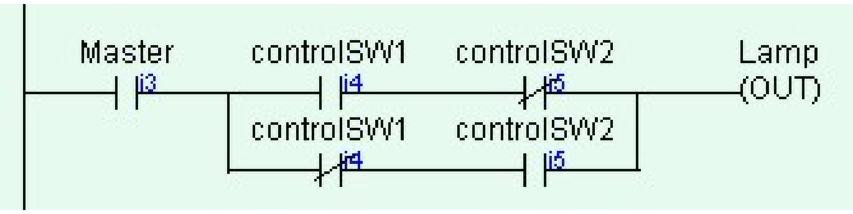
If a contact is connected to a load and the contact is closed, the load will be turned ON. This simple concept can be illustrated by the most basic ladder diagram as follow:



The vertical line on the left is the "Power" line, current must flow through the "Switch" contact in order to turn ON the load "Lamp". (In fact, there should be a second vertical line on the right end of the ladder diagram to provide a return path for the current flow, but this is omitted to simplify the circuit diagram). Now, if instead of wiring the switch to the lamp directly as suggested in the above diagram, you could connect the switch to the PLC's input and connect the lamp to the PLC's output, and then write the above ladder program to perform the same job. Of course it makes little sense to use a PLC if that is all you want to do. We will see how a PLC can simplify wiring shortly.

Note:
The contact "Switch" shown in the above diagram is termed a Normally-open (N.O.) contact.

Now, let's say if there are 3 switches that must work together to control the lamp. A Master switch must be ON, and one of the two control switches "controlsw1" and "controlsw2" must be ON while the other must be OFF in order to turn ON the lamp (think of three-way switches in your house and you will get the idea). We can wire all 3 switches to 3 inputs of the PLC and the lamp to the output of the PLC.   We can write the following ladder program to perform this task:



A contact with a "/" across its body is a Normally-Closed (N.C.) contact. What it means is that the ladder program is using the "inverse" of the logic state of the input to interpret the diagram.

Hence in the above ladder diagram, if "Master" and "controlSW1" are turned ON but "controlSW2" is turned OFF, the lamp will be turned ON since the inverse logic state of an OFF state "controlSW2" is true. Think of an imaginary current flowing through the "Master" contact, then through the "controlSW1" and finally through the normally-closed "controlSW2" contact to turn ON the lamp.

On the other hand, if "controlSW1" is OFF but "controlSW2" is ON, the Lamp is also turned ON because the current could flow via "Master" and then through the lower parallel branch via N.C. "controlSW1" and the N.O. "controlSW2".

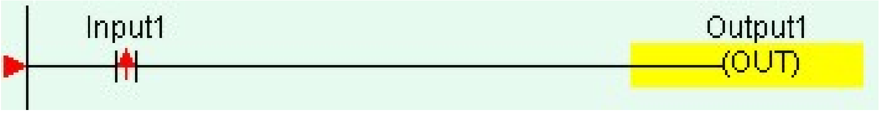
Note:
As you can see, although the switch "controlSW1" is connected to only 1 physical input to the PLC, but it appears twice in the ladder diagram. If you actually try to connect physical wires to implement the above circuits, both "controlSW1" and "controlSW2" will have to be of multiple poles type. But if you use a PLC, then these two switches only need to be of single-pole type since there is only one physical connection which is to the input terminal of the PLC. But in the ladder diagram the same contact may appear as many times as you wish as if it has unlimited number of poles.

The above example may be simple but it illustrates the basic concept of logical AND and OR very clearly. "controlSW1" and "controlSW2" are connected in series and both must be TRUE for the outcome to be TRUE. Hence, this is a logical AND connection. On the other hand, either of the two parallel branches may be used to conduct current and, hence, this is a logical OR connection.

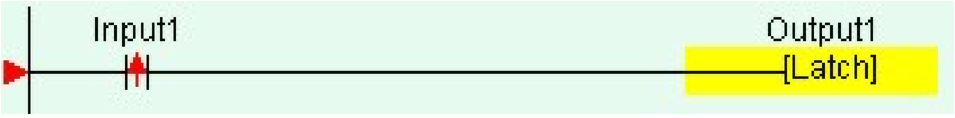
Another type of contact, available for all of the Ethernet PLCs only, is the Rising-Edge contact, which looks like this: in the ladder logic toolbar and like this: when placed in a ladder logic circuit. This type of contact will detect a change of status from off to on and then send a single pulse out (one shot). This contact can be used for any physical input or output or any internal bits (relay, counter, timer). In the case of physical input and output rising-edge contacts, a rising edge will be detected if the I/O has changed from off to on from one I/O scan to another (any I/O status changes that happen during a ladder logic scan won't matter for physical I/O). In the case of the internal bit rising-edge contacts, a rising edge will be detected if the internal bit has changed from off to on from one ladder logic scan to another (any internal bit status changes that happen during an I/O scan won't matter for internal bits).

<u>Here are some examples of this for the physical I/O:</u>

Ex1. In the circuit shown below, if Input1 is off for an I/O scan and then on for the next I/O scan, a single pulse (one shot) will be sent to Output1 and Output1 will be turned on for one program scan time (turned off on the next I/O scan). On the same I/O scan that Output1 is turned off and all following I/O scans, whether Input1 is still on or has been turned off, a rising edge will not have been detected and Output1 will remain off. For a new rising edge to be detected, Input1 must be scanned as off first and then scanned as on in a following I/O scan.



Ex2. In the circuit below, when Input1 goes from the off status to on status (as described previously), a single pulse will be sent to a [Latch] function that will latch Output1 to the on status. The actual Latch function will only be activated for a single scan time (just as Output1 was in the previous example), but it will permanently latch Output1 to on until Output1 is unlatched using the [Clear] function. This way Output1 will remain on even though Input1 has only sent a single pulse and Output1 will not be affected by any further rising-edge detections from Input1 (can only be latched once until it is unlatched).



The same principles can be applied to internal bits and coils as were previously described for physical input contacts and output coils.

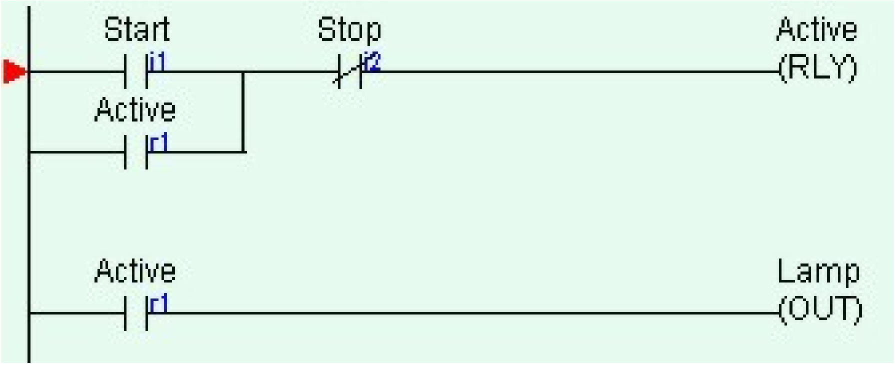
Once you understand these fundamental principles of interpretting a ladder diagram, everything should become clearer and simpler. Ladder diagram programming can be used to create a rather sophisticated control system. However, In i-TRiLOGI we augment its power further by allowing a ladder program to activate customized functions created in TBASIC.

9.1.2 Relay Coils

A contact can also be activated by the presence of an electrical current. This makes it possible for a control system to control the turning ON or OFF of a large load by using electrical current to activate a switch that can conduct high current. The simplest form of this type of contact is a relay.

In traditional electromagnetic relay, a coil of wire is wound around an iron core which turns it into an electromagnet. When current passes through the "coil" the magnet is "energized" and the force is used to either close a contact (that makes it a normally-open contact, closed only when energized) or open it (that will be a normally-closed contact since it is closed when not energized).

Ladder Logic programming language borrows some of those terms used to describe the electromagnetic relay for its own use. You connect a relay coil to the right end of the ladder diagram just like an output, as follows:



In a PLC, there are hundreds of internal "relays" which are supposed to behave like the typical electromagnetic relay. Unlike an output (e.g. the "Lamp" output) which has a physical connection out of the PLC, when an internal relay is turned ON, it is said to be "energized" but you will not see any changes in the PLC's physical I/Os. The logic state is kept internally in the PLC. The contact of the relay can then be used in the ladder diagram for turning ON or OFF of other relays or outputs. A relay contact in the ladder diagram can be Normally-Open or Normally-Closed and there is no limit to the number of contacts a relay can have.

9.1.3 Out Coils

A PLC output is really just an internal relay with a physical connection that can supply electrical power to control an external load. Thus, like a relay, an output can also have unlimited number of contacts that can be used in the ladder program.

9.1.4 Timer Coils

A timer is a special kind of relay that, when its coil is energized, must wait for a fixed length of time before closing its contact. The waiting time is dependent on the "Set Value" (SV)of the timer. Once the delay time is up, the timer's N.O. contacts will be closed for as long as its coil remains energized. When the coil is de-energized (i.e. turned OFF), all the timer's N.O. contacts will be opened immediately.

However, if the coil is de-energized before the delay time is up, the timer will be reset and its contact will never be closed. When a last aborted timer is re-energized, the delay timing will restart afresh using the SV of the timer and not continue from the last aborted timing operation.

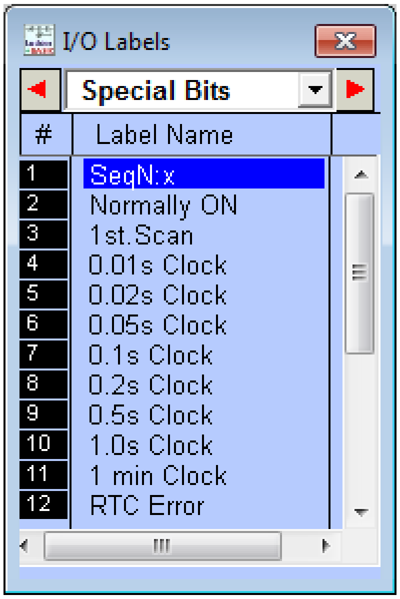
9.1.5 Counter Coils

A counter is also a special kind of relay that has a programmable Set Value (SV). When a counter coil is energized for the first time after a reset, it will load the value of SV-1 into its count register. From there on, every time the counter coil is energized from OFF to ON, the counter decrement its count register value by 1. Note that the coil must go through a complete OFF to ON cycle in order to decrement the counter. If the coil remains energized all the time, the counter will not decrement. Hence counter is suitable for counting the number of cycles an operation has gone through.

When the count register hits zero, all the counter's N.O. contacts will be turned ON. These counter contacts will remain ON regardless of whether the counter's coil is energized or not. To turn OFF these contacts, you have to reset the counter using a special counter reset function [RSctr].

9.2 Special Bits

i-TRiLOGI contains a number of special purpose bits that are useful for certain applications. These include 8 clock pulses ranging from periods of 0.01 second to 1 minute, a "Normally-ON" flag and a "First Scan Pulse", etc. To use any of these bits, enter the ladder editor and create a "contact"; when the I/O table pops up, scroll the windows until a "Special Bits" menu pops up. This menu is located after the "Counter Table" and before the "Input" table. as shown below:



#	Label Name
1	SeqN:x
2	Normally ON
3	1st.Scan
4	0.01s Clock
5	0.02s Clock
6	0.05s Clock
7	0.1s Clock
8	0.2s Clock
9	0.5s Clock
10	1.0s Clock
11	1 min Clock
12	RTC Error

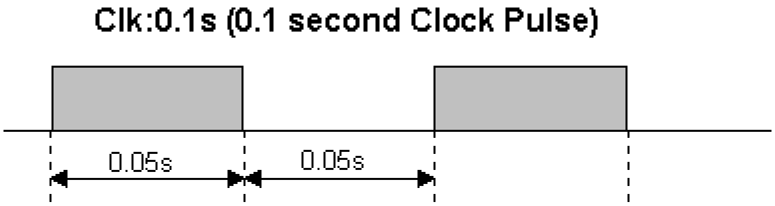
9.2.1 Clock pulse bits

The 8 clock pulses supported by i-TRiLOGI are:

Clock Pulse Period	**Ladder Symbol**
0.01 second	Clk:.01s
0.02 second	Clk:.02s
0.05 second	Clk:.05s
0.1 second	Clk:0.1s
0.2 second	Clk:0.2s
0.5 second	Clk:0.5s

1.0 second	Clk:1.0s
1 minute	Clk:1min

A clock pulse bit is ON for the first half of the rated period, then OFF for the second half. Duty cycles for these clock pulse bits are therefore 50%, as follow:



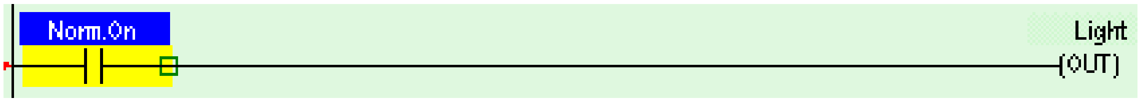
The clock pulse bits are often used with counter instructions to create timers. Additionally, they can be used as timing source for "Flasher" circuit. A reversible counter can also work with a clock pulse bit to create secondary clock pulses of periods that are multiples of the basic clock pulse rate.

9.2.2 SeqN:X

These are special "Sequencer" contacts which are activated only when the step counter of a Sequencer N reaches step #X. E.g. a Normally Open contact Seq2:6 is closed only when Sequencer #2 reaches Step #6. At any other step, this contact is opened. Click this link for detailed explanation and working examples on how to use a Sequencer.

9.2.3 Normally ON Flag - Norm.ON

You can make use of this flag if you need to keep something permanently ON regardless of any input conditions. This is because with the exception of Interlock Off function ———[ILoff], a coil or a special function is not allowed to connect directly to the power line (the vertical line on the left end of the ladder diagram). If you need to permanently enable a coil, consider using the "Normally-ON" bit from the "Special Bits" menu, as follow:



9.2.4 First Scan Pulse - 1st.Scan

This special bit will only be turned ON in the very first scan time of the ladder program. After that it will be permanently turned OFF. This is useful if you need to initialize certain conditions at the beginning. When the program is transferred to the PLC, this bit will only be ON when the PLC is first powered up or after it has been reset.

9.2.5 Real Time Clock Error - RTC.Err

FMD and Nano-10 PLC

This bit is turned ON if an Nano-10 or FMD PLC does not have one of the following installed: FRAM-RTC, FRAM-RTC-0, FRAM-RTC-256 or I2C-FRAM-RTC and the clock has been reset due to power failure or watchdog timer reset. This gives warning to applications that require a correct real world time (such as scheduled ON/OFF operation) that the clock data is incorrect, hence enabling corrective action to be taken.

F-Series PLC

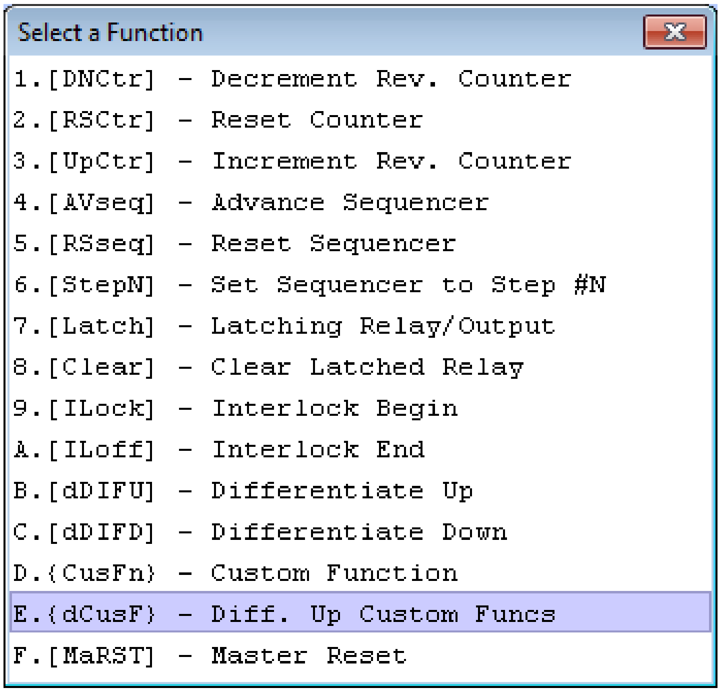
Since the F-Series PLCs do not use the MX-RTC and instead have an on-board batter-backed RTC, the RTC Error event occurs if the RTC is corrupted or damaged (see the F-Series User Manual - section 12.8 for more detail) or if the battery is not installed. The special bit is called RTC.Err and can be obtained from the “Special Bits” I/O Table. The RTC.Err contact can be used to activate an alarm of some kind.

M-Series PLC (Legacy)

This bit is turned ON if the M-series PLC does not have battery-backed MX-RTC option and the clock has been reset due to power failure or watchdog timer reset. This gives warning to applications that require a correct real world time (such as scheduled ON/OFF operation) that the clock data is incorrect, hence enabling corrective action to be taken.

9.3 Special Functions

During ladder circuit editing, when you click on the —[Fn] or └[Fn] icon to create a special function coil, a special function menu will pop up as shown below:



Select a Function

```
1.[DNCtr] - Decrement Rev. Counter
2.[RSCtr] - Reset Counter
3.[UpCtr] - Increment Rev. Counter
4.[AVseq] - Advance Sequencer
5.[RSseq] - Reset Sequencer
6.[StepN] - Set Sequencer to Step #N
7.[Latch] - Latching Relay/Output
8.[Clear] - Clear Latched Relay
9.[ILock] - Interlock Begin
A.[ILoff] - Interlock End
B.[dDIFU] - Differentiate Up
C.[dDIFD] - Differentiate Down
D.{CusFn} - Custom Function
E.{dCusF} - Diff. Up Custom Funcs
F.[MaRST] - Master Reset
```

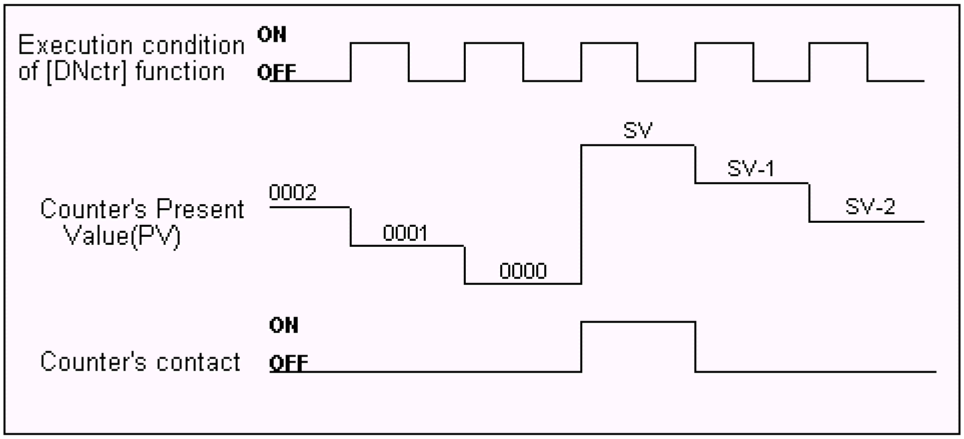
9.3.1 Reversible Counter Functions: [DNctr], [Upctr] and [RSctr]

The [DNctr], [UPctr] and [RSctr] functions work together to implement reversible counter functions on any of the 128 counters supported by i-TRiLOGI.

The ordinary down-counter (created by clicking on the —( ) icon) essentially decrements the counter value by 1 from the "Set Value" (SV) and will stop when its count becomes zero. Unlike the ordinary down-counter, a reversible counter is a circular counter which changes the counter present value (PV) between 0 and the SV. When you try to increment the counter past the "Set Value", it will overflow to become '0'. Likewise if you try to decrement the counter beyond '0', it will underflow to become the "Set Value".

All three counter functions [DNctr], [UPctr] and [RSctr] can operate on the same counter (i.e. assigned to the same counter label) on different circuits. Although these circuits may be located anywhere within the ladder program, it is recommended that the two or three functions which operate on the same counter be grouped together in the following order: DNctr], [Upctr] and [RSctr]. Note that NOT all three functions need to be used to implement the reversible counter.

Decrement Counter [DNctr]


Execution condition of [DNctr] function
ON
OFF
Counter's Present Value(PV)
0002
0001
0000
SV
SV-1
SV-2
Counter's contact
ON
OFF

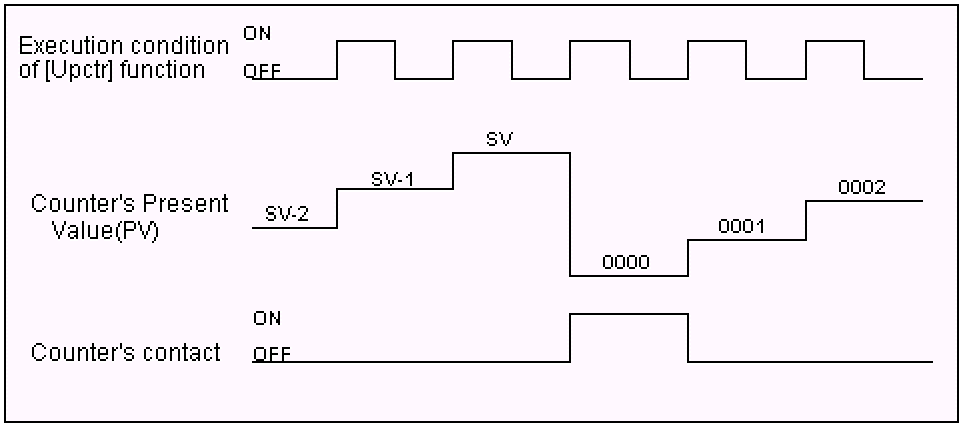
Each time when the execution condition of a [DNctr] function changes from OFF to ON, the present value of the designated counter is changed as follow:

a) If the counter's present value (PV) is inactive, load the counter register with the "Set Value" (SV, defined in the Counter table) minus 1.

b) If the counter's present value (PV) is already '0', then load the counter's PV with the SV defined in the counter table and turn on the counter's contact (also known as the completion flag).

c) Otherwise, decrement the counter PV register by 1.

<u>Increment Counter [Upctr]</u>


Execution condition of [Upctr] function
ON
OFF
Counter's Present Value(PV)
SV-2
SV-1
SV
0000
0001
0002
Counter's contact
ON
OFF

Each time when the execution condition of an [Upctr] function changes from OFF to ON, the present value of the designated counter is affected as follow:

a) If the counter is inactive, load the counter register with the number '0001'.

b) If the counter's present value (PV) is equal to the Set Value (SV, defined in the Counter table), load the counter register with number '0000' and turn on the counter's contact (also known as the completion flag).

c) Otherwise, increment the counter PV register by 1.

Reset Counter [RSctr]

When the execution condition of this function changes from OFF to ON, the counter will reset to its inactive state. This function is used to reset both a reversible counter and an ordinary down-counter coil.

9.3.2 Sequencer Functions: [AVseq], [RSseq] and [StepN]

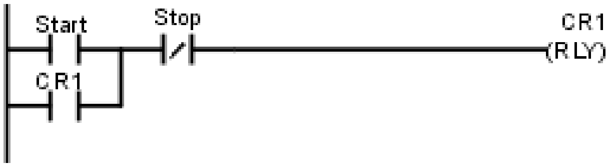
Please refer to the documentation on Using i-TRiLOGI Sequencers

9.3.3 Latch Relay Function [Latch]

Latching relay is convenient for keeping the status of an execution condition even if the condition is subsequently removed. The program elements that are assigned as Latching Relays will remain ON once they are energized. Only Relays and Outputs may be assigned as Latching Relays.

On selecting [Latch] function, you can use the left/right cursor keys or click on the left/right arrow keys to move between the Relay and Output tables. The selected relay or output will now be assigned as a Latching Relay. You will be able to see the label name of the program element above the [Latch] symbol in the ladder diagram.

Although latch-relay can be used in place of self-latching (Seal) circuits, a latch-relay in an interlock section will not be cleared when the interlock occurs. Only a self-latching circuit as shown in the following will be cleared in an interlock section:



9.3.4 Clear Relay Function [Clear]

To de-energize a program element that has been latched by the [Latch] function, it is necessary to use [Clear] function. On selecting [Clear], choose the output or relay to be de-energized. When the execution condition for that circuit is ON, the designated output or relay will be reset. In the ladder diagram, the program element label name will be shown above the [Clear] symbol.

If the execution condition for [Latch] and [Clear] functions are both ON at the same time, then the effect of the designated bit depends on the relative locations of these two functions. Remember that an output or relay bit energized by [Latch] will remain ON until it is turned OFF by [Clear]. It is recommended that [Clear] circuit be placed just after the [Latch] circuit for the same output or relay controlled by these two functions. This ensures that [Clear] function has higher priority over [Latch] function, which is normally so in hardware latch-relay or other industrial PLCs.

9.3.5 Interlock [ILock]

The "Interlock" [ILock] and "Interlock Off" [ILoff] functions work together to control an entire section of ladder circuits. If the execution condition of an [ILock] function is ON, the program will be executed as normal. If the execution condition of [ILock] is OFF, the program elements between the [ILock] and [ILoff] will behave as follow:

a) all output coils are turned OFF.

b) all timers are reset to inactive.

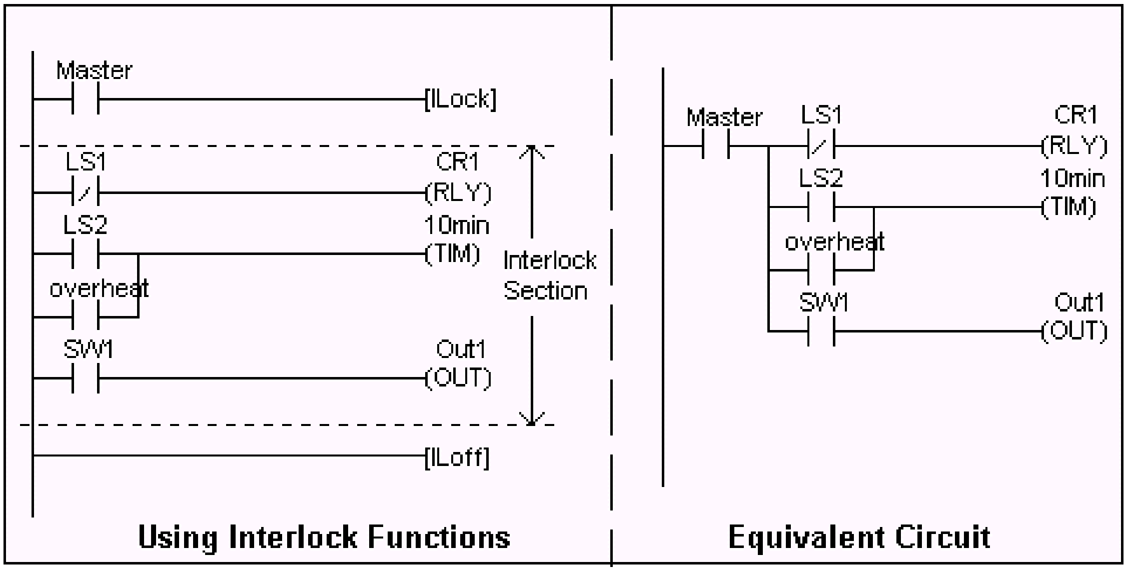
c) all counters retain their present values.

d) Latched relays by [Latch] function are not affected.

e) [dDIFU] and [dDIFD] functions are not executed.

f) all other functions are not executed.

An Interlock section is equivalent to a master control relay controlling a number of sub-branches as follow:



Using Interlock Functions

Equivalent Circuit

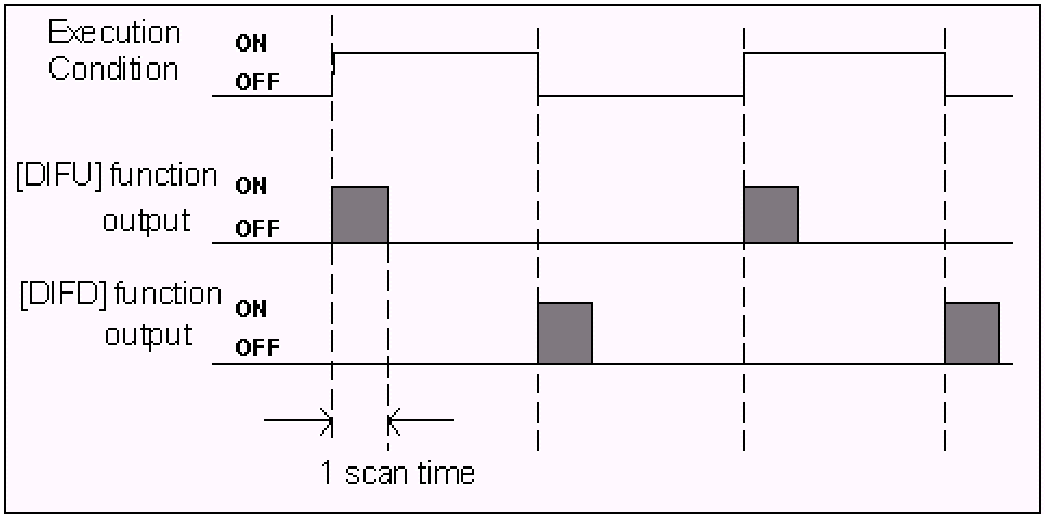
Note that [ILoff] is the only function that does not need to be energized by other program elements. When you use one or more [ILock] functions, there must be at least one [ILoff] function before the end of the program. Otherwise the compiler will warn you for the missing [ILoff]. The logic simulator always clears the Interlock at the end of the scan if you omit the [ILoff] function.

You can program a second or third level Interlock within an Interlock section using a few [ILock] functions. However, you only need to program one [ILoff] function for the outermost Interlock section, i.e. [ILoff] need not be a matching pair for an [ILock] function.

9.3.6 Differentiate Up and Down [d DIFU] and [d DIFD]

When the execution condition for [dDIFU] goes from OFF to ON, the designated output or relay will be turned ON for one scan time only. After that it will be turned OFF. This means that the function generates a single pulse for one scan time in response to the rising-edge of its execution condition. When its execution condition goes from ON to OFF nothing happens to the output or relay that it controls.

On the other hand, when the execution condition for [dDIFD] goes from ON to OFF, the designated output or relay will be turned ON for one scan time only. After that it will be turned OFF. This means that the function generates a single pulse for one scan time in response to the trailing-edge of its execution condition. When its execution condition goes from ON to OFF, nothing happens to the output or relay that it controls.



NOTE:
The [dDIFU] function can now be implemented as a single contact with the Rising-Edge contact that is triggered once every time a rising edge is sensed. More information on this new contact can be found in the Contacts section of the "Contacts and Coils" help page.

9.3.7 Custom Functions: [CusFn] and [dCusF]

These two functions allow you to connect a user-defined custom function (CusFn) to the ladder logic as if it is a relay coil. Custom functions are created using the integrated editor provided by i-TRiLOGI Version6.x. Please refer to TBASIC Reference manual for detailed descriptions of custom function creation and deployment methods.

9.3.8 Master Reset

An ON condition to this function clears all mailbox inputs, outputs, relays, timers and counter bits, resets all timers counters/sequencers to inactive state, and clears all latched relay bits. All integer variables will be cleared to zeros and all string variables will be assigned to empty string.

9.4 Using i-TRiLOGI Sequencers

A sequencer is a highly convenient feature for programming machines or processes which operate in fixed sequences. These machines operate in fixed, clearly distinguishable step-by-step order, starting from an initial step and progressing to the final step and then restart from the initial step again. At any moment, there must be a "step counter" to keep track of the current step number. Every step of the sequence must be accessible and can be used to trigger some action, such as turning on a motor or solenoid valve, etc.

As an example, a simple Pick-and-Place machine that can pick up a component from point 'A' to point 'B' may operate as follow:

Step #	**Action**
0	Wait for "Start" signal
1	Forward arm at point A
2	Close gripper
3	Retract arm at point A
4	Move arm to point B
5	Forward arm at point B
6	Open gripper
7	Retract arm at point B
8	Move arm to point A

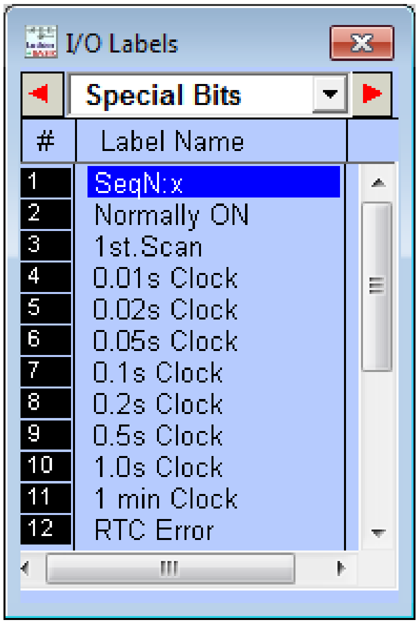
i-TRiLOGI Version 6 and higher supports eight sequencers of 32 steps each. Each sequencer uses one of the first eight counters (Counter #1 to Counter #8) as its step counter. Any one or all of the first eight counters can be used as sequencers "Seq1" to "Seq8".

To use a sequencer, first define the sequencer name in the Counter table by pressing the <F2> key and scroll to the Counter Table. Any counter to be used as sequencer can only assume label names "Seq1" to "Seq8" corresponding to the counter numbers. For e.g. if Sequencer #5 is to be used, Counter #5 must be defined as "Seq5". Next, enter the last step number for the program sequence in the "Value" column of the table.

Construct a circuit that uses the special function "Advance Sequencer" [AVSeq]. The first time the execution condition for the [AVseq] function goes from OFF to ON, the designated sequencer will go from inactive to step 1. Subsequent change of the sequencer's execution condition from OFF to ON will advance (increment) the sequencer by one step. This operation is actually identical to the [UPctr] instruction.

The upper limit of the step counter is determined by the "Set Value" (SV) defined in the Counter table. When the SV is reached, the next advancement of sequencer will cause it to overflow to step 0. At this time, the sequencer's contact will turn ON until the next increment of the sequencer. This contact can be used to indicate that a program has completed one cycle and is ready for a new cycle.

Accessing individual steps of the sequencer is extremely simple when programming with i-TRiLOGI. Simply create a "contact" (NC or NO) in ladder edit mode. When the I/O window pops up for you to pick a label, scroll to the "Special Bits" table as follow:



#	Label Name
1	SeqN:x
2	Normally ON
3	1st.Scan
4	0.01s Clock
5	0.02s Clock
6	0.05s Clock
7	0.1s Clock
8	0.2s Clock
9	0.5s Clock
10	1.0s Clock
11	1 min Clock
12	RTC Error

The "Special Bits" table is located after the "Counters" table and before the "Inputs" table. Then click on the "SeqN:x" item to insert a sequencer bit. You will be prompted to select a sequencer from a pop-up menu. Choose the desired sequencer (1 to 8) and another dialog box will open up for you to enter the specific step number for this sequencer.

Each step of the sequencer can be programmed as a contact on the ladder diagram as "SeqN:X" where N = Sequencers # 1 to 8. X = Steps # 0 - 31.

e.g. Seq2:4 = Step #4 of Sequencer 2.

Seq5:25 = Step #25 of Sequencer 5.

Although a sequencer may go beyond Step 31 if you define a larger SV for it, only the first 32 steps can be used as contacts to the ladder logic. Hence it is necessary to limit the maximum step number to not more than 31.

9.4.1 Special Sequencer Functions

Quite a few of the ladder logic special functions are related to the use of the sequencer. These are described below:

1. Advance Sequencer - [AVseq]

Increment the sequencer's step counter by one until it overflows. This function is the identical to (and hence interchangeable with) the [UpCtr] function.

2. Resetting Sequencer - [RSseq]

The sequencer can also be reset to become inactive by the [RSseq] function at any time. Note that a sequencer that is inactive is not the same as sequencer at Step 0, as the former does not activate the SeqN:0 contact. To set the sequencer to step 0, use the [StepN] function described next.

3. Setting Sequencer to Step N - [StepN]

In certain applications it may be more convenient to be able to set the sequencer to a known step asynchronously. This function will set the selected sequencer to step #N, regardless of its current step number or logic state. The ability to jump steps is a very powerful feature of the sequencers.

4. Reversing a Sequencer

Although not available as a unique special function, a Sequencer may be stepped backward (by decrementing its step-counter) using the [DNctr] command on the counter that has been defined as a sequencer. This is useful for creating a reversible sequencer or for replacing a reversible "drum" controller.

9.4.2 Other Applications

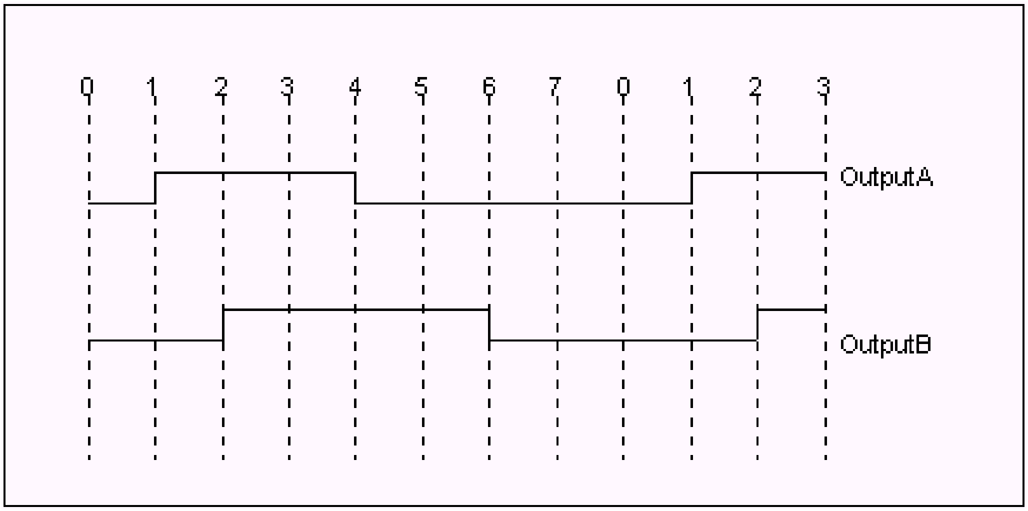
a. Driving Stepper Motor

A sequencer may be used to drive a stepper motor directly. A two-phase stepper motor can be driven by four transistor outputs of the controller directly (for small motors with phase current < 0.5A) or via solid-state relays. The stepper motor can be driven using a sequencer that cycles through Step#0 to Step#3 (full-step mode) or Step#0 through Step#7 (half-step mode). Each step of the sequencer is used to energize different phases of the stepper motor. A clock source is needed to drive the stepper motor through its stepping sequence. The stepping rate is determined by the frequency (which is equal to 1/period) of the clock source.

Clock pulses with periods in multiples of 0.01 second can be generated easily using the "Clk:.01s" bit and an [Upctr] function. For e.g., to generate a clock source of period = 0.05s, use "Clk.01s" to feed to an [Upctr] counter with Set Value = 4. The counter's contact (completion flag) will be turned ON once every 5 counts (0,1,2,3,4), which is equivalent to a 0.05 sec. clock source.

b. Replacing a Drum Controller

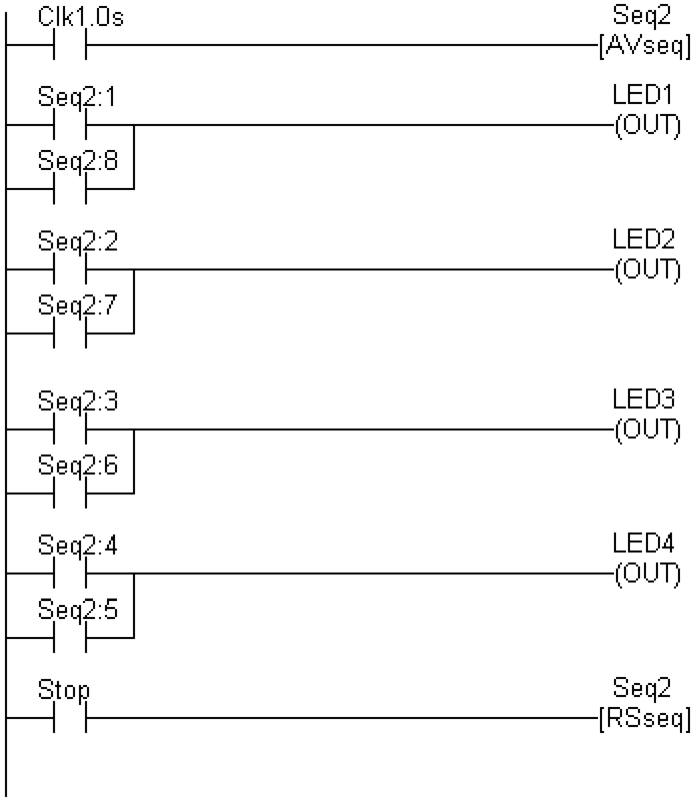
A drum controller can be replaced easily by a sequencer if the timing of the drum's outputs can be divided into discrete steps. Assuming a drum controls two outputs with the timing diagram shown in the following figure:



This can be replaced by an 8-step sequencer. Step 1 (e.g "Seq1:1") turns ON and latch Output A using [Latch] function, Step 2 turns ON and latch Output B, Step 4 turns OFF Output A using the [Clear] function, and Step 6 turns OFF Output B. All other steps (3,5,7,0) have no connection.

9.4.3 Program Example

Assume that we wish to create a running light pattern which turns on the LED of Outputs 1 to 4 one at a time every second in the following order: LED1, LED2, LED3, LED4, LED4, LED3, LED2, LED1, all LED OFF and then restart the cycle again. This can be easily accomplished with the program shown below.



The 1.0s clock pulse bit will advance (increment) Sequencer #2 by one step every second. Sequencer 2 should be defined with Set Value = 8. Each step of the sequencer is used as a normally open contact to turn on the desired LED for the step. A "Stop" input resets the sequencer asynchronously. When the sequencer counts to eight, it will become Step 0. Since none of the LED is turned ON by Step 0, all LEDs will be OFF.

Chapter 10 Introduction to TBASIC Custom Functions

10 INTRODUCTION TO TBASIC CUSTOM FUNCTIONS

10.1 Overview

i-TRiLOGI Version 6.xx supports user-created special functions, known as Custom Functions (the symbol CusFn will be used throughout this manual to mean Custom Functions). Up to 256 CusFns can be programmed using a special language: TBASIC.

TBASIC is derived from the popular BASIC computer language widely used by microcomputer programmers. Some enhancements as well as simplifications have been made to the language to make it more suitable for use in PLC applications.

There are four simple ways to create a new CusFn:

1. From the "Edit" pull-down menu, select the item "Edit Custom Function" and select the function number from a pop-up CusFn selection table which may range from 1 to 256. The selection table allows you to define unique and easily identifiable names for each custom function. Once you have selected the custom function the editor window will open up with the contents of that particular custom function.

2. You may also use the hotkey <F7> to open up the selection table.

3. If you have already created a ladder circuit which connects to either a [CusFn] or [dCusF] function (both appear as menu-items within the "Special Function" pop-up menu), then you can easily open up that particular CusFn by double clicking the left mouse button while the highlight bar is at the [CusFn] or [dCusFn]. Alternatively, you can open the CusFn by clicking the right mouse button while the highlight bar is at the [CusFn] or [dCusFn].

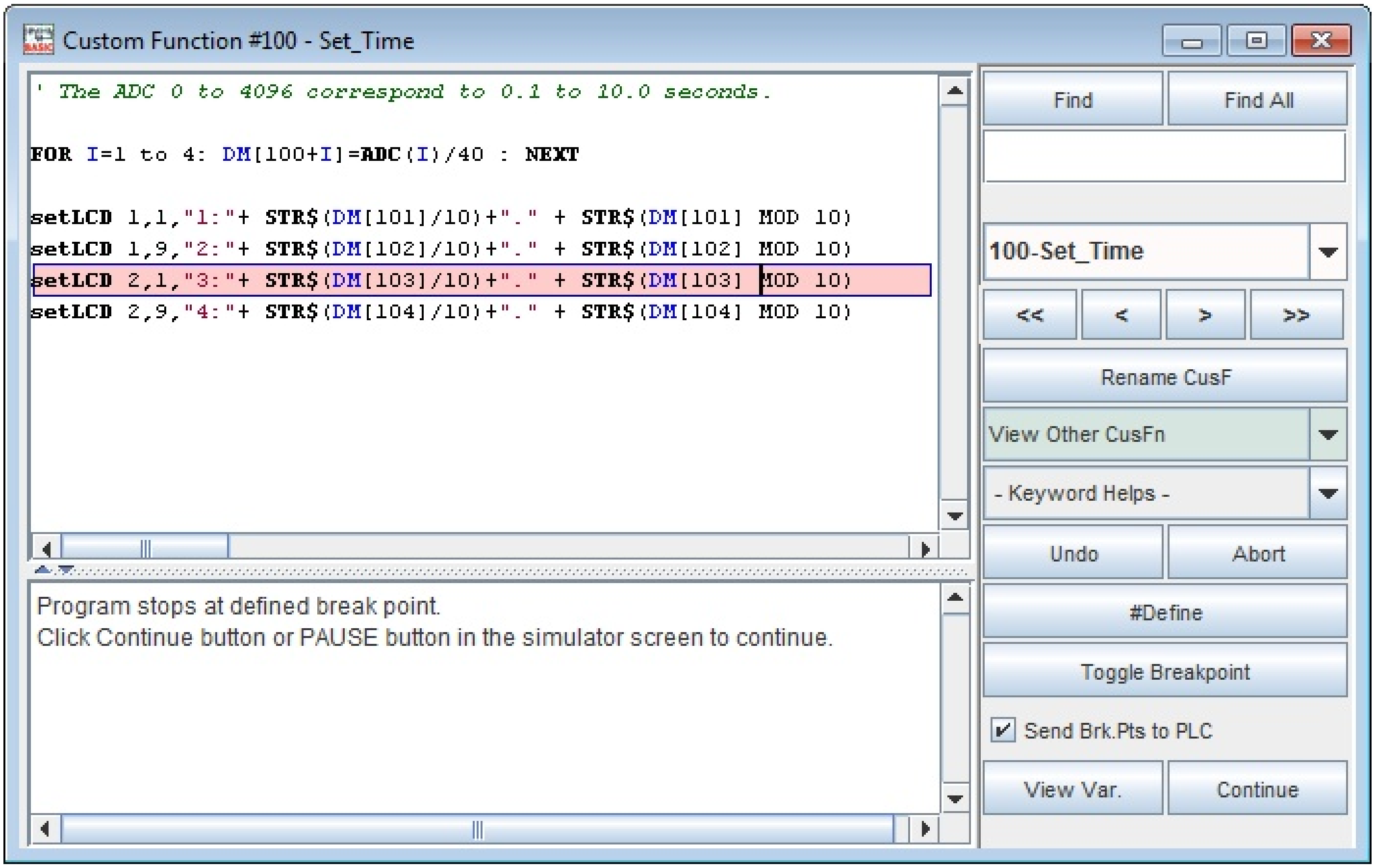
4. You can open a custom function by clicking on the "Open CusFn" button on the status bar below the pulldown menu..

10.2 Custom Function Editor

The custom function editor window allows creation of any number of lines of TBASIC program statements. Since this is a standard text editor, you should have no problem using the key and mouse controls to edit the text. E.g. To copy a paragraph of text, select it using the mouse and the press <Ctrl-C>.

Alternatively, you can use <Ctrl-X>, which will cut the text (copy and delete). Once you have copied or cut the desired text, move the text editing cursor to the destination and press <Ctrl-V> to paste it to the new location.

i-TRiLOGI 6.2 and up introduces a brand new custom function editor. The custom function editor now has a whole new interface with split panes that provide text editing for writing code (just like before) on the upper pane and a web browser/TBASIC help information on the lower pane. The first time the custom function editor is opened after starting i-TRiLOGI, the text editor pane will be empty and the lower pane will display some information about Triangle Research, our products, or software update information (see below). The lower pane can also be used to display help or error messages when writing or compiling your program. Update : A define button has been added as of i-TRiLOGI version 6.42 and higher and a breakpoint feature for debugging purposes.



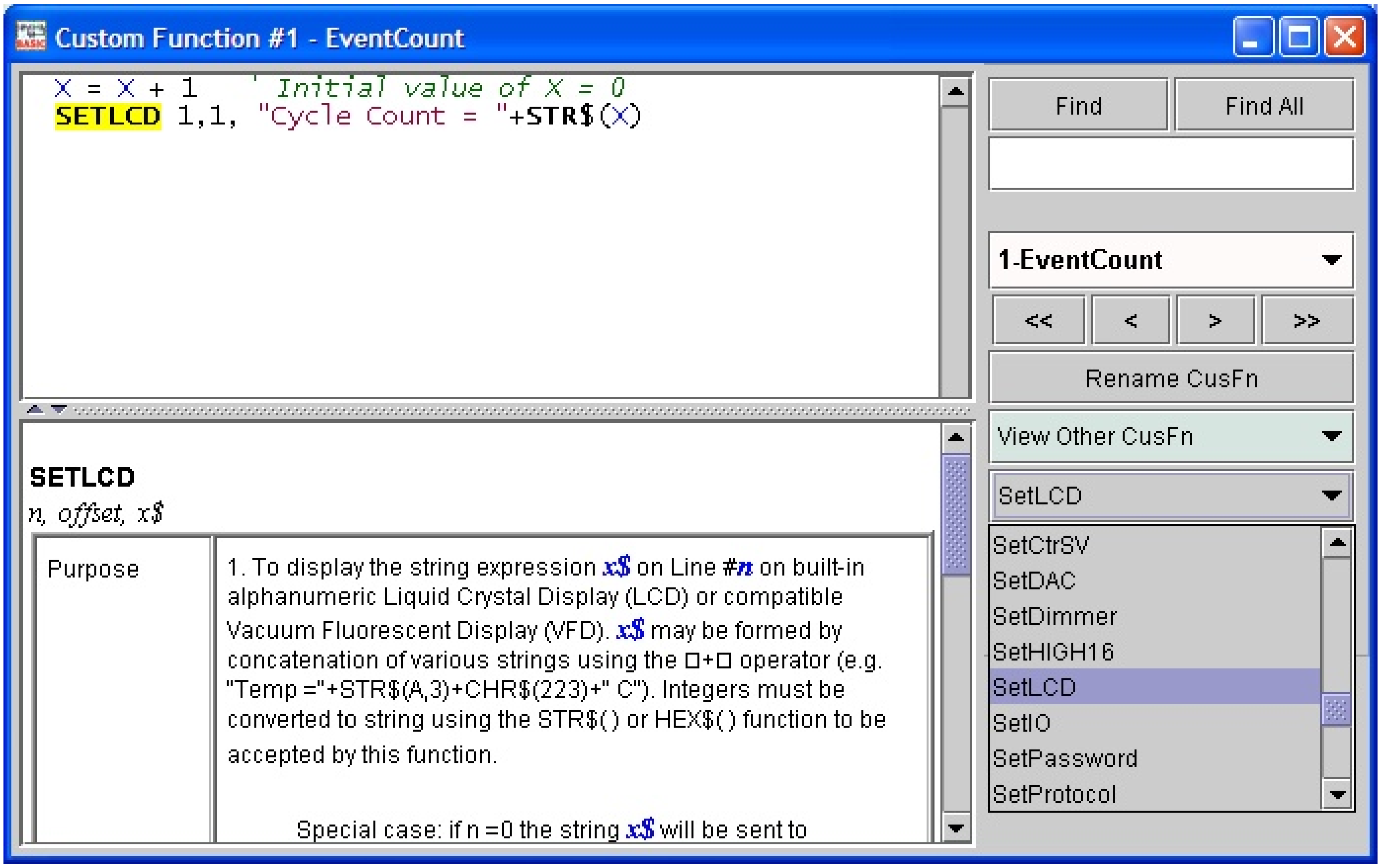
```
' The ADC 0 to 4096 correspond to 0.1 to 10.0 seconds.

FOR I=1 to 4: DM[100+I]=ADC(I)/40 : NEXT

setLCD 1,1,"1:"+ STR$(DM[101]/10)+"." + STR$(DM[101] MOD 10)
setLCD 1,9,"2:"+ STR$(DM[102]/10)+"." + STR$(DM[102] MOD 10)
setLCD 2,1,"3:"+ STR$(DM[103]/10)+"." + STR$(DM[103] MOD 10)
setLCD 2,9,"4:"+ STR$(DM[104]/10)+"." + STR$(DM[104] MOD 10)
```

Program stops at defined break point.
Click Continue button or PAUSE button in the simulator screen to continue.

☑ Send Brk.Pts to PLC

10.2.1 Split Pane Window

The upper pane is used as a text editor for writing TBASIC code and the lower pane can either display information about Triangle Research (as shown above), or provide immediate help for any TBASIC keyword (as shown below). To use the lower pane for help on TBASIC keywords, simply select the keyword from the “Select Keyword” drop box or highlight the text in the editor. For Example: in the following screenshot, the code "SETLCD" is highlighted and the syntax for "SETLCD" would be displayed as it is in the screenshot.



```
  X = X + 1    ' Initial value of X = 0
  SETLCD 1,1, "Cycle Count = "+STR$(X)
```

SETLCD
n, offset, x$

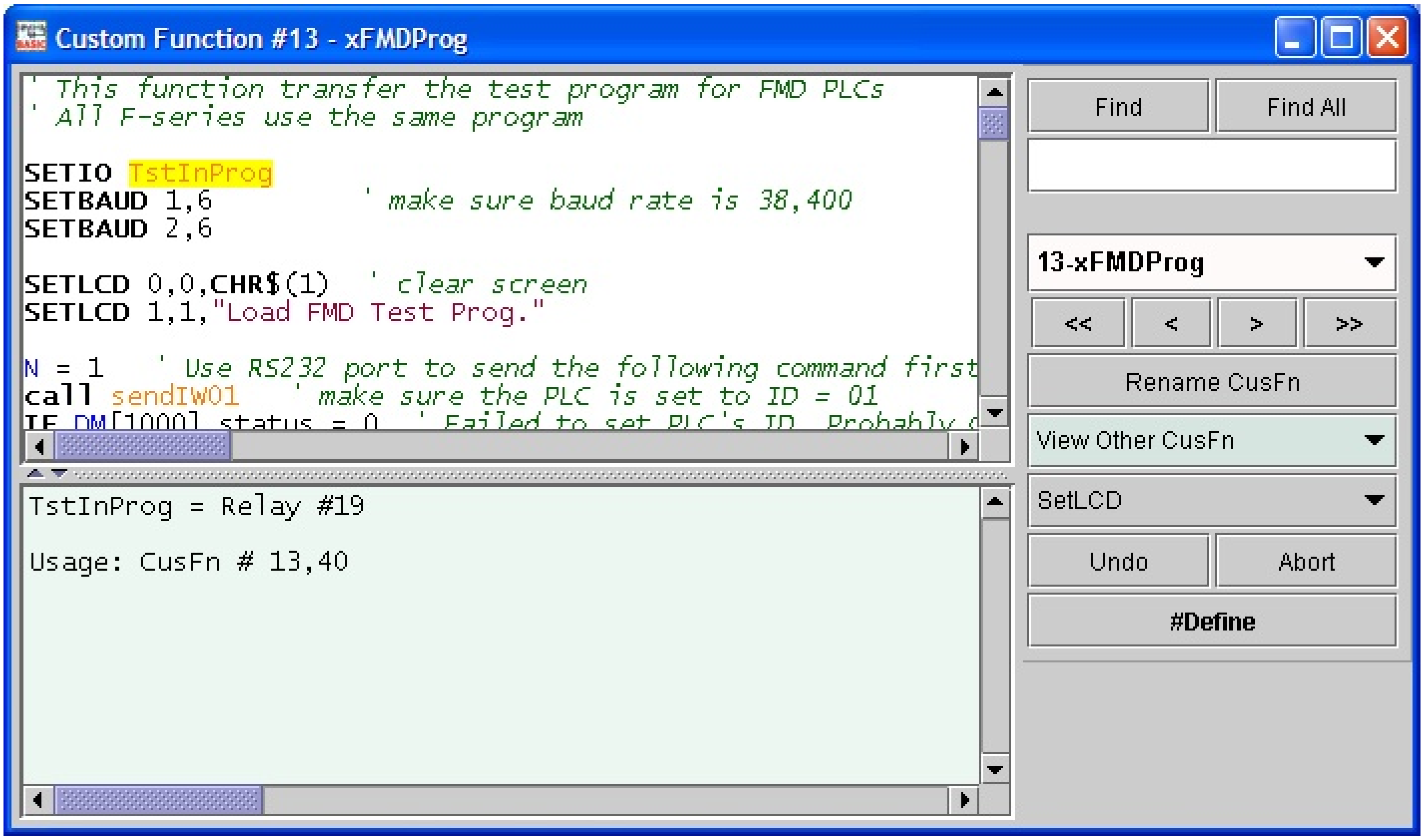
Purpose	1. To display the string expression ***x$*** on Line #***n*** on built-in alphanumeric Liquid Crystal Display (LCD) or compatible Vacuum Fluorescent Display (VFD). ***x$*** may be formed by concatenation of various strings using the □+□ operator (e.g. "Temp ="+STR$(A,3)+CHR$(223)+" C"). Integers must be converted to string using the STR$( ) or HEX$( ) function to be accepted by this function. Special case: if n =0 the string ***x$*** will be sent to

10.2.2 New Help Features.

i-TRiLOGI Version 6.43 added the following useful features to its Help Pane:

a) <u>Select an I/O Label</u>

If the selected text is an I/O label name, the lower pane will display its I/O type and the I/O number as well as the custom function # where the same label name have been used. This can be very useful to find out where a certain variable may be changed in other custom functions.



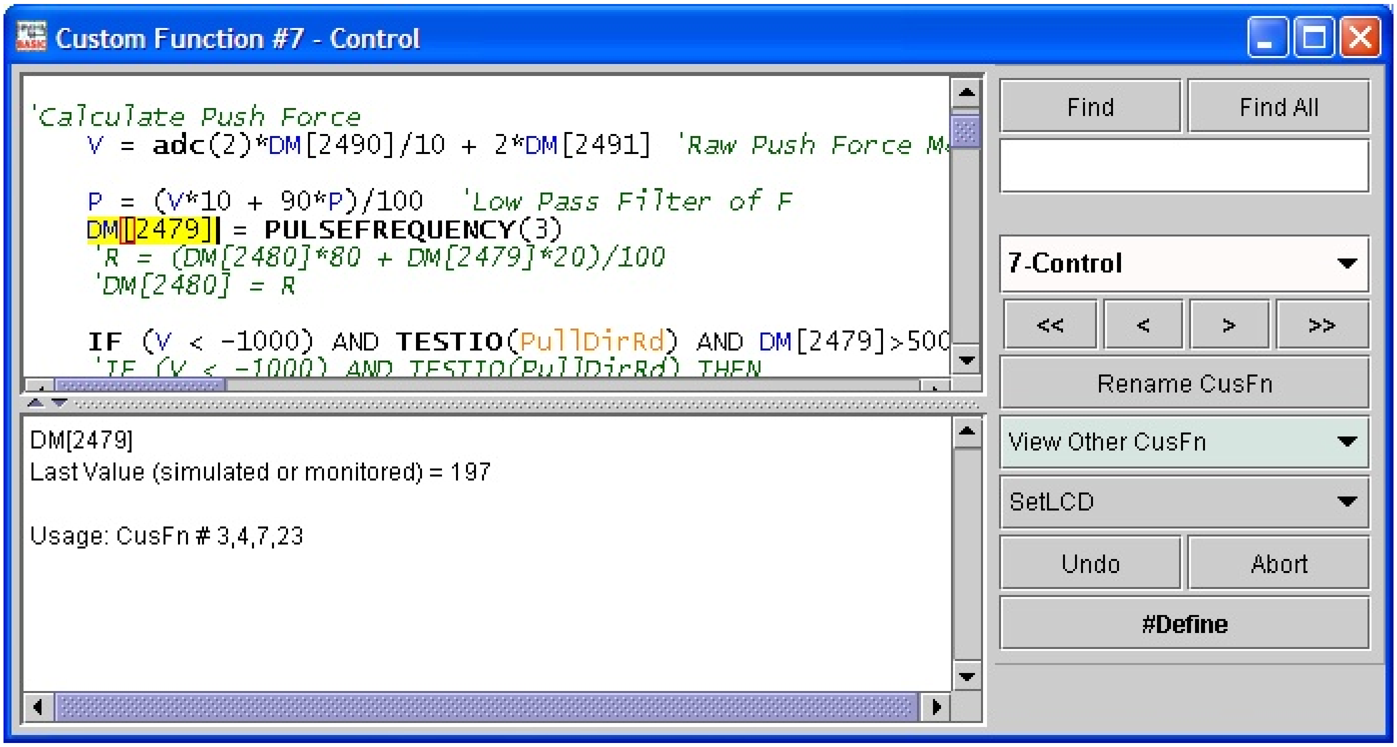
```
' This function transfer the test program for FMD PLCs
' All F-series use the same program

SETIO TstInProg
SETBAUD 1,6         ' make sure baud rate is 38,400
SETBAUD 2,6

SETLCD 0,0,CHR$(1)  ' clear screen
SETLCD 1,1,"Load FMD Test Prog."

N = 1    ' Use RS232 port to send the following command first
call sendIW01    ' make sure the PLC is set to ID = 01
IF DM[1000] status = 0   ' Failed to set PLC's ID. Probably
```

```
TstInProg = Relay #19

Usage: CusFn # 13,40
```

b) Select a Variable

If the selected text is a variable name or a defined name in the #Define table, then the lower pane will display the current value of the selected variable! The value is obtained from the simulator if i-TRiLOGI is currently disconnected from any TLServer or F-Server. If i-TRiLOGI is currently connected to a TLServer or F-server, then the program will retrieve the current value from the connected PLC.

In the following example, variable DM[2479] has a current value = 197 and it is being used in Custom function #3, 4, 7 and 24.



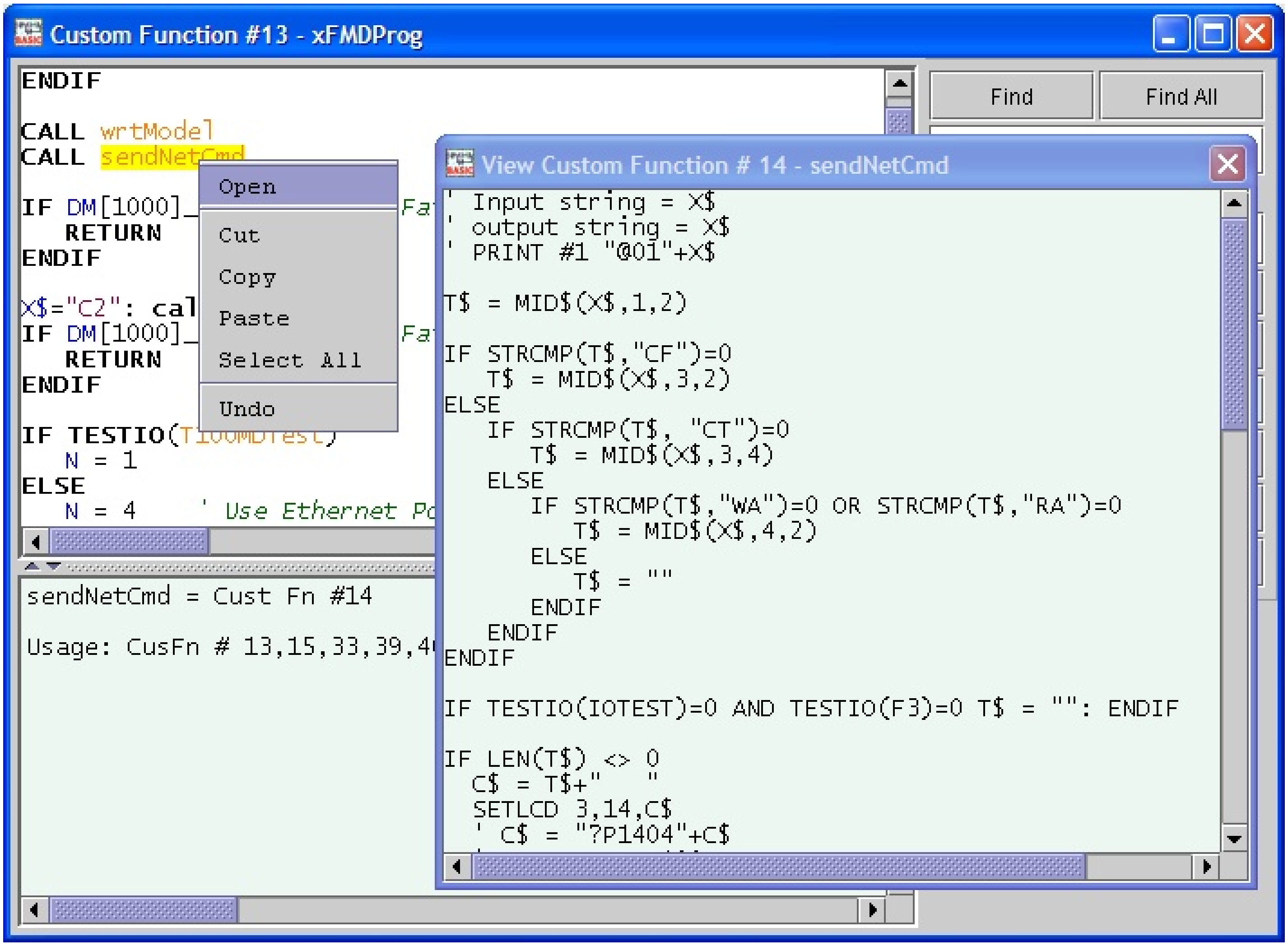
```
'Calculate Push Force
    V = adc(2)*DM[2490]/10 + 2*DM[2491]  'Raw Push Force M

    P = (V*10 + 90*P)/100  'Low Pass Filter of F
    DM[2479] = PULSEFREQUENCY(3)
    'R = (DM[2480]*80 + DM[2479]*20)/100
    'DM[2480] = R

    IF (V < -1000) AND TESTIO(PullDirRd) AND DM[2479]>500
    'IF (V < -1000) AND TESTIO(PullDirRd) THEN
```

```
DM[2479]
Last Value (simulated or monitored) = 197

Usage: CusFn # 3,4,7,23
```

c) <u>Select a Custom Function Name</u>

If the selected text is the name of a custom function, a special read-only popup window will appear which displays the content of the selected function. This enables the programmer to view the code of another custom function CALLed by this function without leaving the current function.

You can also edit the selected custom function by right-clicking on the selected name and select "Open" from the popup menu, as shown below.


Custom Function #13 - xFMDProg
Find
Find All
Open
Cut
Copy
Paste
Select All
Undo
View Custom Function # 14 - sendNetCmd

```
ENDIF

CALL wrtModel
CALL sendNetCmd

IF DM[1000]_                  Fa
   RETURN
ENDIF

X$="C2": cal
IF DM[1000]_                  Fa
   RETURN
ENDIF

IF TESTIO(T100MDTest)
   N = 1
ELSE
   N = 4      ' Use Ethernet Po
```

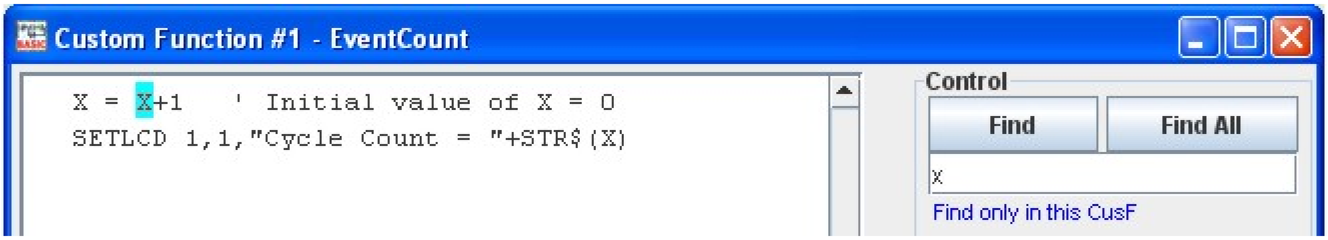
```
sendNetCmd = Cust Fn #14

Usage: CusFn # 13,15,33,39,4
```

```
' Input string = X$
' output string = X$
' PRINT #1 "@01"+X$

T$ = MID$(X$,1,2)

IF STRCMP(T$,"CF")=0
   T$ = MID$(X$,3,2)
ELSE
   IF STRCMP(T$, "CT")=0
      T$ = MID$(X$,3,4)
   ELSE
      IF STRCMP(T$,"WA")=0 OR STRCMP(T$,"RA")=0
         T$ = MID$(X$,4,2)
      ELSE
         T$ = ""
      ENDIF
   ENDIF
ENDIF

IF TESTIO(IOTEST)=0 AND TESTIO(F3)=0 T$ = "": ENDIF

IF LEN(T$) <> 0
  C$ = T$+"   "
  SETLCD 3,14,C$
  ' C$ = "?P1404"+C$
```

10.2.3 Search / Find

It is now possible to search for a word or phrase in the current custom function (local search) or in all of the custom functions in your program (global search).

To do a local search, simply type the text in the command line below the "Find" and "Find All" buttons and then click the "Find" button. If the text is found in the current custom function, it will be highlighted in the text editor as shown below. Also, the text "Find only in this CusF" will be displayed below the command line in the search area, indicating a local search. Each time the "Find" button is clicked, the next instance of the search text will be highlighted until the text cant be found anymore. At this point the message in the search area will change to "Text Not Found" and the next time "Find" is clicked, the first result will be highlighted again.

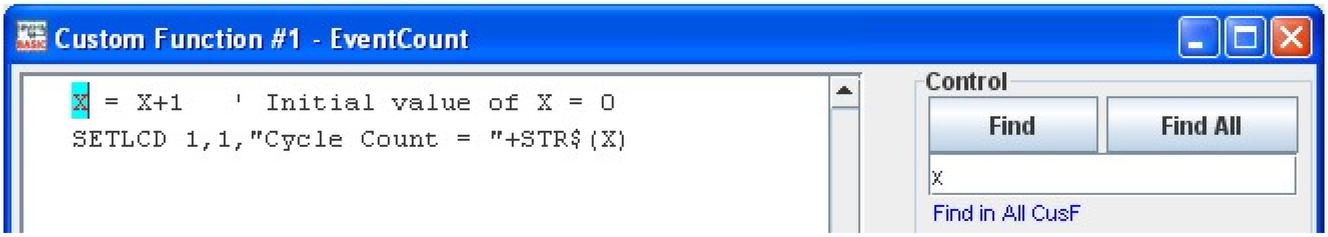


```
X = X+1    ' Initial value of X = 0
SETLCD 1,1,"Cycle Count = "+STR$(X)
```

x

Find only in this CusF

To do a global search, simply type the text in the same command line and click the "Find All" button. If the text is found in any custom function within the program, it will be highlighted in the text editor as shown below. Also, the text "Find in all CusF" will be displayed below the command line in the search area, indicating a global search. Each time the "Find All" button is clicked, the next instance of the search text will be highlighted until the text cant be found anymore (in any custom function). At this point the first result will be highlighted again. If no text matches the search text, the message in the search area will change to "Text Not Found".



```
X = X+1    ' Initial value of X = 0
SETLCD 1,1,"Cycle Count = "+STR$(X)
```

x

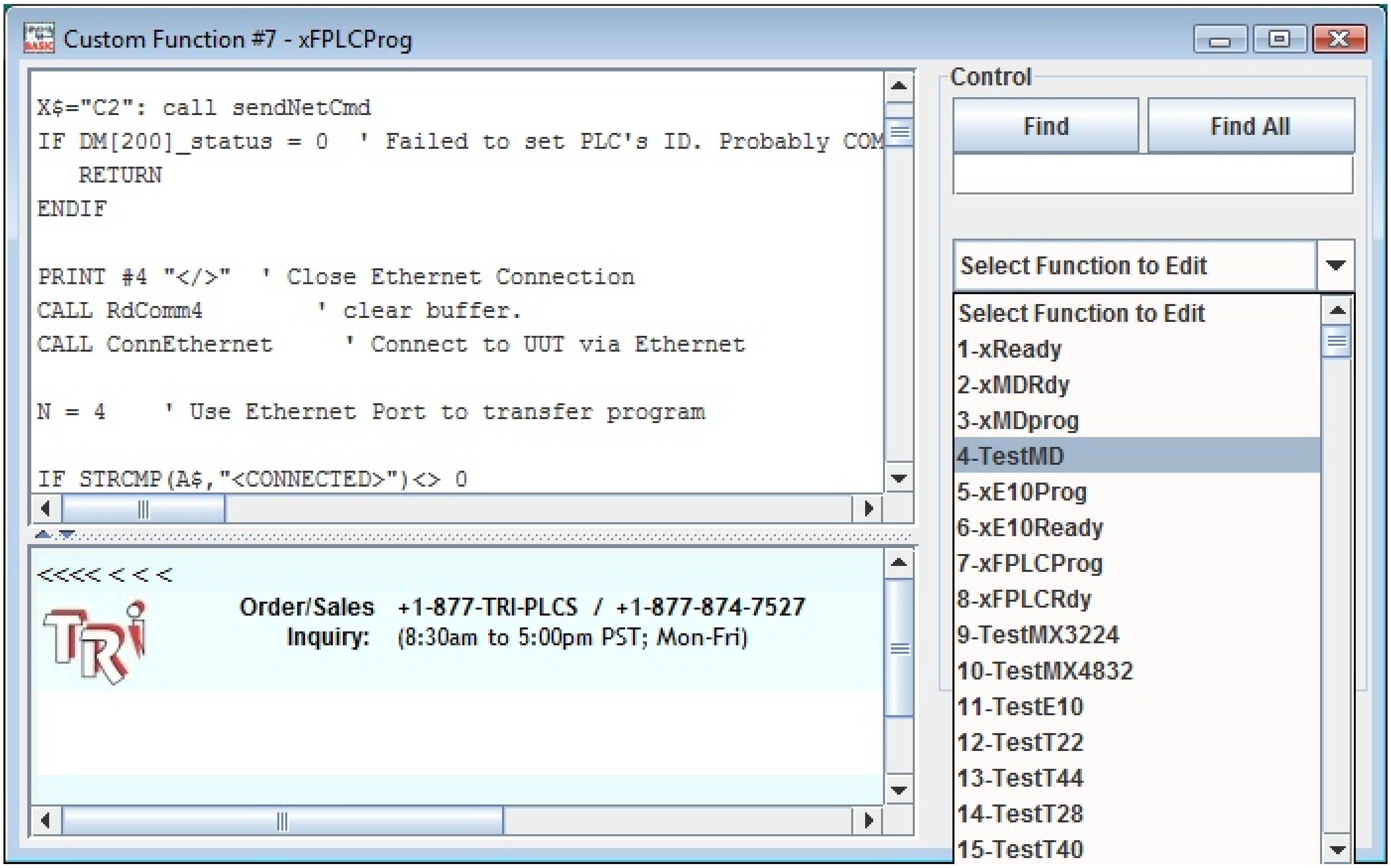
Find in All CusF

NOTE:

Highlighting of text found in the editor during a user initiated search or during compilation error tracking now works with JRE (Java Runtime Environment) 1.5 and 1.6

10.2.4 Navigating Custom Functions

It is now possible navigate to any other function directly by choosing the function name from the “Select Function To Edit” drop box, as shown below.



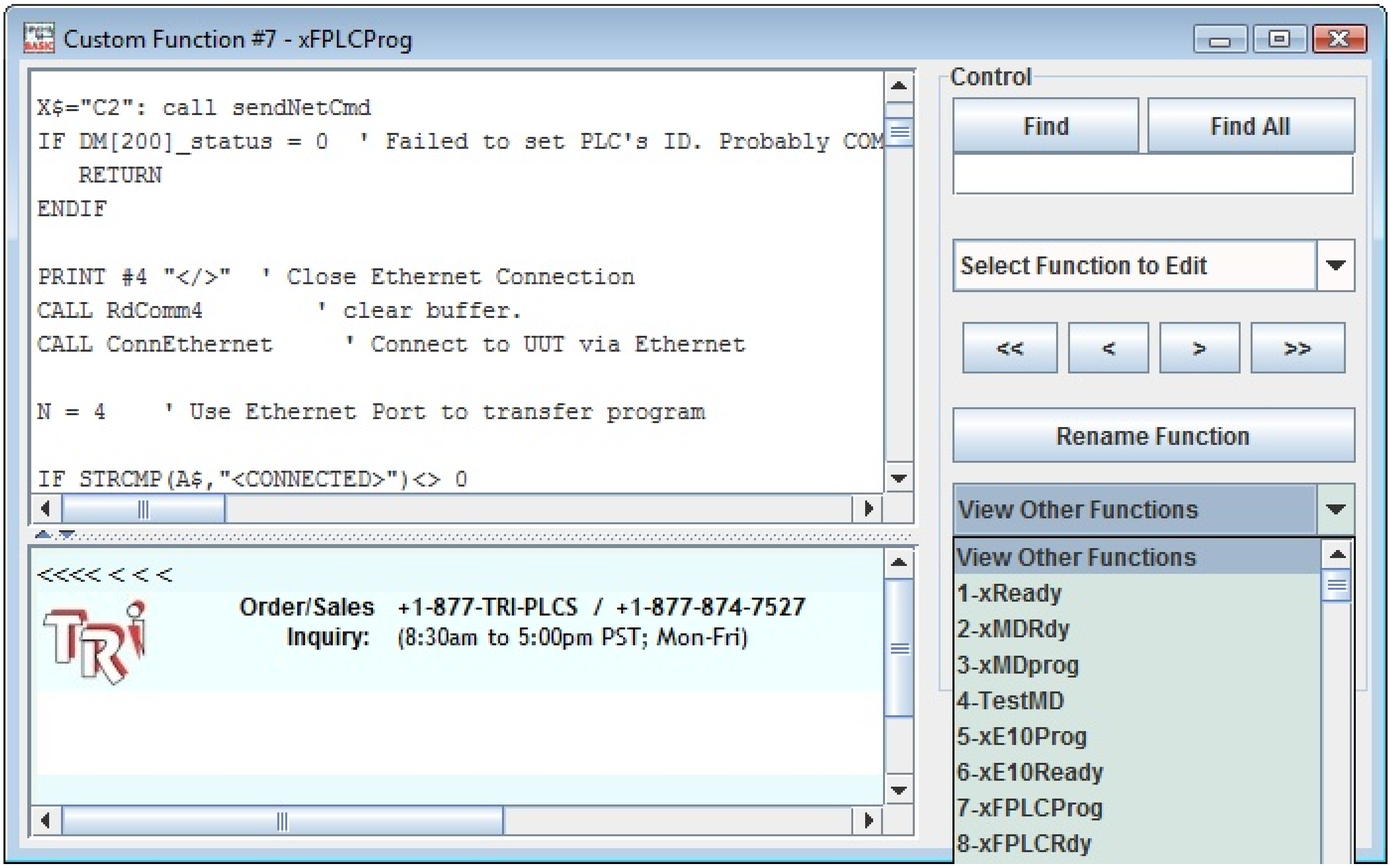
```
X$="C2": call sendNetCmd
IF DM[200]_status = 0  ' Failed to set PLC's ID. Probably COM
   RETURN
ENDIF

PRINT #4 "</>"  ' Close Ethernet Connection
CALL RdComm4        ' clear buffer.
CALL ConnEthernet     ' Connect to UUT via Ethernet

N = 4    ' Use Ethernet Port to transfer program

IF STRCMP(A$,"<CONNECTED>")<> 0
```

```
<<<< < < <
          Order/Sales   +1-877-TRI-PLCS  /  +1-877-874-7527
              Inquiry:  (8:30am to 5:00pm PST; Mon-Fri)
```

Alternatively, you can scroll from one custom function to the next one using the Fn Traverse2.jpg (1104 bytes) and Fn Traverse3.jpg (1113 bytes) keys. However, clicking on the Fn Traverse1.jpg (1199 bytes) and Fn Traverse4.jpg (1122 bytes) buttons allows you to scroll to the previous or the next non-empty CusFn. All empty functions will be skipped. This is useful if you need to browse through all the custom functions to locate something.

10.2.5 View Other Custom Functions

The "View Other Functions" choice box allows you to view another custom function via a read-only popup window without leaving the currently edited custom function. This is very convenient when you need to view the content of another custom function while editing one custom function.



```
X$="C2": call sendNetCmd
IF DM[200]_status = 0  ' Failed to set PLC's ID. Probably COM
   RETURN
ENDIF

PRINT #4 "</>"  ' Close Ethernet Connection
CALL RdComm4        ' clear buffer.
CALL ConnEthernet     ' Connect to UUT via Ethernet

N = 4    ' Use Ethernet Port to transfer program

IF STRCMP(A$,"<CONNECTED>")<> 0
```

<<<< < < <

Order/Sales +1-877-TRI-PLCS / +1-877-874-7527
Inquiry: (8:30am to 5:00pm PST; Mon-Fri)

10.2.6 Undo Changes

Undo

Click this button to undo any changes you just made to the custom function. You may also use the <Ctrl-z> shortcut key to achieve the same.

The “Undo” button (shown below) can be used to undo the last change to the text in the currently opened Custom Function.

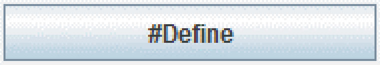
10.2.7 Aborting Changes

Abort

The “Abort” button can be used to abort any changes made to the currently opened Custom Function since it was opened. The custom function editor will be closed when you abort an edit.

Note: If you navigate away from the currently opened custom function, or if you exit a custom function by using the <ESC> key or by clicking the CusFn Editor-3.jpg (1142 bytes) button in the top-right corner of the editor, then the content of the custom function currently in the editor will be updated into the internal memory and you will not be able to undo or abort changes. Hence you can only abort changes to the currently opened custom function before you exit the editor or navigate away to another Custom Function.

10.2.8 #Define Table



In previous versions of i-TRiLOGI it was not possible to define variable names for the A-Z, DM[], A$-Z$, etc variables. However, in i-TRiLOGI versions 6.42 and up, a define button has been added to the custom function editor that allows users to create a table of variable and expression definitions for all integer/string variables available in TRiLOGI as well as constant values. The next section will describe in details how to use the #Define Table.

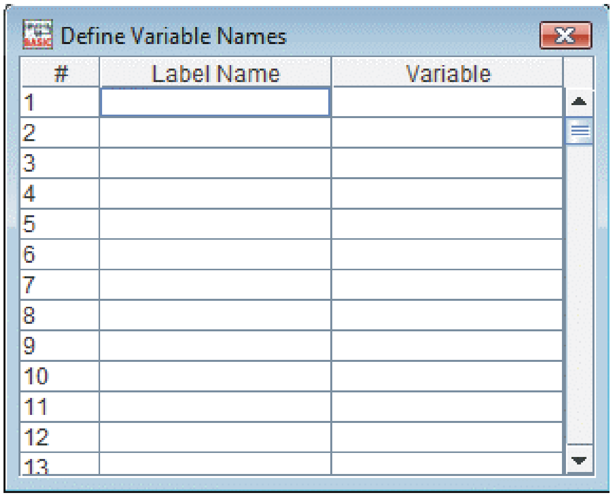
10.2.9 Toggle Breakpoint

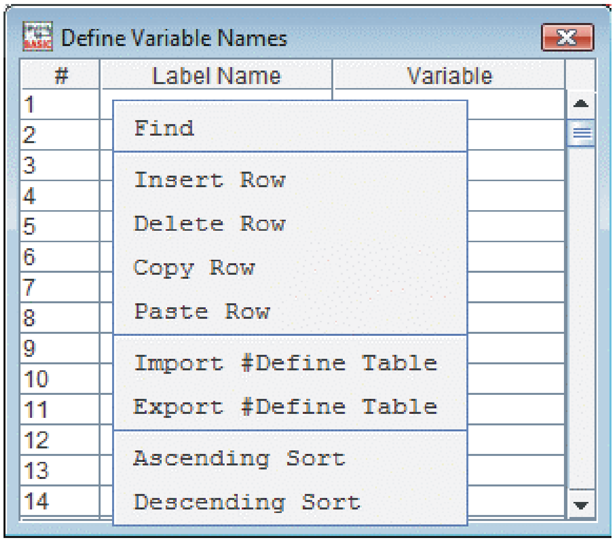
This new debugging aid feature is introduced since I-TRiLOGI version 6.45. For more information please refer to Section 7. Debugging & Breakpoints

10.3 Define Table

When the above #Define button is pressed a table of definitions will open as a new window, as shown by the picture on the left below. Only the Label Name and Variable columns can be edited such that the name you would like to use should go in the Label Name column and the name of the variable (eg. DM[1], A, A$, etc.) should go in the Variable column.

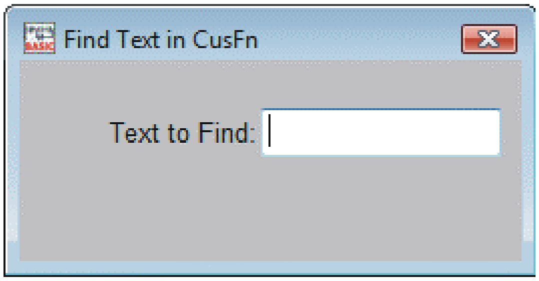
It is then possible to right-click on any of the cells to bring up the list of options shown by the picture on the right below.





10.3.1 Find

If you select the "Find" menu, you will see the following window popup and you will be able to type text in the field that will be searched for within the Define table.



10.3.2 Insert Row

A new blank row will be inserted above the row where the cell is highlighted when the mouse was right clicked. The highlighted row and all other rows below will be shifted down by 1 row and their row number will increase by 1.

10.3.3 Delete Row

The row that was highlighted when the mouse was right clicked will be deleted. All other rows below the highlighted row will be shifted up by 1 row and their row number will decrease by 1.

10.3.4 Copy Row

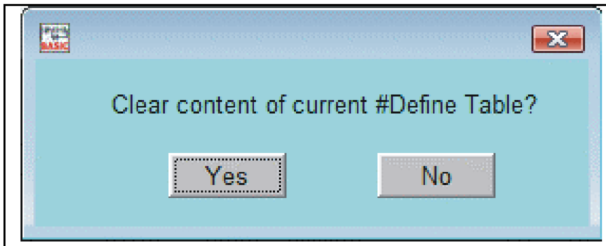
The row that was highlighted when the mouse was right clicked will be copied.

10.3.5 Paste Row

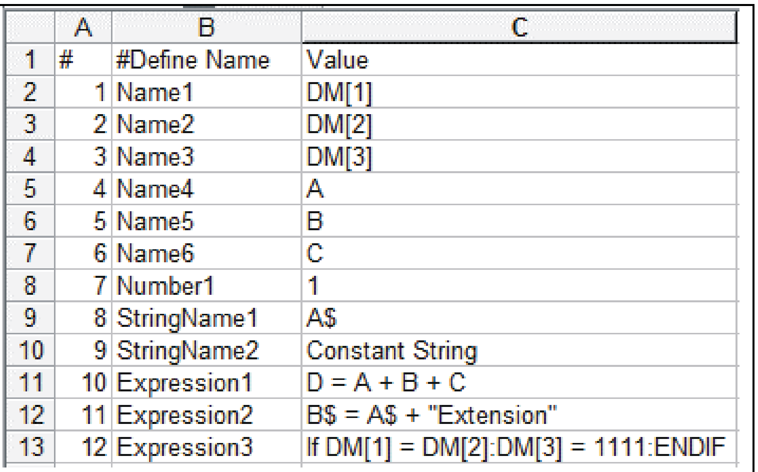
The row that was previously copied will be pasted to the row highlighted when the mouse was right clicked.

10.3.6 Import #Define Table

You will first be asked if you want to clear the content of the #Define Table and you will want to say 'Yes' to proceed with the import (say 'No' and do an export first if you want to save the current table).



Clear content of current #Define Table?



	A	B	C
1	#	#Define Name	Value
2	1	Name1	DM[1]
3	2	Name2	DM[2]
4	3	Name3	DM[3]
5	4	Name4	A
6	5	Name5	B
7	6	Name6	C
8	7	Number1	1
9	8	StringName1	A$
10	9	StringName2	Constant String
11	10	Expression1	D = A + B + C
12	11	Expression2	B$ = A$ + "Extension"
13	12	Expression3	If DM[1] = DM[2]:DM[3] = 1111:ENDIF

An example define table exported to an Excel file (automatically exported in "Tab Delimited" format) is shown above.

Note: row #1, which must be included exactly as shown above in order for the file to properly be imported into the define table.

You will then need to browse your PC for the .txt (Tab delimited text file) that you have previously exported or created and are now importing. Once you select it, the define table will be populated with all the definitions.

Note:

If you are have manually created a "#Define" table using Excel spreadsheet program, you will need to use the correct format in your spreadsheet (shown above) and ensure that the file is saved in the "Tab Delimited format" (not the standard spreadsheet format) with a ".txt" file extension.

10.3.7 Export #Define Table

You can export all the items in the #Define table of a currently opened .PC6 file to the PC harddisk so that the these definitions can be imported into another .PC6 program. When you select "Export #Define Table" you will just need to name and save the file with ".txt" extension to your desired location. The data are stored in TAB-delimited format, which can be opened directly by a Microsoft Excel program for viewing or editing.

10.3.8 More Information on #Define Tables

Please refer to the Integer Constants, Variables and Operators section or the String Constants, Variables and Operators section for more information on how to use the define table in your PLC program.

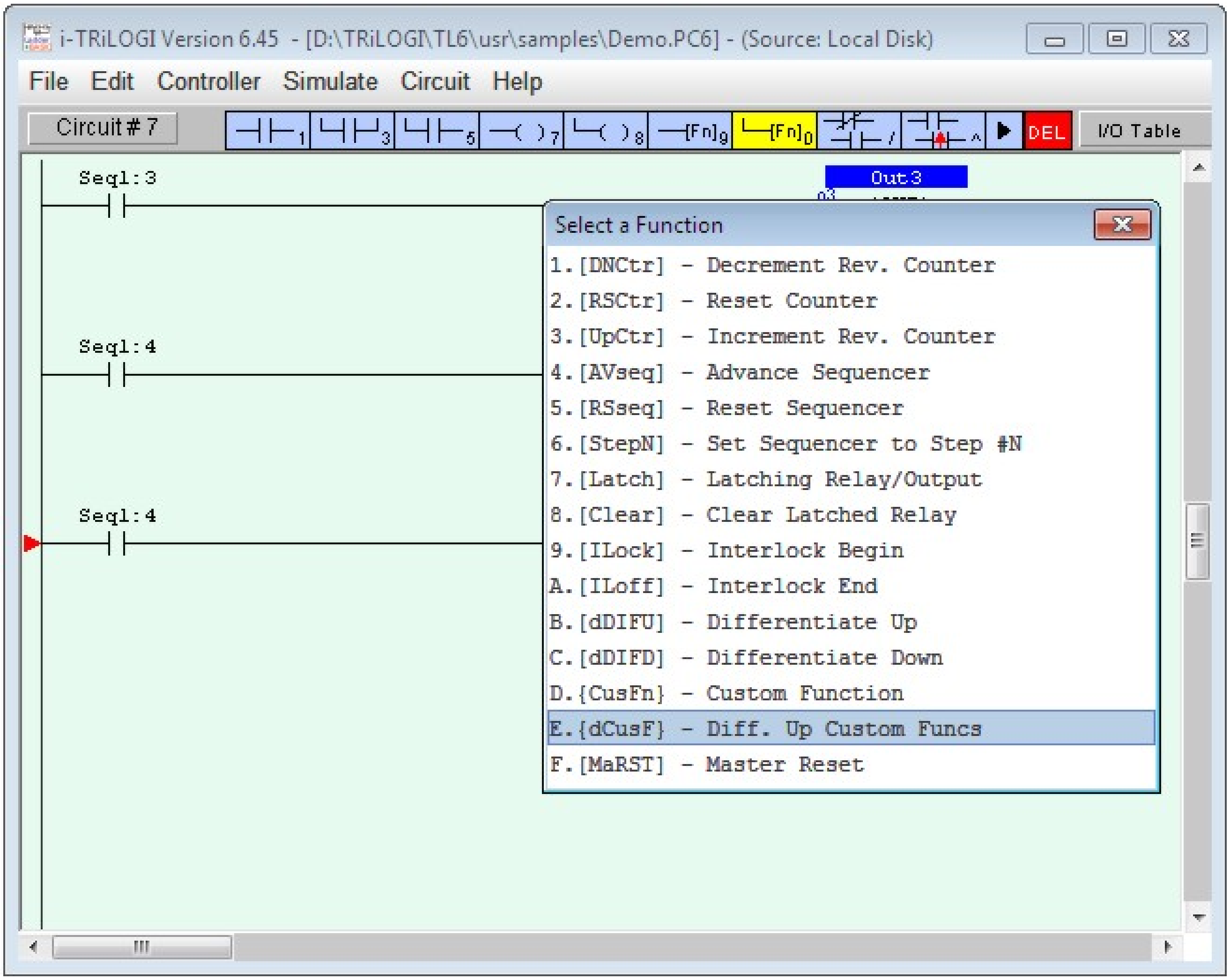
10.4 How to Use A Custom Function

It is important to understand when and how a TBASIC-based Custom Function is executed with respect to the rest of the program. There are basically two ways in which a CusFn will be executed:

10.4.1 Triggered by Ladder Logic Special function coil [CusFn]

A custom function may work the same way as any other special functions in the i-TRiLOGI ladder diagram programming environment. When you are in ladder circuit editing mode, press <Ins> key to open the "Ins Element" menu.

Select the item 9: ——[FUNC] or 0: └——[FUNC] to create a special function output. A pop-up "Select a Function" menu will appear.


i-TRiLOGI Version 6.45 - [D:\TRiLOGI\TL6\usr\samples\Demo.PC6] - (Source: Local Disk)
File Edit Controller Simulate Circuit Help
Circuit # 7
DEL
I/O Table
Seql:3
Seql:4
Seql:4
Out3
Select a Function
1.[DNCtr] - Decrement Rev. Counter
2.[RSCtr] - Reset Counter
3.[UpCtr] - Increment Rev. Counter
4.[AVseq] - Advance Sequencer
5.[RSseq] - Reset Sequencer
6.[StepN] - Set Sequencer to Step #N
7.[Latch] - Latching Relay/Output
8.[Clear] - Clear Latched Relay
9.[ILock] - Interlock Begin
A.[ILoff] - Interlock End
B.[dDIFU] - Differentiate Up
C.[dDIFD] - Differentiate Down
D.{CusFn} - Custom Function
E.{dCusF} - Diff. Up Custom Funcs
F.[MaRST] - Master Reset

Select either item to create a CusFn:

" D : [CusFn] - Custom created Function" or
" E : [dCusF] - Diff. Up Custom Functn"

You will be required to enter the selected custom function number from 1 to 256. Note that a CusFn created using

" E :Diff. Up Custom Functn[dCusF]"

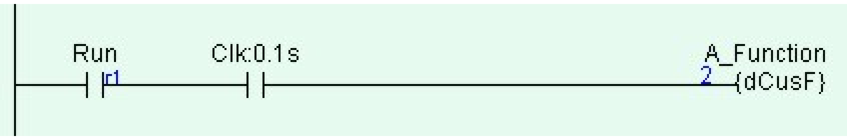
is a "Differentiated Up" instruction. This means that the function will be executed only once every time when its execution condition goes from OFF to ON. Nothing will happen when its execution condition goes from ON to OFF.

On the other hand, using "D: Custom created Function [CusFn]" will mean that the CusFn will be executed every scan as long as its execution condition is ON.

This is often not desirable and the coil created using this menu item will be highlighted in RED color to serve as an alarm to programmer. You will probably find that you will use the differentiated version [dCusF] far more frequently.

10.4.2 Periodic Execution of a Custom Function

There are many situations when you need the PLC to periodically monitor an event or perform an operation. For example, to monitor the temperature reading from a probe or check the real time clock for the scheduled time, and to continuously display changing variables on the LCD display. It is not efficient to use the continuous [CusFn] function for such purposes. It is far better to use the built-in clock pulses to trigger a differentiated Custom function [dCusF]. You can choose a suitable period from 0.01s, 0.02s, 0.05s, 0.1s, 0.2s, 0.5s, 1.0s and 1 minute for the application. Other periods can also be constructed with a self-reset timer. The custom function will only be executed once every period controlled by the system clock pulse or the timer, as follow:



For example, you don't need to update the value of a variable displayed on the LCD screen any faster than the human eye can read them. So using a 0.5s clock pulse may be sufficient and this will not take up too much CPU time for the display. For slow processes such as heating, a 1.0s clock pulse to monitor temperature change is more than sufficient.

IMPORTANT

1. When the CPU scans the ladder logic to a circuit which contains a CusFn, and the execution condition of the circuit is TRUE, the corresponding CusFn will be immediately executed. This means that the CPU will not execute the remaining ladder circuits until it has completed execution of the current CusFn. Hence if the CusFn modifies a certain I/O or variable, it is possible to affect the running of the remaining ladder program.

2. Note that the INPUT[n] variables contain data obtained at the beginning of the ladder logic scan and not the actual state of the physical input at the time of the CusFn execution. Thus, it will be futile to wait for the INPUT[n] variable to change inside a CusFn unless you execute the REFRESH statement to refresh the physical I/O before you examine the INPUT[n] variable again.

3. Likewise, any changes to the OUTPUT[n] variable using the SETBIT or CLRBIT statement will not be transferred to the physical outputs until the end of the current ladder logic scan. Hence do not wait for an event to happen immediately after executing a SETBIT or CLRBIT statement on an OUTPUT[n] because nothing will happen to the physical output until the current ladder logic scan is completed.

4. If you want to force the output to change immediately you will need to execute the REFRESH statement. Consideration must be given to how such an act may affect the other parts of the ladder program since not the entire ladder program has been executed.

5. Like all ladder circuits, the relative position of the circuit which triggers the CusFn may affect the way the program works. It is important to consider this fact carefully when writing your ladder program and TBASIC CusFns. Always remember that the CPU executes the ladder logic and CusFn sequentially, even though the equivalent circuits in hard-wired relay may seem to suggest that the different rungs of ladder circuits were to work simultaneously.

6. In line with the typical Ladder Logic programming rules, a CusFn may appear only once within the ladder diagram, regardless of whether it appears in the normal or differentiated form. A compilation error will occur if a CusFn appears in more than one circuit.

7. However, a CusFn may be "CALLed" as a subroutine by any other CusFn and there is no restriction placed on the number of repeated CALL of a CusFn by more than one CusFn. A CusFn may also modify the logic states of an I/O element or the value of internal timers and counters using its powerful TBASIC commands (such as SetBit, ClrBit). The compiler however will not alarm the user that a CusFn may inadvertently alter the logic state of an I/O already controlled by some other ladder circuit.

This power and flexibility offered by the TBASIC-based custom functions must therefore be handled with greater care by the programmer. It is important to prevent conflicting output conditions due to an I/O being controlled or modified at more than one place within a logic scan. The net result is that the logic state of the I/O appears to be in different states at different parts of the ladder circuit. This could lead to bizarre outcomes that may be difficult to trace and debug.

10.4.3 Interrupt Service CusFn

A CusFn may also serve as an "Interrupt Service Routine" which is executed asynchronously from the normal ladder logic execution. An interrupt-driven

CusFn is run when the condition which causes the interrupt occurs. The response time to execution is very short compared to the scan time of the ladder program. There are several interrupt sources which can trigger a CusFn:

a) Physical Interrupt inputs

The NANO-10/FMD series/F-Series PLCs contain some hardware "Interrupt" inputs which, when enabled by the INTRDEF statement, will trigger a particular CusFn defined in the INTRDEF statement when the logic level at the interrupt pin changes state (either from OFF to ON or from ON to OFF).

b) Periodic Timer Interrupt (PTI)

The Periodic Timer Interrupt (PTI) lets you define a custom function that will be executed by the CPU precisely every x number of milliseconds (ms). The PTI runs independently of the ladder logic and its execution is therefore not affected by the total PLC program scan time.

NOTE: The PTI is not available on the legacy T100MD/T100MX PLC

c) Power Failure Interrupt (PFI)

The NANO-10/FMD series/F-Series PLCs CPU has a built-in power failure sensing circuit that will call a custom function when it detects an impending power failure. This allows you to perform such critical function such as saving critical data to the PLC's non-volatile memory just before power failure.

NOTE: The PFI is not available on the legacy T100MD/T100MX PLC

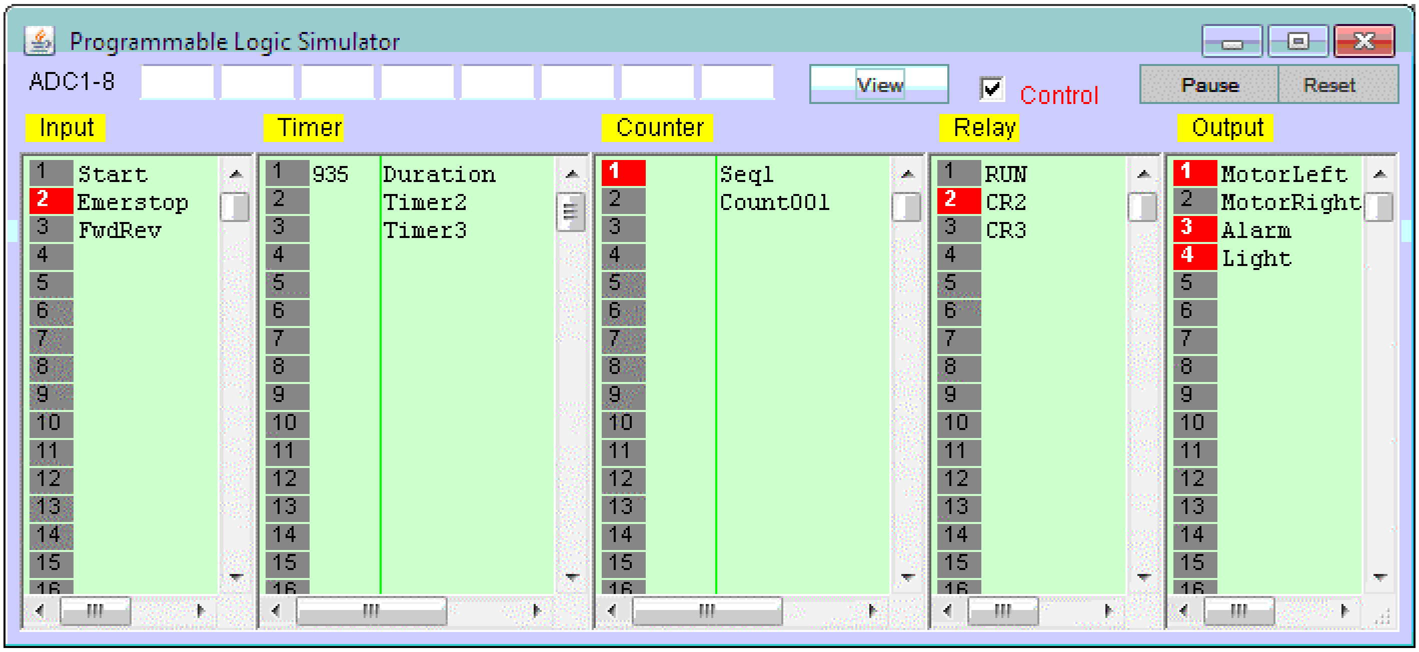
d) High Speed Counters (HSC) Reach Target Count

These PLCs also contain some "High Speed Counter" inputs which, when enabled by the HSCDEF statement, will trigger a particular CusFn defined in the HSCDEF statement when the counter reaches a preset target count value. This enables the CPU to carry out immediate action such as stopping a motor or performing some computation.

10.5 Simulation & Examination of TBASIC Variables

10.5.1 Simulation Run of CusFn.

i-TRiLOGI fully supports simulation of all TBASIC commands. After you have completed coding a CusFn, test the effect of the function by connecting it to an unused input. Run the simulator by pressing <F9> or <Ctrl-F9> key. Execute the CusFn by turning ON its control input. If your CusFn executes a command that affects the logic state of any I/O, the effect can be viewed on the simulator screen immediately. However, if the computation affects only the variables, than you may need to examine the internal variables.



An I/O or internal relay bit that has been turned ON is indicated by a RED color rectangular lamp that simulate a LED being turned ON. You can pause the logic simulator at any time by pressing the <P> key or clicking on the [Pause] button. Likewise the simulator engine can be reset by clicking on the [Reset] button.

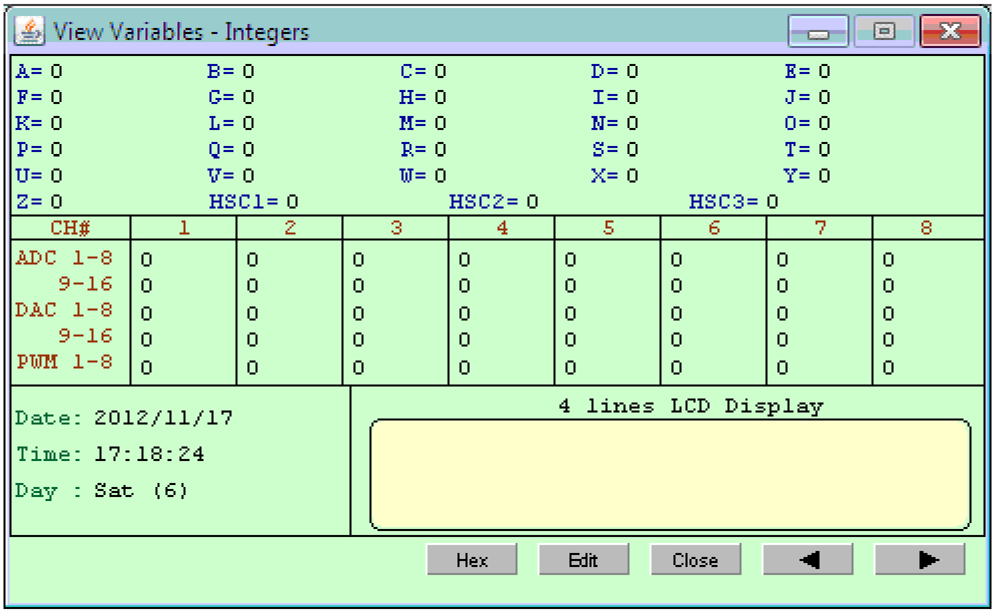
Simulation of ADC Inputs

Along the top edge of the Programmable Logic Simulator screen, you will find 8 text fields adjacent to the label "ADC1-8". The programmer can enter the expected ADC values for ADC#1 to #8 in these text fields. In effect, these simulate the potential signal strength at their respective ADC input pins. These values will be captured by the TBASIC program when an ADC(n) command is executed in a custom function for ADC #n.

Note: values entered at the ADC input text field will only be updated when the user press the <Enter> key or the <TAB> key to ensure that only finalized entries are used by the TBASIC program. (otherwise, imagine if you try to enter the value 123 at ADC #1, the program would first be receiving "1", then "12" and then "123" which was not the intention).

10.5.2 Viewing TBASIC Variables

The values of the internal variables as a result of the simulation run can be viewed by pressing the <V> (which stand for "View") key or by clicking on the [View] button while in the simulation screen. A pop-up window will appear with the values of all the variables as well as special peripheral devices supported by TBASIC. The variables are organized into 4 screens. You can move from screen to screen using the left/right cursor keys or by clicking on the navigation buttons:



A= 0 B= 0 C= 0 D= 0 E= 0
F= 0 G= 0 H= 0 I= 0 J= 0
K= 0 L= 0 M= 0 N= 0 O= 0
P= 0 Q= 0 R= 0 S= 0 T= 0
U= 0 V= 0 W= 0 X= 0 Y= 0
Z= 0 HSC1= 0 HSC2= 0 HSC3= 0

CH#	1	2	3	4	5	6	7	8
ADC 1-8	0	0	0	0	0	0	0	0
9-16	0	0	0	0	0	0	0	0
DAC 1-8	0	0	0	0	0	0	0	0
9-16	0	0	0	0	0	0	0	0
PWM 1-8	0	0	0	0	0	0	0	0

Date: 2012/11/17
Time: 17:18:24
Day : Sat (6)

a) Integer variables Screen

The first screen comprises all 26 32-bit integer variables A-Z, the system DATE and TIME, ADC, DAC, PWM and the resulting values of setLCD commands. The initial DATE and TIME figures shown during simulation are taken from the PC's internal real-time clock values. However, subsequent values can be affected by the values assigned to the variable DATE[n] and TIME[n].

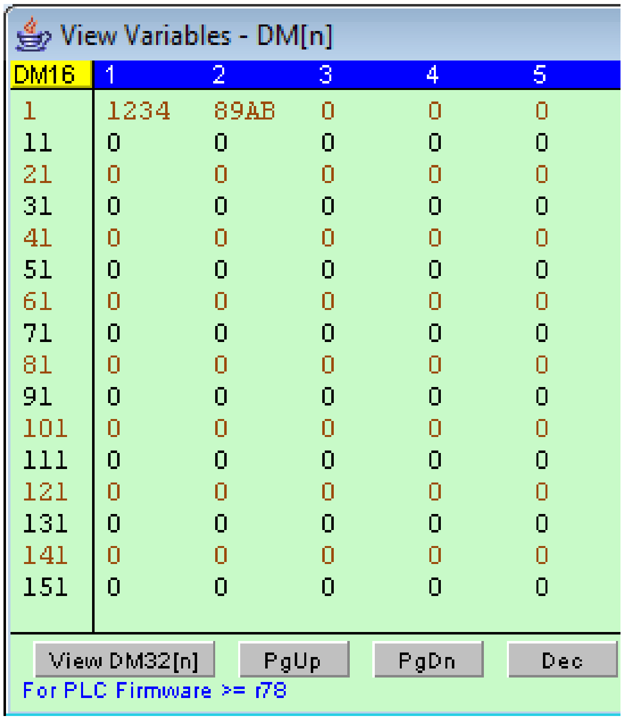
The present values of the first 3 high speed counters: HSC1 to HSC3 are also shown on this page. Note that ADC data for any particular A/D channel #n will only be shown if an ADC(n) function has been executed. Otherwise the ADC value shown on screen will not reflect the true current value of the ADC port.

b) Data Memory Screen

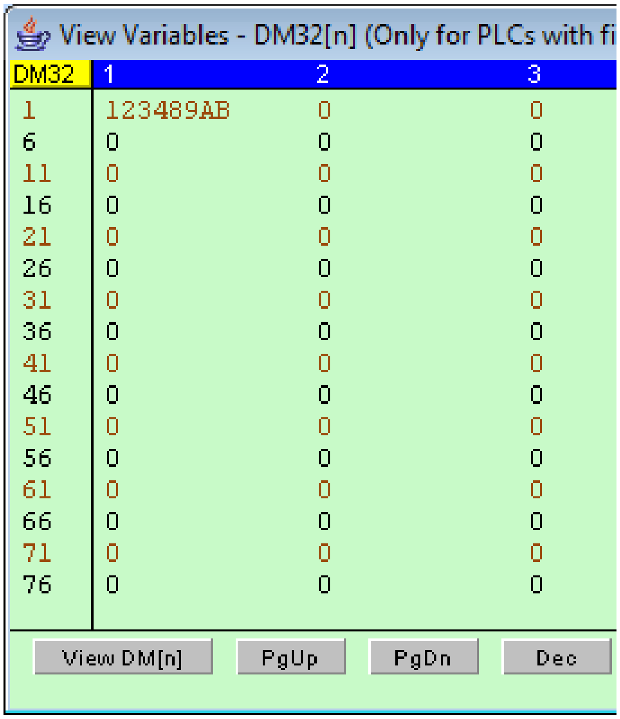
The second screen displays, in 25 pages, the values of the 16-bit DM variables from DM[1] to DM[4000]. Each page displays 16 rows x 10 columns = 160 DM variables. You can scroll up and down the pages by clicking on the [PgUp] or [PgDn] buttons or using the corresponding keys on the keyboard.

Starting from version 6.45 you can use two consecutive 16-bit DM variables as a single 32-bit DM32 variable in the following manner: DM[1] & DM[2] = DM32[1]; DM[3] & DM[4] = DM32[2].... DM[2n-1] & DM[2n] = DM32[n]. A new button "View DM32[n]" is added and when clicked, toggles the view of the DM variables between DM and DM32, as shown below. The range of DM[1] to DM[4000] are mapped to DM32[1] to DM32[2000].

Note:
Although this version of TRiLOGI allows you to view any super PLC's DM[] as DM32[] during online monitoring, only new Super PLCs with firmware version r78 or later is able to actually use DM32[n] in its program. i-TRiLOGI will check the PLC firmware version number during program transfer and will not transfer TBASIC program that contains DM32[] variables to those PLC with < r78 firmware.



DM16	1	2	3	4	5
1	1234	89AB	0	0	0
11	0	0	0	0	0
21	0	0	0	0	0
31	0	0	0	0	0
41	0	0	0	0	0
51	0	0	0	0	0
61	0	0	0	0	0
71	0	0	0	0	0
81	0	0	0	0	0
91	0	0	0	0	0
101	0	0	0	0	0
111	0	0	0	0	0
121	0	0	0	0	0
131	0	0	0	0	0
141	0	0	0	0	0
151	0	0	0	0	0



DM32	1	2	3
1	123489AB	0	0
6	0	0	0
11	0	0	0
16	0	0	0
21	0	0	0
26	0	0	0
31	0	0	0
36	0	0	0
41	0	0	0
46	0	0	0
51	0	0	0
56	0	0	0
61	0	0	0
66	0	0	0
71	0	0	0
76	0	0	0

c) String Variable Screen

The third screen displays the value of the 26 string variables A$ to Z$ in 4 pages, depending on the length of each string. If the execution condition is ON and the

CusFn is not of the differentiated type, then the CusFn will be continuously executed. The result of the variable will be continuously updated on the viewing window.

d) <u>System Variable Screen</u>

System variables such as INPUT[n] , RELAY[n] and emINT[n] are visible in this screen. You may wish to click on the [Hex] button to view the values in hexadecimal notation as they are more commonly used by programmers to identify the bit patterns in these variables.

10.5.3 Changing the Contents of Variables

While the "View Special Variables" window is open, you may change the contents of the following variables by clicking on the [Edit] button:

A-Z, A$ to Z$, DM[n], DM32[n], DATE[n], TIME[n], INPUT[n], OUTPUT[n], RELAY[n], TIMERBIT[n], CTRBIT[n], TIMERPV[n], CTRPV[n] and HSCPV[n], emINT[n], emLINT[n].

A text entry window will pop up and you will have to enter the values in the form of assignment statements, such as:

```
A = 5000;
DM[99]=5678;
DM32[100]=&H12789ABC
OUTPUT[2]=&H01AB
B$ = "Welcome to TBASIC"
```

The variable will take up the new value as soon as it is entered, and if the execution condition for any CusFn is ON, the simulator will process the newly entered data immediately and produce the new outcomes. This gives you greater flexibility in controlling the simulation process.

10.5.4 Decimal and Hexadecimal Representation

All the numeric data shown in the "Special Variables" window are by default displayed in decimal notation. You can display the number in hexadecimal format by clicking on the [Hex] button or by pressing the <H> key. Press the <D> key if you wish to switch back to the decimal format. This feature is very useful for programmers who are familiar with hexadecimal representation of a binary number. The [Hex] button will become the [Dec] button when you enter the Hex display mode.

10.6 On-line Monitoring of TBASIC Variables

If you execute the "On-Line Monitoring/Control" command from the "Controller" pull-down menu, i-TRiLOGI will continuously query the PLC for the values of all their internal variables. These variables' values will be updated in real time in the "View Special Variables" window as described in Section 5.2. You may also alter the value of any variables in the PLC using the "Edit Variable" window (by clicking on the "Edit" button at the "View xxx Variables" window.

This ability of i-TRiLOGI to provide instant and full visibility of all the PLC's internal variables greatly facilitates the programmers' debugging process. The ease of programming offered by the i-TRiLOGI programming environment is really what really sets the Nano-10, FMD series, and F-Series PLCs apart from many other PLCs.

10.6.1 PAUSE and RESET of Target PLC

During On-Line Monitoring, if the "View Special Variables" window is opened, you can still reset the PLC's internal data by pressing the <Ctrl-R> key. The PLC can also be halted by pressing the <P> key. A halted PLC can subsequently be released from the halted mode by pressing the <P> key again. You can also change any internal variable data using the "Edit" button on the View Variable screen.

10.6.2 Using LCD Display for Debugging

You should take advantage of the built-in LCD display port of the PLC to display internal data at the location where you want to track their values, especially if the value changes rapidly which may not be constantly captured by on-line monitoring screen.

10.7 Debugging With PAUSE & Breakpoints

10.7.1 Using the TBASIC PAUSE Statement

When debugging your program it is often easier to identify program bugs by pausing the PLC at a certain execution point so that you can examine the data in the variables to understand why the program does not behave the way you expect it to. TBASIC offers the "PAUSE" statement which is always available in all versions of i-TRiLOGI software and supported on all firmware versions of the Super PLCs.

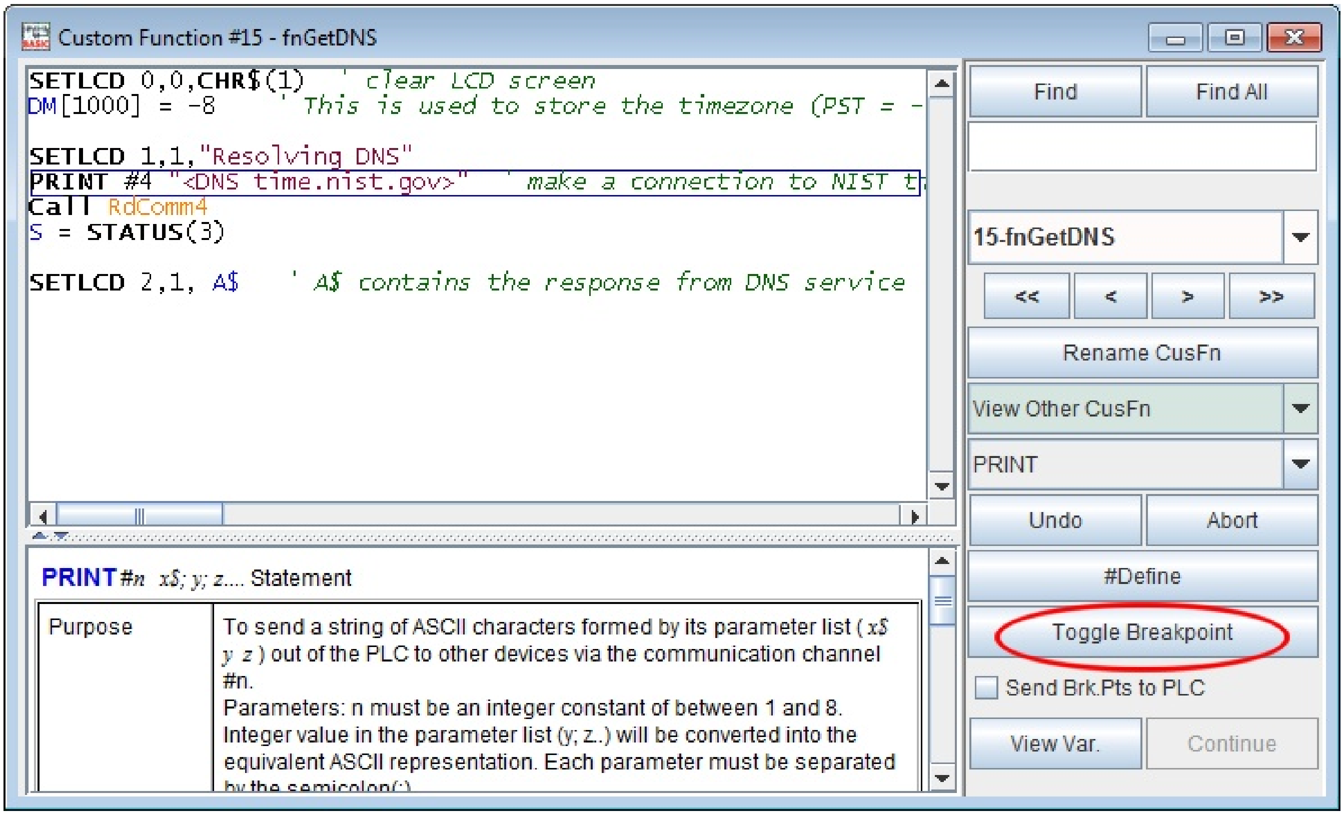
When the program reaches a PAUSE statement it will enter a PAUSE Mode and you can then examine the variable in the PLC. You can add as many PAUSE statement as you need in different part of the program to halt the program execution. However, since the software does not show you exactly where the program is halted except to tell you that a PAUSE statement is known to have been executed in a particular function number, and if you have more than 1 PAUSE statement in a particular custom function then you will need to put some markers so that you know where the program has stopped. The marker could be displaying of some text on the LCD (e.g. "Pause @21") that a certain PAUSE point has been reached or you can assign a certain value to a reserved variable and the value corresponds to the location of your PAUSE statement. This unfortunately means that you need to modify your program in order to break the PLC program at a certain location.

Although it may be only a small effort to add and delete PAUSE statements in the TBASIC program and then re-run the test immediately during simulation, if you don't use the simulator but instead debug your program directly on the PLC, then you must re-transfer the program to the PLC every time you add or delete a PAUSE statement. This can be time consuming if your program is large or you have a very long custom function. So the new BREAKPOINT features described in the next section can be good a time saver.

10.7.2 Using Breakpoints

Starting from i-TRiLOGI version 6.45 a new "Breakpoint" feature has been added to the program so that debugging can be more easily performed without the need to modify and transfer the program to the PLC. The Breakpoint feature is supported on all Super PLCs with firmware version r78 and above. The break point feature is always available to the simulator in this software version but the program does not let you send the program break points to a PLC with firmware older than r78.

To set a break point in any part of the TBASIC program, open up the custom function and place the cursor on the line where you want the program to pause when it reach the line. Then click on the "Toggle Breakpoint" button to set the break point. A breakpoint that is set will be shown up as a blue rectangular box around the line, as shown below:



```
SETLCD 0,0,CHR$(1)    ' clear LCD screen
DM[1000] = -8      ' This is used to store the timezone (PST = -

SETLCD 1,1,"Resolving DNS"
PRINT #4 "<DNS time.nist.gov>"    ' make a connection to NIST t
Call RdComm4
S = STATUS(3)

SETLCD 2,1, A$    ' A$ contains the response from DNS service
```

PRINT #*n* *x$; y; z*.... Statement

Purpose	To send a string of ASCII characters formed by its parameter list (*x$ y z*) out of the PLC to other devices via the communication channel #n. Parameters: n must be an integer constant of between 1 and 8. Integer value in the parameter list (y; z..) will be converted into the equivalent ASCII representation. Each parameter must be separated by the semicolon(;)

If you click on the "Toggle Breakpoint" button again the blue box will disappear, which means the breakpoint has been cleared.

Note:

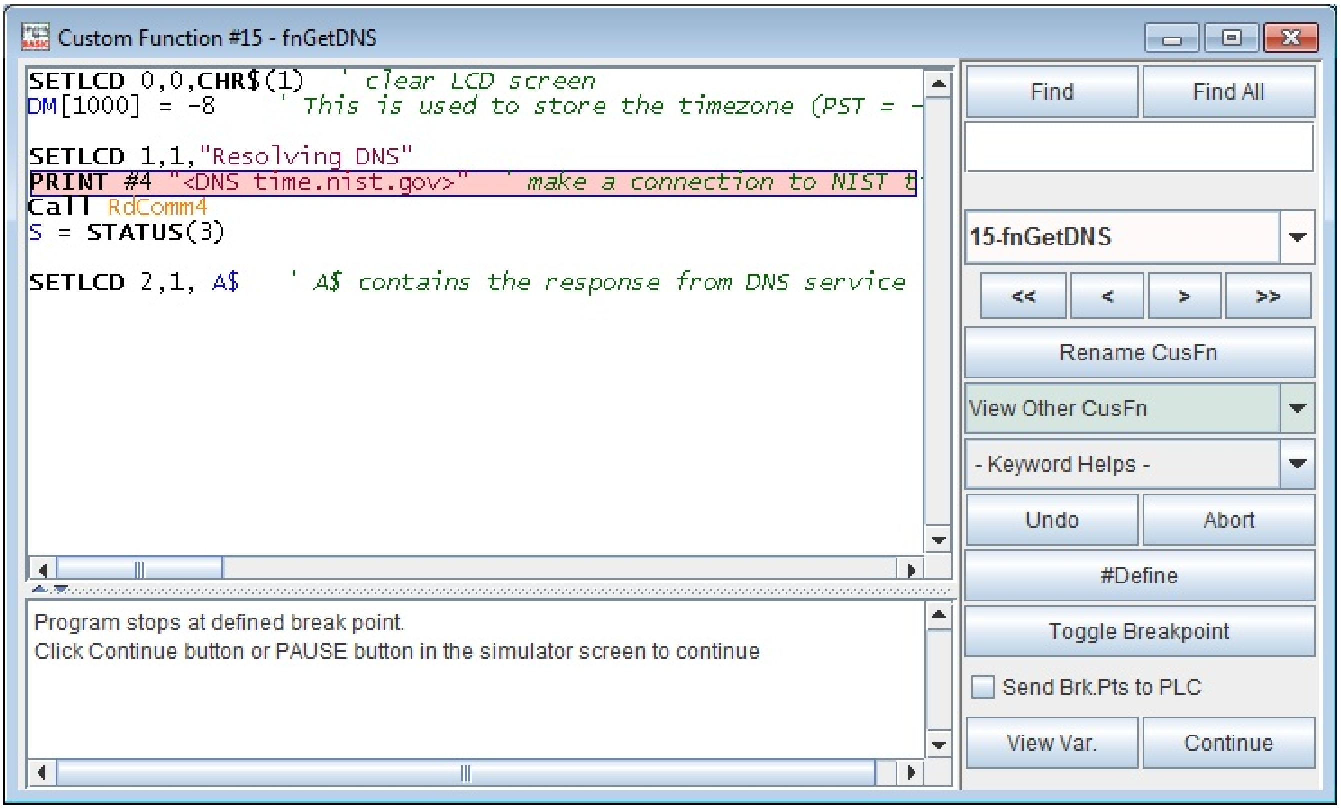
1. You can define up to a maximum of 8 breakpoints in a program.

2. Comment/Remark lines and blank lines are not compiled and breakpoint should not be set on any of these lines.

3. The breakpoints that you've defined are saved along with the program. Each breakpoint can be cleared individually by placing the cursor on the breakpoint line and the click the "Toggle Breakpoint" button. If you wish to clear all breakpoints in the program you can click on the "Edit" pull down menu and select "Clear All Breakpoints".

10.7.3 Debugging Using Breakpoint on Simulator

With a break point set in the custom function and if you run the program using the simulator, when the simulator runs to the line where a breakpoint is set the program will pause and the custom function where the breakpoint is defined will be opened with the breakpoint line highlighted as shown in the following picture:



```
SETLCD 0,0,CHR$(1)   ' clear LCD screen
DM[1000] = -8      ' This is used to store the timezone (PST = -

SETLCD 1,1,"Resolving DNS"
PRINT #4 "<DNS time.nist.gov>"   ' make a connection to NIST t
Call RdComm4
S = STATUS(3)

SETLCD 2,1, A$    ' A$ contains the response from DNS service
```

Program stops at defined break point.
Click Continue button or PAUSE button in the simulator screen to continue

At this point you can examine the variable data by clicking on the "View Var." button, which will open up the "View Variable" screen. You can release the program from its PAUSE state by either clicking on the "Continue" button or by clicking on the "Pause" button on the on-line monitoring screen. The program will continue execution until it hits the next breakpoint. Note that when the program is halted at a breakpoint you can define more breakpoints so as to track the program execution. You can remove old breakpoint at any time if you run out of the maximum 8 breakpoints limit.

10.7.4 Debugging Using Breakpoints on PLC

Notice the ☐ Send Brk.Pts to PLC checkbox below the "Toggle Breakpoint" button? If you wish to halt the actual PLC (not the simulator) when the program runs to the breakpoint, you can send all your defined breakpoints to the PLC by clicking on this checkbox. i-TRiLOGI will connect to the PLC and check if the program in the PLC is the same as the currently open program by verifying the checksums. If the program is current then it will transfer all the defined break point to the PLC.

With the checkbox shown as "checked" any subsequent addition or deletion of breakpoints will be immediately transferred to the PLC

When the PLC program reaches the defined breakpoint it will stop execution and the PAUSE LED on the PLC will light up. If you then go online monitoring the program will open up the custom function and highlight the breakpoint line where the breakpoint was encountered.

Note that the breakpoint sent to the PLC is volatile. If you reboot or power-on reset the PLC all the breakpoints will be disabled. This ensure that production PLC will not halt at any breakpoint that you forget to clear.

To manually clear all the breakpoints, you can un-check the checkbox and the i-TRiLOGI program will disable the breakpoints in the PLC. If the checkbox was already unchecked but for some reasons the PLC breakpoints have not been cleared yet, then you will need to first check the checkbox to transfer the breakpoints to the PLC and then followed with an uncheck to disable them.

If a PLC stops at a breakpoint that is not defined in your current file then the i-TRiLOGI program will give you an alert and will not be able to display the line where the program break.

10.8 Error Handling

Since the CusFn text editor does not restrict the type of text that may be entered into its editor, the i-TRiLOGI compiler will have to check the syntax of the user’s TBASIC program to look out for mis-spelling, missing parameters, invalid commands, etc. Such errors which can be tracked down during compilation process are know as "Syntax Errors".

10.8.1 Syntax Error

i-TRiLOGI employs a sophisticated yet extremely user-friendly syntax error tracking system: When a syntax error is encountered, the compilation will be aborted immediately and the CusFn which contains the error is automatically opened in the text editor. The location of the offending word is also highlighted and a pop-up message window reports to you the cause of the error. You can then immediately fix the error and re-compile until all the errors have been corrected.

Error Message	**Cause / Action**
Undefined symbol found	Only TBASIC commands and legal variable names are allowed. See Chapter 3.
Compiler internal error	Serious trouble, please email to the manufacturer support@tri-plc.com to inform us.
") " found without matching " ("	-
Integer expected	Expect to see either an integer variable or integer constant.
Value is out-of-range	Check the language reference for allowable range of values for the command.
Duplicate line label number	Label for goto must be unique within the same CusFn.
Undefined GOTO destination:	Put a matching label at the place where the GOTO statement is supposed to go.
Invalid GOTO label	@# must be in the range 0-255
Type mismatch (numeric and string types may not mix)	In an expression, strings and integers may not be mixed unless converted using the conversion function. e.g. STR$, VAL, etc.
String is too long	A string is limited to 70 characters
Too many line labels	There should not be more than 20 GOTO labels within the same CusFn.
Unknown Keyword	Most likely wrong spelling for TBASIC statement or function.
WHILE without ENDWHILE	Every WHILE statement must be ended with a matching ENDWHILE statement. Nested WHILE loop must have proper matching ENDWHILE for each WHILE.
IF without ENDIF	Every IF statement must be ended with a matching ENDIF statement to define the boundaries for the block controlled by the IF statement. For multiple IF THEN statement, each IF must be matched by a corresponding ENDIF.
FOR without NEXT	Every FOR statement must be ended with a matching NEXT statement to define the boundaries for the block controlled by the FOR statement. For nested FOR loops, each FOR must be matched by a corresponding NEXT.
Expect keyword "TO"	Required by FOR statement.

Must be an integer	String variable or constant not allowed.
Must be an integer variable only	Integer constant not allowed.
Must be an integer constant only	Integer variable not allowed.
Must be a string	Integer constant or variable not allowed.
Must be a string variable only	String constant not allowed.
Must be a string constant only	String variable not allowed.
Incomplete Expression	Expression not ended properly.
String constant missing closing "	String constants must be enclosed between a pair of opening and closing quotation character (")
Must be Integer A to Z only	index for FOR..NEXT loop must be A-Z.

10.8.2 Run-Time Errors

Certain errors only become apparent during the execution of the program, e.g. A = B/C . This expression is perfectly OK except when C = 0, then you would have attempted to divide a number by zero, which does not yield any meaningful result. In this case a "run-time error" is said to have occurred. Since run-time errors cannot be identified during compilation, i-TRiLOGI also checks the validity of a command during simulation run and if a run-time error is encountered, a pop-up message window will report to the programmer the cause and the CusFn where the run-time error took place. This helps the programmer locate the cause of the run-time errors to enable debugging. The possible run-time errors are listed in the following table and they are generally self-explanatory.

Run-Time Error Message
Divide by zero
Call stack overflow! Circular CALL suspected!
FOR-NEXT loop with STEP = 0!
SET_BIT position out-of-range!
CLR_BIT position out-of-range!
TEST_BIT position out-of-range!
STEPSPEED channel out-of-range!
Illegal Pulse Rate for STEPMOVE!
Illegal acceleration for STEPMOVE!
STEPMOVE channel out-of-range!
STEPSTOP channel out-of-range!

ADC channel out-of-range
DAC channel out-of-range
LED Digit # within (1-12) Only!
PWM Channel out-of-range!
LCD Line # must be (1-4) Only!
PM channel out-of-range!
System Variable Index Out-of-range!
Shifting of (A-Z) Out-of-range!
Illegal Opcode - Please Inform Manufacturer!
Timer or Counter # Out-of-Range!

Chapter 11 TBASIC Statements, Functions, Operators and Variables

11 TBASIC STATEMENTS, FUNCTIONS, OPERATORS AND VARIABLES

11.1 What are TBASIC Statement and Functions?

11.1.1 STATEMENT

A STATEMENT is a group of keywords used by TBASIC to perform certain action. A statement may take 0,1,2 or more arguments. The following are some TBASIC statements: PRINT, LET, IF, WHILE, SETLED ...etc.

11.1.2 FUNCTION

A FUNCTION acts on its supplied arguments and return a value. The returned value may be an integer or a string. A function can usually be embedded within an expression as if it is a variable or a constant, since its content will be evaluated before being used in the expression. e.g.

```
A$ = "Total is $"+STR$(B+C)
```

STR$(n) is a function which returns a string and therefore can be used directly in the above string assignment statement.

The most distinguishable feature of a FUNCTION is that its arguments are enclosed within parenthesis "(" and ")". e.g. ABS(n), ADC(n), MID$(A$,n,m), STRCMP(A$,B$).

Note: Statements or functions and their arguments are NOT case-sensitive. This means that commands such as PRINT and PriNt are identical. However, for clarity seek we use a mix of upper and lower case characters in this manual.

11.1.3 DELIMITER

A TBASIC program consists of many statements. Each statements are usually separated by a different line. The new line therefore acts as a "delimiter" which separate one statement from another. Some statements such as IF..THEN..ELSE..ENDIF span multiple statements and should be separated by proper delimiters.

To make a program visually more compact, the colon symbol ":" may be used to act as delimiter. e.g.

```
IF A > B THEN
      C = D*5
ELSE
      C = D/5
ENDIF
```

may be written more compactly as

```
IF A >B : C=D*5:ELSE:C=D/5:ENDIF
```

11.2 TBASIC Integer Constants, Variables & Operators

The TBASIC compiler in i-TRiLOGI supports full 32-bit integer computations. However, only variable A to Z are 32 bits in length which allow them to represent number between -231 to -231, the remaining system variables and data memory DM[n] are all 16-bit variables which means that they can only store number between -32768 to +32767. However, all numerical computations and comparisons in TBASIC are carried out in 32-bit signed integer, regardless of the bit-length of the variables involved in the numerical expression.

11.2.1 Integer Constants

These may be entered directly in decimal form, or in hexadecimal form by prefixing the number with the symbol "&H". e.g.

```
12345678
&H3EF =1007 (decimal)
```

If the result of an expression is outside the 32-bit limits, it will overflow and change sign. Care must therefore be exercised to prevent unexpected result from an integer-overflow condition.

A constant may be used in an assignment statement or in an expression as follow:

```
A = 12345
IF A*30 + 2345/123 > 100
THEN ....ENDIF
```

IMPORTANT (16-bit variables comparison)

When entering an integer constant using the hexadecimal prefix "&H", it is important to note the sign of the intended value and extend the signs to most significant bit of the 32 bit expression. E.g. to represent a decimal number "-1234", the hexadecimal representation must be "&HFFFFFB2E" and not "&HFB2E".

Assuming that a 16-bit variable DM[1] contains the number -1234 and a comparison statement is made to check if the number is -1234. The 32-bit hexadecimal representation of constant -1234 is &HFFFFFB2E. If you enter the constant as 16-bit representation "&HFB2E" as follow:

```
IF DM[1] <> &HFB2E CALL 5
```

TBASIC translates the number "&HFB2E" into a 32-bit decimal number 64302, which when compared to the number "-1234" contained in DM[1] will yield a "False" result which is an error. The following are the correct representation:

```
a) IF DM[1] <> -1234 CALL 5 : ENDIF
b) IF DM[1] <> &HFFFFFB2E" CALL 5: ENDIF
```

11.2.2 Integer variables:

Variables are memory locations used for storing data for later use. All Integer variables used in TBASIC are GLOBAL variables - this means that all these variables are shared and accessible from every custom function.

With i-TRiLOGI version 6.2 and above, you can now attach comments to any variable name of any length to make program easier to read. See section 5 (Integer Variable Comments) below for more details. Specific examples with each variable type are shown below.

TBASIC supports the following integer variables:

1. 26 Integer variables A, B, C....Z which are 32-bit variables. Note that the variable name is a single character by default; however, as mentioned above, you can append a comment to the variable. EG: A_temperature, A_distance - both refer to the same variable A

2. A large, one-dimensional 16-bit integer array from DM[1] to DM[4000], where DM stands for Data Memory. A DM is addressed by its index enclosed between the two square brackets "[" and "]". e.g. DM[3], DM[A+B*5], where A and B are integer variables. A comment can also be

appended to DM[] variables as follows: DM[1]_Current_status_of _product1 - refers to variable DM[1].

3. Starting from version 6.45, the DM[1] to DM[4000] can also be used as 32-bit variables DM32[1] to DM32[2000] in the following manner: DM[1] & DM[2] == DM32[1], DM[3] & DM[4]==DM32[2]... DM[2N-1] & DM[2N] = DM32[N]. If you require lots of 32 bit variables it may be simpler to use only the DM32 and not the 16-bit DM. If you mix the use of both DM and DM32 then you need to manage the memory properly to ensure that they don't overwrite each other memory space.

   The simulator fully support the use of DM32 variables in any expression. However, for the actual PLC only those with firmware r78 supports DM32 variables. PLC with older firmware cannot directly use DM32 in the program.

   E.g. DM32[100] = DM[1] * A

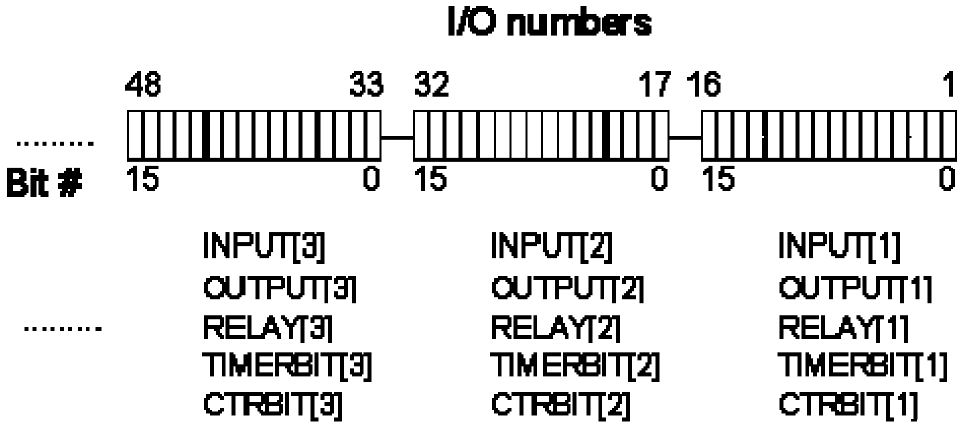
4. System variables. These are special integer variables that relate to the PLC hardware, which will be described in the next section.

11.2.3 System variables:

NOTE:
All of the following System Variables can have comments appended to them with the same format as described in section 5 (Integer Variable Comments).

a) Inputs, Outputs, Relays, Timers and Counters Contacts

The bit addressable I/Os elements are organized into 16-bit integer variables INPUT[n], OUTPUT[n], RELAY[n], TIMERBIT[n] and CTRBIT[n] so that they may be easily accessed from within a CusFn. These I/Os are arranged as shown in the following diagram:



b) Timers and Counters Present Values

The present values (PV) of the 128 timers and 128 counters in the PLC can be accessed directly as system variables:

timerPV[1] to timerPV[256], for timers' present value

ctrPV[1] to ctrPV[256], for counters' present value

c) DATE and TIME Variables

The PLC's Real-Time-Clock (RTC) derived date and time can be accessed via variables DATE[1] to DATE[3] and TIME[1] to TIME[3], respectively as shown in the following table:

Date		**Time**	
YEAR	DATE[1]	HOUR	TIME[1]
MONTH	DATE[2]	MINUTES	TIME[2]
DAY	DATE[3]	SECOND	TIME[3]
Day of Week	DATE[4]		

DATE[1] : may contain four digits (e.g. 1998, 2003 etc).
DATE[4] : 1 for Monday, 2 for Tuesday, ... 7 for Sunday.

d) High Speed Counters

The NANO-10, FMD series, and F-Series PLCs support High Speed Counters (HSC), which can be used to capture high frequency incoming pulses from positional feedback encoder. These high speed counters are accessible by CusFn using the variables HSCPV[1] to HSCPV[8]. All HSCPV[n] are 32-bit integer variables.

e) Special Variables - Used by EMIT.

4 x special 16 bit integer variables: EMEVENT[1] to EMEVENT[4] - emEvent[1] is also used for email purpose.

16 x 16-bit integer variables: EMINT[1] to EMINT[16]

16 x 32-bit integer variables: EMLINT[1] to EMLINT[16]

11.2.4 Integer operators:

"Operators" perform mathematical or logical operations on data. TBASIC supports the following integer operators:

a) Assignment Operator: An integer variable (A to Z, DM and system variables, etc) may be assigned a value using the assignment statement:

```
A = 1000
X = H*I+J + len(A$)
```

b) Arithmetic Operators:

Symbol	Operation	Example
+	Addition	A = B+C+25
-	Subtraction	Z = TIME[3]-10
*	Multiplication	PRINT #1 X*Y
/	Division	X = A/(100+B)
MOD	Modulus	Y = Y MOD 10

c) Bitwise Logical Operators: logical operations is perform bit-for-bit between two 16-bit integer data.

Symbol	Operation	Example
&	logical AND	IF input[1] & &H02 ...
|	logical OR	output[1] = A | &H08
^	Exclusive OR	A = RELAY[2] ^ B
~	logical NOT	A = ~timerPV[1]

d) Relational Operators : Used exclusively for decision making expression in statement such as IF expression THEN ..... and WHILE expression ....

Symbol	Operation	Example
=	Equal To	IF A = 100
<>	Not Equal To	WHILE CTR_PV[0]<> 0
>	Greater Than	IF B > C/(D+10)
<	Less Than	IF TIME[3] < 59
>=	Greater Than or Equal To	WHILE X >= 10
<=	Less Than or Equal To	IF DM[I] <= 5678
AND	Relational AND	IF A>B AND C<=D
OR	Relational OR	IF A<>0 OR B=1000

e) Functional Operators : TBASIC supports a number of built in functions which operate on integer parameters as shown below:

ABS(n), ADC(n), CHR$(n), HEX$(n), STR$(n)

For detailed explanation of these functions please refer to the next chapter: "Programming Language Reference"

11.2.5 Hierarchy of Operators

The hierarchy of operators represent the priority of computation. Eg. X = 3 + 40*(5 - 2). The compiler will generate codes to compute 5 - 2 first because the parentheses has the higher hierarchy, the result is then multiplied by 40 because multiplication has a higher priority then addition. Finally 3 will be added to the result. If two operators are of the same hierarchy, then compiler will evaluate from left to right. e.g. X = 5 + 4 - 3. 5+4 is first computed and then 3 will be subtracted. The following table list the hierarchy of various operator used.

Hierarchy	Symbol	Descriptions
Highest	( )	Parentheses
	*, / , MOD	Multiplication/Division
	+, -	Add/Subtract
	-	Negate
	&, |, ^,~	Logical AND,OR,XOR,NOT
Lowest	=,<>,>,>=,<,<=	Relational operators

11.2.6 Integer Variable Comments

With i-TRiLOGI version 6.2 and above, you can now attach comments to any variable/register name of any length to make program easier to read. The compiler will ignore any alphanumeric characters (A to z, 0 to 9 and '_') that are attached behind the variable name following an underscore character "_". It is important to understand that by appending these comments to variables, no new variables are being created.

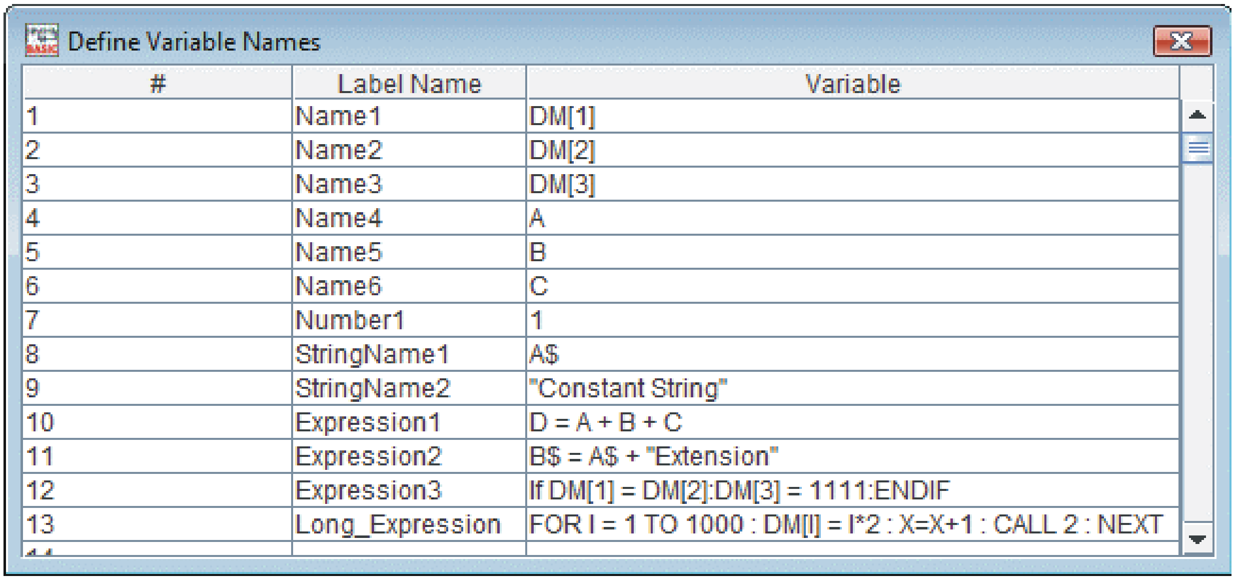
For Example: "X_Some_Integer" corresponds to "X"

An integer variable (such as DM[1]) can have different comments each time it is referenced in the same program because the compiler ignores the comments anyways. Although, in most cases it may be best to limit a variable to one comment to avoid potentially overwriting data since no new variables are actually create, as mentioned above.

For Example: If an integer variable, DM[1], was named "DM[1]_one_integer" in one part of a program and named "DM[1]_two_integer" in another part of the program, then they will still both refer to DM[1]. If each variation has different integer data, then the data that was stored in the last variation that was updated will be the data in DM[1]. Then if the previous variation is accessed, it wont contain the data that was originally stored in it.

11.2.7 Variable Define Table (NEW!)

With TRiLOGI version 6.42 and higher, it is possible to create a table of variable/constant/expression definitions. Please see section "8. Define" in the Introduction to TBASIC & Custom Functions for more information on how to interface to the Define Table. Here is an example of a possible Define Table :



#	Label Name	Variable
1	Name1	DM[1]
2	Name2	DM[2]
3	Name3	DM[3]
4	Name4	A
5	Name5	B
6	Name6	C
7	Number1	1
8	StringName1	A$
9	StringName2	"Constant String"
10	Expression1	D = A + B + C
11	Expression2	B$ = A$ + "Extension"
12	Expression3	If DM[1] = DM[2]:DM[3] = 1111:ENDIF
13	Long_Expression	FOR I = 1 TO 1000 : DM[I] = I*2 : X=X+1 : CALL 2 : NEXT

The following explanations will reference the above example define table.

a) Creating Variable Definitions

It is possible to define label names for any variable available in TRiLOGI. The following results will happen based associated program code :

TRiLOGI Program Code	Result
Name1 = 10	DM[1] would contain the value 10
Name4 = Name5 * 5	A would contain B x 5
DM[Name2] = Number1	DM[DM[1]] would equal 1

b) Creating Constant Definitions

It is possible to define label names for constant values. The following results will happen based associated program code :

TRiLOGI Program Code	Result
A = A + Number1	A would contain its current value plus 1
DM[Number1] = 2	DM[1] would equal 2

c) Creating Expression Definitions

It is possible to define label names for entire expressions or code snippets. The following results will happen based associated program code :

TRiLOGI Program Code	Result
Expression1	D would equal A + B + C

IF A = 1
Expression3
ELSE
B = 1
ENDIF
	If A equals 1, Expression3 will execute and the following code will run :
	If DM[1] = DM[2]:DM[3] = 1111:ENDIF
	If A does not equal 1, B will be equal to 1

Long_Expression	The following code would execute : FOR I = 1 TO 1000 : DM[I] = I*2 : X=X+1 : CALL 2 : NEXT

11.3 String Constants, Variables & Operators

A string is a sequence of alphanumeric characters (8-bit ASCII codes) which collectively form an entity.

11.3.1 String Constants

A string constant may contain from 0 to 70 characters enclosed in double quotation marks. e.g.

"TBASIC made PLC numeric processing a piece of cake!"
"$102,345.00"

11.3.2 String Variables

TBASIC supports a maximum of 26 string variables A$, B$ ... Z$. Each string variable may contain from 0 (null string) up to a maximum of 70 characters.

Note:
For all NANO-10, FMD Series, and F-Series PLCs (as well as legacy M-series PLCs with firmware version r44 and above), you can access the 26 string variables using an index: $$[1] to $$[26]. I.e. A$ is the same as $$[1], Z$ is the same as $$[26]. Note that $$[1] to $$[26] are not additional string variables, it just give you a way to index the string variables not possible on previous firmware version. Also, only i-TRiLOGI version 6 and above properly support these variable names. Caution: Do not try to transfer a program using $$[n] variable to a PLC with firmware earlier than r45 as it can cause the PLC operating system to crash.

With i-TRiLOGI version 6.2 and above, you can now attach comments to any variable name of any length to make program easier to read. See section 10.3.4 (String Variable Comments) below for more details.

11.3.3 String Operators

a) Assignment Operator

A string variable (A to Z, DM and system variables, etc) may be assigned a string expression using the assignment statement:

```
A$ = "Hello, Welcome To TBASIC"
Z$ = MID$(A$,3,5)
```

b) Concatenation Operators

Two or more strings can be concatenated (joined together) simply by using the "+" operator. e.g.

```
M$ = "Hello " + A$ + ", welcome to " + B$
```

If A$ contains "James", and B$ contains "TBASIC", M$ will contain the string: "Hello James, welcome to TBASIC".

c) Comparison Operators

Two strings may be compared for equality by using the function STRCMP(A$,B$). However, the integer comparator such as "=", "<>", etc cannot be used for string comparison.

d) Functional Operators

TBASIC supports a number of statement and functions which take one or more string arguments and return either an integer or a string value. e.g.

```
LEN(x$), MID$(A$,x,y), PRINT #1 A$,.... SETLCD 1, x$ VAL(x$),
```

11.3.4 String Variable Comments

With i-TRiLOGI version 6.2 and above, you can now attach comments to any variable name of any length to make program easier to read. The compiler will ignore any alphanumeric characters (A to z, 0 to 9 and '_') that are attached behind the variable name following an underscore character "_". It is important to understand that by appending these comments to variables, no new variables are being created.

For Example: "A$_Some_String" corresponds to "A$" (or $$[1]).

A string variable (such as A$) can have different comments each time it is referenced in the same program because the compiler ignores the comments anyways. Although, in most cases it may be best to limit a variable to one comment to avoid potentially overwriting data since no new variables are actually create, as mentioned above.

For Example: If a string variable, A$, was named "A$_one_string" in one part of a program and named "A$_two_string" in another part of the program, then they will still both refer to A$. If each variation has different string data, then the data that was stored in the last variation that was updated will be the data in A$. Then if the previous variation is accessed, it won't contain the data that was originally stored in it.

11.3.5 Variable Define Table (NEW!)

This new feature allows you to define an acceptable name for any variable, constant or even an entire expression. Please see section 9.3 Define Table and 10.2.7 Variable Define Table for more details.

Chapter 12 TBASIC Keyword Reference

12 TBASIC KEYWORD REFERENCE

12.1 ABS(x)

Purpose : To return the absolute value of the numeric expression x

Examples : A = ABS(2*16-100)

Comments : A should contain the value 68.

12.2 ADC(n)

Purpose : To return the value from the Analog-To-Digital Converter channel #n. n should be between 1 and 16.

Examples : A = ADC(2)

Comments : n may be a numeric expression which returns a value between 1 and 16. If it is out-of-range, a run-time error will be reported and the function will be aborted.

TRiLOGI software is able to support up to 16 channels of 16-bit bipolar ADC (which may has a range between -32768 and 32767. The actual number of ADC channels and the resolution will depend on the target PLC.

Note:

On the Legacy T100MD+ and T100MX+ PLCs, there are 8 or less ADCs and all the A/D are normalized to 12-bit with a range of between 0 and 4096.

12.3 ASC(x$, n)

Purpose : To return the numeric value that is the ASCII code for the nth character of the string x$. If x$ is a null string, ASC(x$,n) returns value 0. n may start from 1 up to the length of the string.

Examples : B = ASC("Test String",6)

Comments : B should contain the value 83 (which is ASCII value of 'S'). If n is less than 1 or greater than string length, ASC(x$, n) returns a 0.

See Also : CHR$(n)

12.4 CALL n

Purpose : To call another Custom Function CusFn #n as subroutine. When the called function returns, execution will continue from the following statement. n must be either an integer constant between 1 and 256, or the label name of the Custom Function defined in the Custom Function table.

Examples : IF B > 5 THEN CALL 8 : ENDIF
CALL Addition “Addition” must be a defined name.

See Also : RETURN

12.5 CHR$(n)

Purpose : To convert a number n into its corresponding ASCII character. n must be a numeric constant (0 to 255)

Examples : C$ = "This is Message #" + CHR$(&H35)

Comments : C$ should contain: "This is Message #5", since CHR$(&H35) returns the character '5'. In TL6 you can also use “Escape” sequence to represent non-printable ASCII characterr E.g. "/0D” to represent CR character.

See Also : ASC( )

12.6 CLRBIT v, n

Purpose : To clear the Bit #n of the integer variable v to '0'. n is an integer constant or variable of value between 0 and 15. v may be any integer variable or a system variable such as relay[n], output[n], etc. If v is a 32-bit integer, CLRBIT will only operate on the lower16 bits.

Following digital electronics convention, bit 0 refers to the least significant bit (right most bit) and bit 15 the most significant bit (left most bit) of the 16-bit integer variable.

Examples	:	CLRBIT output[2],11
Comments	:	Physical output #28 will be turned OFF. (Output channel #2 bit #11 = Output #17 +11 = 28)
See Also	:	SETBIT, TESTBIT, SETIO, CLRIO,TOGGLEIO & TESTIO

12.7 CLRIO labelname
SETIO labelname
TOGGLEIO labelname
TESTIO (labelname)

Purpose : Manipulate the logic states of any input, output, relay, timer or counter contact bit within a CusFn. The labelname refers to the label names defined in the input, output, relay, timer or counter tables.

SETIO set a bit to ON, CLRIO clear the bit to OFF, and TOGGLEIO flip the current logic state of that I/O bit. TESTIO function returns a 1 if the bit is ON and a 0 if the bit is OFF.

Examples : SETBIT alarm
IF TESTBIT(alarm) THEN … ELSE …ENDIF

Comments : This function offers a more efficient way of manipulating the I/O bits compared to the SETBIT and CLRBIT function. However, SETBIT and CLRBIT functions have the advantage that they can use variables to indicate the index and bit position of the bit to be affected, whereas the I/O bit that affected by the commands here are fixed during compile time. Note that output bit changed in custom function will only be updated at the physical output at the end of the ladder logic scan unless a "REFRESH" command is being executed.

See Also : SETBIT, CLRBIT

12.8 CONTINUE

Purpose	:	To command the CPU to jump to the next iteration of the WHILE..ENDWHILE or FOR..NEXT loop.
Syntax	:	CONTINUE
Comments	:	This new command is available to all Super PLCs since firmware version r30. Upon executing the CONTINUE command the CPU will jump to the next iteration in the WHILE..ENDWHILE or FOR..NEXT loop.

Example :

```
FOR I = 1 to 10000
   X = X + 1
   IF INCOMM(1) < 0
      CONTINUE
   ENDIF
   Y = Y + 2
NEXT
```

If INCOMM(1) returns a -1 as in the second line of the FOR loop, the CPU will not execute the Y = Y + 2 statement, but will instead jump to the next iteration, which is to increment index I by 1 and start executing the statement X = X + 1.

See Also : WHILE..ENDWHILE , FOR..NEXT , EXIT

12.9 CRC16 (var, count)

Note	:	{Applicable to all Nano-10, FMD Series, and F-series, but only to the legacy T100MD/MX PLC with firmware r44 or higher}
Purpose	:	This function returns the computed CRC16 for a range of integers starting from variable “var” with the range indicated in the parameter “count”. CRC16 is a 16-bit version of “Cyclic Redundancy Check” – a popular mathematical formula for checking error in a data stream.

Examples :

```
DM[100] = CRC16(DM[5],8)
X = CRC16(RELAY[2],4)
```

Comments	:	CRC16 for DM[5], DM[6].....DM[12] will be assigned to DM[100] CRC16 for RELAY[2], RELAY[3], RELAY[4], RELAY[5] will be assigned to X.

12.10 DELAY n

Purpose	:	To provide a time delay of n millisecond to the process.
Example	:	DELAY 100
Comments	:	Provide a 100 ms (0.1s) delay to the current custom function.

It is important to note that this is a "brute force" delay method and only to be used with caution. When a DELAY function is executed the CPU waits at the statement until the period specified by the "delay" is over. This means that all the remaining ladder programs and other custom functions will stop responding to changing input conditions, only system services (serial input, countdown timers and host link commands etc) as well as interrupt driven CusFns will work during the period of delay. This may not be desirable if the rest of the process must respond to fast changing inputs. For delays longer than 0.1s a much better way is to invoke the regular PLC timer and use the timer contact to trigger another custom function at the end of the delay.

Note that for Nano-10, FMD Series, and F-series PLCs, the minimum delay is 1ms and the resolution is 1ms, whereas for the Legacy T100MD+ and T100MX+, the minimum delay provided by this function is 10ms, and the resolution of the time delay is 10ms. This means that if you execute DELAY 155 the actual delay will be rounded to 160ms, whereas for DELAY 154 the actual delay will be 150ms.

12.11 EXIT

Purpose	:	To exit from a WHILE.. ENDWHILE or FOR....NEXT loop

Syntax : EXIT

Comments : This new command is only available to PLC with firmware version r75 and above. Instead of using the GOTO command to get out of the loop, the EXIT command will orderly end a WHILE..ENDWHILE loop or a FOR...NEXT loop and jump to the next statement after the ENDWHILE or the NEXT command

Example :

```
S = 1
WHILE S
    S = S+1
    IF S >= 100 EXIT: ENDIF
    DELAY 1
ENDWHILE

X$ = INPUT$(1)
```

When S has been incremented to 100, the CPU will exit the loop and continue executing the command after the ENDWHILE statement, which is the X$ = INPUT$(1) statement.

See Also : WHILE..ENDWHILE , FOR..NEXT , CONTINUE

12.12 FOR ... NEXT

Purpose : To execute a series of instructions for a specified number of times in a loop.

Syntax :

```
FOR variable = x TO y [STEP z]
.    .    .    .

NEXT
```

where variable may be any integer variable A to Z only and is used as a counter. x, y and z are numeric expressions. STEP z is an optional part of the statement.

x is the initial value of the counter, y is the final value of the counter.

Program lines following the FOR statement are executed until the NEXT statement is encountered. Then the counter is incremented by the amount specified by STEP. If STEP is not specified, the increment is assumed to be 1.

A check is performed to see if the value of the counter is greater than the final value y if STEP is positive (or smaller than the y if STEP is negative). If it is not greater, the program branches back to the statement after the FOR statement, and the process is repeated. If it is greater, execution continues with the statement following the NEXT statement. This is called a FOR-NEXT loop.

A run-time error will result if STEP is evaluated to be 0.

Examples :

```
FOR I=1 TO 10
    FOR J = 100 to 1 STEP -10
            DM[I] = DM[J]
    NEXT
NEXT
```

Comments : FOR-NEXT loops may be nested; i.e. a FOR-NEXT loop may be placed within the context of another FOR-NEXT loop. When loops are nested, each loop must have a unique variable name as its counter. The NEXT statement for the inside loop must appear before that for the outside loop. Each Loop must have a separate NEXT statement to mark the end of the loop.

See Also : WHILE..ENDWHILE , CONTINUE

12.13 GetCtrSV (n)
GetTimerSV (n)

Purpose : Return the Set Value (S.V,) of the Counter #n or Timer #n. n should be between 1 and the maximum number of timers and counters in your PLC.

Note : Although the present values (P.V.) of timers and counters #n can be accessed directly as variables “TimerPV[n]” & “CtrPV[n]”, the Set Values however can only be obtained by these two functions.

See Also : SetCtrSV, SetTimerSV

12.14 GETHIGH16(v)

Purpose : This function returns the upper 16-bit of a 32-bit integer variable v. This can be used to break the value of a 32-bit integer data or variable into two 16-bit values so that they can be saved to the EEPROM or to the DM[n].

Examples :

```
DM[1] = GetHIGH16(A)
save_EEP   GetHIGH16(&H12345678), 10
```

See Also : SETHIGH16

GOTO @ n

Purpose : To branch unconditionally out of the normal program sequence to a specified line with label @n within the present Custom Function.

The destination line must have a corresponding line label marked as "@n", where n must be a constant within 0-255. Note that the label is local only to the present CusFn. i.e. another CusFn may have a label with the same n but the GOTO @n will only branch to the line label within the same CusFn.

Examples :

```
@156 SETBIT 0,3
        . . .
GOTO @156
```

Comments : An error message will appear during compilation if the destination label is undefined.

12.15 HEX$(n)
HEX$ (n, d)

Purpose : To return a string that represents the hexadecimal value of the numeric argument n. If the second format is used then this function will return a string of ‘d’ number of characters.

Examples : A$ = HEX$(1234)
B$ = HEX$(1234,7)

Comments : A$ will contain the string : "4D2" , B$ will contain the string “00004D2”.

See Also : HEXVAL( ), STR$( ), VAL( )

HEXVAL(x$)

Purpose : To return the value of a hexadecimal number contained in the argument x$.

Examples : B = HEXVAL("123")*100

Comments : B should contain the value 29100 (&H123 =291)

See Also : HEX$( ), STR$( ), VAL( )

12.16 HSCDEF ch, fn_num, value

Purpose : Enable and set up parameters for the High Speed Counters channel ch. These counters operate independently of the ladder logic scan time and can capture high speed input pulses generated by position encoders.

ch = channel number (1-8)
fn_num = Custom Function # to trigger when value is reached.
value = trigger when HSC reach this (32-bit) integer value.

If the PLC supports quadrature encoder inputs, then the HSC counter variable HSCPV[ch] will increment/decrement according to direction of rotation. When value is reached, the specified custom function activates immediately.

Important : All High Speed Counters are disabled automatically when the PLC is reset unless they are enabled by the HSCDEF statement. However, if more than one HSCDEF for the same channel ch is executed, only the last executed HSCDEF statement will take effect. Hence you should put the next HSCDEF statement within the CusFn triggered by the first HSCDEF. By chaining the HSCDEF statement from one CusFn to another, you can control the motion of the machine using the HSC value to execute a series of CusFn one by one. Within these CusFn you can program what to do to control the motion. E.g. changing the speed, putting on the brake, change direction of motion, etc. You can use the SETIO, CLRIO for digital ON/OFF control and setDAC, setPWM for proportional control.

Example :
```
HSCPV[1] = 0
HSCDEF 1,19,-3310003
. . . .
SETLCD 1,1,STR$(HSCPV[1],6)
```

Comments : Enable High-Speed Counter #1 and make it activate function #19 when the counter reaches -33,100,003. Present value of HSC#1 was cleared to 0 before activating it. Note that TRiLOGI Version 5.x does not perform simulation of the High Speed counter operation since there is no High Speed Counter inputs on the simulator screen.

Note that the HSDDEF statement will also activate the Pulse Measurement hardware as described in the Pulse Measurement section of the Nano-10, FMD series, and F-series User Manuals (does not apply to the legacy T100M series PLCs)

See Also : HSCOFF

12.17 HSCOFF ch

Purpose : Disable High Speed Counter #ch (ch = 1 to 8)

If you no longer need the high speed counter, it should be disabled in order not to waste the CPU’s time to service the interrupt generated by the change of state at the HSC input..

12.18 HSTIMER n

Purpose : To define PLC Timer #1 to #n as “High Speed Timers” (HST). A HST counts down every 0.01s instead of every 0.1s for normal timer, and their other properties are identical to normal timer.

Those Timers whose number are above n are not affected and remain ordinary timers.

12.19 I2C_READ i2cslave, dmstart, count

Purpose : Special command to execute a I2C_READ out of the PLC's I2C port (requires I2C-FRTC module to be installed).

Comments : The CPU will send a I2C START bit, followed by the slave address byte (i2cslave) with "R/W" bit set to high, and then send out the number of clock pulses required to read count number of data bytes from the the slave. The data bytes received from the slave will be stored in the memory location DM[dmstart] to DM[dmstart+count-1]. After receiving all the required data bytes the CPU will send the I2C STOP bit to the slave to end the communication.

i.e. there is no need execute the I2C_STOP command after an I2C_READ command.

Parameters :

i2cslave = The 7-bit slave address that the CPU is reading from.

dmstart = The starting index of the DM[ ] that is to receive the first data

count = number of data bytes to read from the slave.

Example : I2C_READ &H0C,21,2 ' read 2 bytes into DM[21] and DM[22]

After sending the START bit, the CPU will write the 7-bit slave address &H60 (=110 0000 binary) and a R/W bit set to 1, followed by 16 clock pulses to read 2 bytes of data and store into DM[21] and DM[22], and then the CPU will generate the STOP bit.

See Also : I2C_WRITE, I2C_STOP

Note:

1. This command is applicable only to Nano-10 and FMD PLC models with firmware >=r74 and I2C interface.

2. This command is ignored by i-TRiLOGI simulator.

12.20 I2C_STOP

Purpose	:	Special command to generate an I2C STOP bit out of the PLC's I2C port (requires I2C-FRTC module to be installed).
Examples	:	DM[5]= 12: DM[6]= 34 : DM[7]= 56 I2C_WRITE &H60,5,3 I2C_STOP Since the I2C_WRITE command does not automatically generate the STOP bit, this command is used to end I2C communication started by an I2C_WRITE command.

See Also I2C_READ, I2C_WRITE

Note:

1. This command is applicable only to Nano-10 and FMD PLC models with firmware >=r74 and I2C interface.

2. This command is ignored by i-TRiLOGI simulator.

12.21 I2C_WRITE i2cslave, dmstart, count

Purpose	:	Special command to execute a I2C WRITE out of the PLC's I2C port (requires I2C-FRTC module to be installed). The CPU will send a I2C START, followed by the slave address byte (i2cslave) and count number of data bytes from the DM[dmstart] up to DM[dmstart+count-1]
Comments	:	The command automatically checks for an ACK sent from the slave device, and the user program can check the status of this operation by testing the STATUS(2) function. STATUS(2)

returns a 1 if an ACK is received, and 0 if no ACK is received after time out.

Parameters :

i2cslave = The 7-bit slave address that the CPU is writing to.

dmstart = The starting index of the DM[ ] that contains the first data byte

count = number of byte data to send (maximum is dependent on the slave).

Example :

```
DM[5]= 12: DM[6]= 34 : DM[7]= 56
I2C_WRITE &H60,5,3
I2C_STOP
```

Following the START bit, the CPU will write the 7-bit slave address &H60 (=110 0000 binary) and a R/W bit set to 0, followed by the byte data stored in DM[5], DM[6] and DM[7].

Note : This command does not automatically generate the I2C STOP bit, this is to allow the CPU to perform a I2C_READ following a I2C_WRITE. I2C READ after WRITE is commonly encountered in I2C protocol which requires using the I2C_WRITE to set the internal pointer address in the slave device and then followed by the I2C_READ command.

Therefore, if your command involves only I2C_WRITE, you must end the WRITE command by executing a I2C_STOP statement.

See Also : I2C_READ, I2C_STOP

Note:

1. This command is applicable only to Nano-10 and FMD PLC models with firmware >=r74 and I2C interface.
2. This command is ignored by i-TRiLOGI simulator.

12.22 IF .. THEN .. ELSE .. ENDIF

Purpose : To make a decision regarding program flow based on the result returned by an expression.

Syntax :

```
IF expression [THEN]
.....
[ELSE]
.....
ENDIF
```

If the result of the expression is non-zero (logical true), the block of program lines between the THEN and the ELSE statements will be executed. If the result of the expression is zero (false), the block between the IF and ELSE will be ignored, and the block between the ELSE and ENDIF statements will be executed instead.

If there is no ELSE statement, and if the result of the expression is false, the block of program lines between the THEN and the ENDIF statement will be ignored, but execution will continue right after the ENDIF statement.

Nesting of IF statement: Statement blocks within the IF..THEN..ELSE statement may contain other IF..THEN..ELSE blocks (nesting). Note that each IF statement must be ended with the ENDIF statement. Otherwise an error message "IF without ENDIF" will be reported during compilation.

Testing Equality: Special comparison operators may be used in the expression of the IF statement. Only integer expression may be compared. For comparison of strings, please refer to the "STRCMP(A$, B$)" function.

Equal	=
Not Equal	<>
Greater than	>
Less than	<
Greater than or Equal to	>=
Less than or Equal to	<=

Examples :

```
IF A >= B*5-20*C OR C=20
        B = B-1
ELSE
        B = B*3
ENDIF
```

Comments : A few comparison expressions may be linked with logical-AND (AND statement) or logical-OR (OR statement) operator as shown in the above examples.

12.23 INCOMM(ch)

Purpose : To return a single 8-bit binary data obtained from Comm. channel # ch.

ch must be a numeric constant between 1 and 8. The actual target hardware determines the valid port #. This function returns -1 if there is no data waiting at serial port.

Example :

```
FOR I=1 to 100
    DM[I] = INCOMM(2):
    IF DM[I]<0 RETURN :ENDIF
NEXT
```

Comments : Usually the PLC buffers the serial data arriving at its COMM port so that the program does not need to continuously check the COMM port for data. When the program is ready to process the data it can use the FOR..NEXT loop shown in the above example to read in all the data in the COMM buffer until it encounters a -1, which indicates that the buffer is empty.

Note: INCOMM is supported on all COMM ports of the Nano-10, FMD series, and F-series families of PLCs (as well as the legacy T100MD/T100MX PLCs).

See Also : OUTCOMM, INPUT$( ), PRINT #

12.24 INPUT$(ch)

Purpose : To return a string obtained from communication port # ch.

ch must be a numeric constant between 1 and 8. The actual target hardware determines the valid port #. This function returns an empty string if there is no valid string waiting at serial port in order not to hold up the CPU.

Example :

```
For I=1 to 1000       'loop 1000 times
        A$ = INPUT$(1)   ' Read from COMM1
        IF LEN(A$)<> 0   ' String is not empty.
                RETURN          ' A$ contains data.
        ENDIF
NEXT
SETLCD 1,1, "Nothing received"
```

Comments : A Carriage Return (CR) or ASCII code 13 marks the end of a valid input string from the communication port. The returned string however will exclude the CR character. In TRiLOGI simulator, the user will be prompted to enter the string in a pop-up window.

See Also : INCOMM( ), PRINT #, OUTCOMM

12.25 INTRDEF ch, fn_num, edge

Purpose : Enable Interrupt Input channel #ch.
ch = channel number (1-8)
fn_num = Custom Function number to execute when interrupt pin changes according to the defined edge. This is the Interrupt Service Routine ISR.
edge = Positive number means rising edge-triggered. 0 or negative number means falling-edge triggered.

See Also : INTROFF

12.26 INTROFF ch

Purpose : Disable Interrupt Input channel # ch.

See Also : INTRDEF

12.27 LEN (x$)

Purpose : To return the number of characters in x$.
Examples : L = LEN("This is a test string"+CHR$(13))

Comments : L = 22 because blanks and non-printing characters are counted.

12.28 LET

Purpose : To assign the value of an expression to a variable
Syntax : [LET] variable = expression
Examples : LET D = 11
A$ = "Welcome to TBASIC"

Comments : LET statement is optional: i.e. the equal sign is sufficient when assigning an expression to a variable name. The variable type on both sides of the equal side must be the same. i.e. string variable may not be assigned to a numeric expression and vice-versa.

Important : a) When assigning a 16-bit variable to a 32-bit integer, only the lower 16 bits of the 32-bit integer will be assigned. Hence the programmer must take special care if the 32-bit number is out of the range of a 16-bit number (which is between -32768 to 32767).

b) If a negative 16-bit number is assigned to 32-bit integer variable, then the sign bit will be extended to 32 bits.
e.g. DM[1] = -123.
A = DM[1]

The 16-bit hexadecimal value of -123 is &HFF85, but A will be assigned the hexadecimal value &HFFFFFF85. Their decimal representation are however the same.

12.29 LOAD_EEP(addr)
LOAD_EEP8 (addr8)
LOAD_EEP16 (addr16)
LOAD_EEP32 (addr32)

Note : LOAD_EEP8, LOAD_EEP16, and LOAD_EEP32 are only available on PLCs with firmware >= r78. LOAD_EEP is equivalent to LOAD_EEP16 and is available on all firmware versions.

Purpose : To return a 16-bit integer data stored in the EEPROM/FRAM by the SAVE_EEP statement, or 8-bit integer data stored in the FRAM by the SAVE_EEP8 statement, or 32-bit integer data stored in the FRAM by the SAVE_EEP32 statement.

addr8 - Byte address of EEPROM/FRAM .
addr16 - Word address of EEPROM/FRAM .
addr32 - Long Word address of EEPROM/FRAM .

Please refer to your PLC's reference manual for the upper limit of available FRAM/EEPROM storage.

Byte, Word and Long Word Addressing: Note that the EEPROM space for 8-bit, 16-bit and 32-bit SAVE_EEPXX and LOAD_EEPXX command are the same space and they are mapped as shown below

LOAD_EEP32	**LOAD_EEP16**	**LOAD_EEP8**
1	1 and 2	1, 2 3 & 4
2	3 and 4	5,6,7,8
…	**…**	…
N	2N-1 & 2N	4N-3, 4N-2, 4N-1, 4N

If you intend to use the EEPROM/FRAM space to store more than 1 type (8, 16 or 32-bit) of data then you must properly manage the EEPROM/FRAM data space reserved for storing each type of data so that they don't overwrite each other address space. The easiest way is always to reserve enough space for 32-bit data, followed by space reserved for 16-bit and/or 8 bit data.

E.g. If you have 6000 words (16-bit) of FRAM space - this means you have space for storing 6000*2 = 12000 bytes of 8-bit data and you only have 6000/2 = 3000 long words for storing 32-bit data.

Now assume that your program needs to store up to 500 x 32-bit data and the balance for storing 8 bit data.

If you reserve 500 long words for 32-bit data, that means you are using up 500*4 = 2000 bytes of the FRAM space. This leaves you 12000 - 2000 = 10000 bytes of FRAM space for storing 8-bit data starting from byte address space 2001 to 12000.

Examples : DM[1] = LOAD_EEP8(10): A = LOAD_EEP8(2)
relay[1] = LOAD_EEP(10): A = LOAD_EEP(2)
DM32[1] = LOAD_EEP32(10): A = LOAD_EEP32(2)

See Also : SAVE_EEP

12.30 LOAD_EEP$ (addr)

Note : Applicable to all Nano-10, FMD Series, and F-series, but only to the legacy T100MD/MX PLC with firmware r44 or higher.

Purpose : This function returns a string previously saved into the PLC's internal data EEPROM using the "SAVE_EEP$ command.

Examples :

```
X$ = Load_EEP$(5)
FOR I = 1 to 5
        $$[I] = Load_EEP$(I+10)
NEXT
```

Comments : 1. String saved in EEPROM string location #5 is loaded into X$
2. Strings saved in EEPROM string locations #11 to #15 are loaded into A$ to E$ ($$[1] to $$[5] represents A$ to E$)

See Also : SAVE_EEP$ for explanation of how the data EEPROM area are organized in these PLC's to provide storage area for both integers and strings.

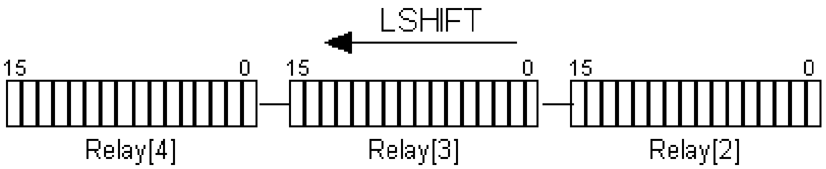
12.31 LSHIFT i, n

Purpose : To shift 1 bit to the left the integer variable i which must be either an integer variable, a DM[n] or a system variable such as relay[n], output[n], etc.

LSHIFT instruction permits more than one variable to be chained together before performing a bit shift. The parameter n indicates the number of channels to be chained starting from i upward. n =1 if only one variable is involved.

Examples : LSHIFT relay[2],3

Comments : The relay channels #2,#3, and #4 (which represent relays number #17 to #64) are chained together in the following manner:



Bits are shifted from the lower channel towards the upper channel. Bit #15 of Relay[2] will be shifted into Bit #0 of Relay[3] and so on. Bit #15 of the highest channel Relay[4] will be lost.

See Also : RSHIFT

12.32 MID$(x$, n, m)

Purpose : This function returns a sub-string of m characters from x$, beginning with the nth character.

x$ - any string expression, variable or constant.

n - any numeric expression producing a result of between 1 to 255

m - any numeric expression producing a result of between 0 to 255.

Examples : A$ = MID$("Welcome to TBASIC",4,7)

Comments : A$ should contain the string :"come to".

12.33 NETCMD$ (ch, x$)

Purpose : This function sends a multi-point host link command string specified in the x$ via serial port #ch to another Nano-10, FMD series, F-series, or H-series/E10 PLCs (or the legacy M-series PLCs). It will then wait for a specified amount of time for a response string from the other PLC and this response string is then returned.

ch - This refer to the communication port #. Please refer to the target PLC for details.

x$ - contains a valid host link command in multi-point format, excluding the Frame Check Sequence (FCS) and the terminator characters (* and CR). NETCMD$ function will automatically compute the FCS and append to the end of x$ and together with the terminator characters will be sent to the other PLC via COMM #ch.

Note : 1) If the target PLC does not respond then this function returns an empty string.

2) This function checks the FCS of the response string, and if the FCS is wrong it indicates an error in the serial reception and it will return an empty string.

Examples : A$ = NETCMD$(3, "@05RI00")

Comments : To read the Input channel #0 of the PLC with ID = 05 connected to COMM #3 of this PLC. The response string will be assigned to A$.

Special : If the last character of x$ is a “~” character, NETCMD$ will send out the string without the ‘~’ character. It will not append the FCS and ‘*’ to the outgoing string and it will not send out the carriage return (ASCII 13) character. It will also NOT check the response string for FCS. This allow NETCMD$ to be used to interface to third-party ASCII devices with different command/response formats.

E.g. A$ = NETCMD$(3, “Hello World~”)

The string “Hello World” will be sent out of serial COMM port #3. A$ will receive the full returned string without applying any FCS check on the return string.

12.34 OUTCOMM n, x

Purpose : This statement can be used to send an 8-bit byte of data ' x ' via Comm port #n. This command is added because PRINT#n command cannot be used to send out CHR$(0). Zero is treated as the end of a string in TBASIC and will be ignored if you use PRINT #n statement to send out CHR$(0).

Examples : OUTCOMM 2,225

12.35 PAUSE

Purpose : To set a breakpoint for executing the CusFn. This is used mainly for debugging a CusFn. By Inserting a PAUSE statement at the place of interest, you can suspend the program execution when PAUSE is encountered, after which you may examine the values of the relevant variables. You can continue to perform on-line monitoring of the PLC that has been paused. Program execution can also be continued by pressing the <P> key during Simulation or On-line Monitoring.

12.36 PIDcompute(ch, E)

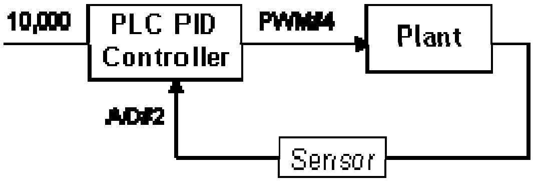
Purpose : This function computes the output for the PID compensator/ controller, using the P,I, and D Gains defined in the PIDdef statement for the same channel ch. The integral and differential values are stored within the channel's internal data space and will be automatically used by the PID computation routine. The PIDcompute( ) function uses the lmt (max. limit) term of PIDdef statement to limit the results of its computation. If the absolute value of the computed result is greater then "lmt", then the result will be set equal to "lmt" for +ve number and to "-lmt" for negative value. When this happens, the integral term will not accumulate the current error to prevent an "integrator windup" which is very undesirable for the system.

ch = channel number (1-16)

E = Closed-loop Error.
(i.e. Set point value - Feedback Value)

The controller may obtain feedback from ADC, High Speed Counters, PULSEFREQUENCY or other means. The obtained result is then scaled and subtracted from the desired (set point) value to get "Err ". All computations are performed in 32-bit integers and the function returns a 32-bit integer that can be assigned to any variable. Any scaling for actual output (DAC or PWM) will be computed by the user within the same CusFn and sent to the output.

Example :



E.g. Implementing Closed-loop Digital Control with PID computation function

```
E = 10000 - ADC(2)*20
A = PIDcompute(5,E)
setPWM 4, (A + 8000)/100
```

Comments: The set point value is 10000 units, the feedback value is read from ADC channel #2 and then multiplied by 20 to convert (scale) it to the same unit as the parameter to be controlled. PID computation channel #5 (assume somewhere in the program a PIDdef for channel #5 has been executed before) is then used to compute the desired controller output value using the error signal (= set point - feedback value ADC(2) x 20).

The desired output (stored in variable A) is then added to the offset value 8000 and then scaled down by a factor of 100 before being sent out physically via PWM Channel #4.

Important : In actual implementation, use a clock pulse such as 0.1s, 0.5s or 1s etc to periodically activate the PIDcompute( ) function so that digital control in discrete-time can be implemented. The PID sampling period depends on the time constant of the system. For very slow response processes such as the cooking temperature of a large body of water, the time constant is very large and even slower than 1.0 seconds clock may be sufficient. Do not use unnecessarily short sampling time because it increases computation time and slows down overall performance of the system.

12.37 PIDdef ch, lmt, P, I, D,

Purpose: To set up the parameters for a Proportional, Integral and Derivative (PID) Controller function. The function PIDcompute( ) will make use of the parameters defined here for the corresponding channel #ch.

Ch = channel number (1-16)
Lmt = Maximum (saturation) limit for the computed result.
P = Proportional Gain (KP)
I = Integral Gain (KI)
D = Differential Gain (KD)

Transfer Function of a PID Controller are defined as follow:

$$G(s) = K_P + \frac{KI}{s} + K_D\, s$$

$$K_P = \text{Proportional Gain} = \frac{1}{\text{Porportional Band}}$$

$$K_I = \text{Integral Gain} = \frac{1}{\text{Integral Time Constant}}$$

All four parameters: lmt, P, I & D can be either 16 or 32-bit integer constants or integer variables. For the lmt term, the computed controller output value by the PIDcompute( ) function is not allowed beyond the + lmt value (i.e. lmt represents the saturation point of the computed controller output). PIDcompute( ) function implements "Integrator anti-windup" feature, which will avoid integrating the error signal when output is already saturated .

Important: When this statement is run, the integral and differential terms of channel ch is set to zero. Hence PIDdef should be run only once during initialization and not repeatedly executed. Otherwise the PIDcompute( ) function will not run properly because of the loss of integral and differential data.

See Also : PIDcompute( )

12.38 PMON ch
PMOFF ch

Purpose : PMON enables Pulse Measurement Function at channel #ch, whereas PMOFF disables the channel. After enabling the channel, you may then use the functions PULSEWIDTH(ch) and PULSEPERIOD(ch) to obtain the width and period of the input pulses arriving at the pulse measurement input pin. You must call PMON once during initialization to enable the pulse measurement hardware. Otherwise the two functions will only return 0. You should avoid repeatedly executing PMON function, otherwise the pulse measurement hardware will be reset repeatedly as well, and accurate measurement cannot be obtained.

If you no longer need to measure the pulse-width or period for a particular channel which has been PMON before, you should disable it using PMOFF to save CPU time because pulse measurement is interrupt driven and consumes CPU time.

Example: PMON 1 : PMOFF 5

See Also : PULSEWIDTH( ) / PUSEPERIOD( ) / PULSEFREQUENCY

12.39 PRINT# n x$; y; z.... Statement

Purpose : To send a string of ASCII characters formed by its parameter list (x$; y; z) out of the PLC to other devices via the communication port #n.

Parameters : n must be an integer constant of between 1 and 8. Integer value in the parameter list (y; z..) will be converted into the equivalent ASCII representation. Each parameter must be separated by the semicolon(;).

Action : The ASCII string is first formed by the PRINT statement using all the arguments in the argument list and the completed string is then sent out of the serial channel #n at one go. The PRINT statement automatically sends a Carriage Return (CR-ASCII 13) out of the specified serial port after sending out the last character in the argument list. A PRINT statement that ends with a semi-colon “;", will not send the CR character.

If you have a long string to send than you can use ";" to break the whole command into several lines, with each line ending with a ";" except the last lines.

Examples : PRINT #2 "The value of A+B = ";A+B;
PRINT #2 "Units"

Comments : IF A=5 and B=100, the string "The value of A+B = 105 Units" and a CR character will be sent out via Comm. port #2. In TRiLOGI simulation mode, the ASCII string will be displayed on a pop-up window to simulate PRINT action.

See Also : INPUT$( )

12.40 PULSEFREQUENCY(ch)
PULSEPERIOD(ch)
PULSEWIDTH(ch)

Purpose : Return in Hz the frequency of the last input pulse; Return in microseconds the width or period of the input pulses arriving at channel #ch of the pulse-measurement pin.

Parameters :

ch = channel # (1-8)

Comments : The pulse-measurement channel #ch must have been enabled by the PMON statement already. However, if you need to use the HSC and the Pulse Measurement on the same channel, then you don’t need to execute both the HSCDEF and PMON, so you should only need to execute the

HSCDEF. The HSCDEF function will automatically configure the Pulse Measurement function so it is not necessary to use the PMON function (in fact If you run the PMON command after HSCDEF, it actually disables the HSC). However, if you use only the PMON function, it would not enable the HSC function.

If the pulses stop coming in, then PULSEFREQUENCY will return a zero while the other two functions will saturate at a certain maximum value (for Nano-10, FMD series, and F-series PLCs it is equivalent to about 3.28 seconds)

The SETSYSTEM 20 command can be used to switch the resolution between 1us (default) and 0.1us.

Example : A = PULSEWIDTH(1)

See Also : PMON / PMOFF

12.41 READMODBUS (ch, ID, addr)

Purpose : Use the MODBUS ASCII or RTU protocol to automatically query a MODBUS ASCII or RTU slave device and obtain the desired 16-bit register data. The communication baud rate is the default baud rate of that Comm port unless it has been changed by the SETBAUD command.

Parameters :

ch - PLC Comm port number
(1 to 3 using Modbus ASCII or 11to 13 using Modbus RTU).
ID - device ID of the MODBUS device (1 to 255)
addr - zero-offset address of the holding register in the MODBUS device.

Example : relay [3] = READMODBUS(3, 5, 101)

Comments : The relay will contain the 16-bit data obtained from the MODBUS device with ID = 05 and from register offset address 101 (in MODBUS term this refer to the #40102 holding register) . Reading it into the relay[ ] channel allows bit level manipulation by ladder logic. It can of course also be read into any data memory.

This command automatically checks the response string received from the slave device for the correct LRC and the slave address. The status of the operation can be checked by the user program by testing the STATUS(2) function, which will return a '0' if there is any error or if the slave device is not present.

Special : This command by default uses MODBUS function 03 to perform the read, which means that it is designed to read from holding register addresses (4-xxxx). You can force it to change to using function 04 in order to read from addresses 3-xxxx by executing SETSYSTEM 6,4 (only for CPU with firmware revision r47 and above).

See Also : READMB2, WRITEMB2, WRITEMODBUS, STATUS(2), NETCMD$( ), SETSYSTEM

12.42 READMB2 ch, ID, addr, var, count

Note : Applicable to all Nano-10, FMD Series, and F-series, but only to the legacy T100MD/MX PLC with firmware r44 or higher

Purpose : Think of this as the multi-word version of READMODBUS command. Unlike the READMODBUS command which is a function that returns a single 16-bit word, this command is implemented as a statement so that multiple words of data can be stored into the PLC internal memory .

Parameters :

ch - PLC COMM port number
(1to 8 using Modbus ASCII or 11to 18 using Modbus RTU).

ID - Device ID of the MODBUS slave device (1 to 255)

addr - Zero-offset address of the holding register in the MODBUS slave device starting from 0 = 40001.

var - the starting variable in the master for storing the returned data. (may be a DM or any system variable) count - number of variables to read (max. =16).

Example : READMB2 3,5,101,DM[10],8

Comments : The PLC will use MODBUS ASCII protocol, via its Comm port #3, to query the slave MODBUS device with ID = 05 and ask for 8 words of data starting from register offset address 101 (in MODBUS term this refer to the #40102 holding register) . Once it receives the returned data these 8 words will be stored in the memory locations: DM[10], DM[11],.....DM[17].

This command automatically checks the response string received from the slave device for the correct slave address and LRC (or CRC16 RTU protocol is used). Like READMODBUS command, the status of this operation can be checked by the user program by testing the STATUS(2) function.

Special : This command by default uses MODBUS function 03 to perform the read, which means that it is designed to read from holding register addresses (4-xxxx). You can force it to change to using function 04 in order to read from addresses 3-xxxx by executing SETSYSTEM 6,4 (only for CPU with firmware revision r47 and above).

See Also : READMODBUS, WRITEMODBUS, WRITEMB2, STATUS(2), SETSYSTEM

12.43 REFRESH

Purpose : To Force immediate refresh of the physical inputs and outputs. This can be used after executing a SETBIT or CLRBIT command on an output[n] variable and to force the physical output to change immediately (subject to I/O refresh time delay). Otherwise, the physical output will only be updated during the normal refresh cycle which will occur only at the end of every ladder logic scan.

This is useful for situations that require immediately action such as shutting down a load during an emergency. This command is likely to be used mainly by an Interrupt CusFn.

Comments : The REFRESH command takes approximately 0.5ms to complete on the Nano-10 and F-series PLCs, 1ms on the FMD series PLCs, and up to 2ms on the legacy M-series PLC. Note that if any expansion boards are connected to the FMD series or F-series PLCs, it could also take up to 2ms to complete. If used frequently, it could increase the program scan time significantly.

12.44 REM (or ') Statement

Purpose : To allow explanatory remarks to be inserted in a program. The text after the REM statement until the end of the line will be ignored by the compiler. An abbreviation for the REM statement is the apostrophe (')

Examples :

```
REM    Waiting for the right time to turn on
'    This is also a remark line.
```

12.45 RESET

Purpose : To perform a software reset of the PLC from within a CusFn. All the variables will be reset to zero or inactive and all the hardware outputs such as DAC and PWM will be turned OFF. The effect is the same as the Master Reset [MaRST] function in the ladder logic. The first scan bit (1st.Scan) will also be turned ON for one scan time.

A RESET will also perform a FLASH memory backup for all NANO-10 and FMD series PLCs. This is especially important after any SETTIMERSV, SETCTRSV, or SET_IPADDR commands are executed because the timer and IP data must be backed up into FLASH. However, it is better to use the SETSYSTEM 252,0 command to perform a FLASH backup since a RESET will cause all volatile data to be lost.

Comments : If the program is stuck at some dead loop (such as WHILE, FOR-NEXT) in a CusFn, then [MaRST] would not be executed since the ladder program would not have a chance to scan the ladder rung containing the [MaRST] function. If this command is used by an interrupt service function, then it is possible to get the system out of the dead loop since the interrupt function can interrupt the dead loop and reset the PLC.

12.46 RESTART REBOOT

Purpose : To perform a cold reboot of the PLC from within a CusFn. The effect is identical to a power-on-reset except that you execute this command without turning OFF power to the PLC.

Comments : RESTART may be necessary for some changes to take effect, such as for the PLC to assume new IP address after executing the SET_IP command.

Note : This command is ignored by i-TRiLOGI simulator.

12.47 RETURN

Purpose : Unconditionally ends the execution of the current CusFn and return to the caller (which is either the ladder program or another CusFn which has executed a CALL command).

Use of the RETURN statement is optional if there is no conditional ending required. After executing the last statement the CusFn will return to the caller automatically.

See Also : CALL

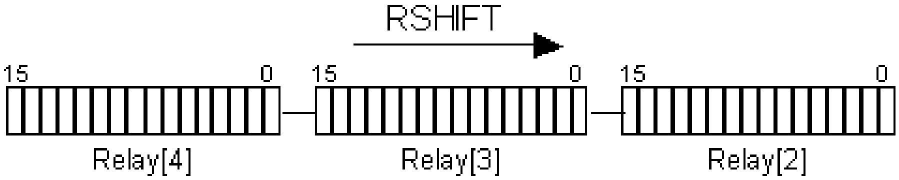
12.48 RSHIFT i,n Statement

Purpose : To shift the integer variable i 1 bit to the right. i must be either an integer variable, a DM[n] or a system variable such as relay[n], output[n], etc.

RSHIFT instruction permits more than one variable to be chained together before performing a bit shift. The parameter n indicate the number of channels to be chained starting from i upward. n =1 if only one variable is involved.

Examples : RSHIFT relay[2],3

Comments : The relay channels #2,#3, and #4 (which represent relays number #17 to #64) are chained together in the following manner:



Bits are shifted from the upper channel towards the lower channel. Bit #0 of Relay[4] will be shifted into Bit #15 of Relay[3] and so on. Bit #0 of the lowest channel Relay[2] will be lost.

See Also : LSHIFT

12.49 SAVE_EEP data, addr
SAVE_EEP8 data, addr8
SAVE_EEP16 data, addr16
SAVE_EEP32 data, addr32

Note : SAVE_EEP8, SAVE_EEP16, and SAVE_EEP32 are only available on PLCs with firmware >= r78. SAVE_EEP is equivalent to SAVE_EEP16 and is available on all firmware versions.

Purpose : To store an 8-bit, or 16-bit, or 32-bit integer in the user's definable EEPROM/FRAM space at address *addr8*/addr16/addr32 for non-volatile storage.

Comments : SAVE_EEP8

If you attempt to save a 32-bit data, only the lower 8 bits will be saved.

SAVE_EEP / SAVE_EEP16

To save the entire 32 bits of data,

a) If your PLC firmware is < r78, then save the upper 16-bit using the GETHIGH16( ) function and the lower 16-bit directly in two separate locations.

b) If your PLC firmware is >= r78, use SAVE_EEP32 command to save 32-bit variable to the EEPROM/FRAM space.

Byte, Word and Long Word Addressing: Note that the FRAM space for 8-bit, 16-bit and 32-bit SAVE_EEPXX and LOAD_EEPXX commands are the same space and they are mapped as shown below

SAVE_EEP32	SAVE_EEP16	SAVE_EEP8
1	1 and 2	1, 2 3 & 4
2	3 and 4	5,6,7,8
…	…	…
N	2N-1 & 2N	4N-3, 4N-2, 4N-1, 4N

If you intend to use the EEPROM/FRAM space to store more than 1 type (8, 16 or 32-bit) of data then you must properly manage the EEPROM/FRAM data space reserved for storing each type of data so that they don't overwrite each other address space. The easiest way is always to reserve enough space for 32-bit data, followed by space reserved for 16-bit and/or 8 bit data.

E.g. If you have 6000 words (16-bit) of FRAM space - this means you have space for storing 6000*2 = 12000 bytes of 8-bit data and you only have 6000/2 = 3000 long words for storing 32-bit data.

Now assume that your program needs to store up to 500 x 32-bit data and the balance for storing 8 bit data.

If you reserve 500 long words for 32-bit data, that means you are using up 500*4 = 2000 bytes of the FRAM space. This leaves you 12000 - 2000 = 10000 bytes of FRAM space for storing 8-bit data starting from byte address space 2001 to 12000.

Please refer to your PLC's reference manual for the upper limit of available FRAM/EEPROM storage.

Parameters :

data - May be an integer constant or variable.
addr - EEPROM/FRAM address for data storage.

Examples :

```
save_EEP relay[1],100
save_EEP8 relay[1],100
save_EEP16 relay[1],100
save_EEP32 relay[1],100
```

See Also : LOAD_EEP( ), GETHIGH16( ), SETHIGH16, LOAD_EEP$( ) and SAVE_EEP$

12.50 SAVE_EEP$ strdata, addr

Note : Applicable to all Nano-10, FMD Series, and F-series, but only to the legacy T100MD/MX PLC with firmware r44 or higher.

Purpose : To store a string strdata in the user's definable EEPROM address addr for non-volatile storage.

Parameters :

strdata - may be any string constant or string variable.
addr - FRAM/EEPROM address (1,2,3...). Please refer to your PLC's reference manual for the upper limit of FRAM/EEPROM space.

Implementation: Save_EEP allows you to save "strings" into the non-volatile data area of the PLC, which could be FRAM, FLASH, or EEPROM (legacy M-series PLCs only). This data space is divided into 40-byte chunks for string storage. I.e. regardless of the length of the string, each string storage location will occupy a fixed 40-character length. Hence if "stringdata" parameter is longer 40 characters, then only the first 40 characters will be stored in the memory space and the remaining characters will be discarded.

The string and integer data actually share the same pool of data space. However, the string spaces are allocated from the top of the data space downward, while the integer spaces are allocated from the bottom of the data space and grow upward. This implementation allows say both SAVE_EEP n, 1 and SAVE_EEP$ x$, 1 to be executed in the same program without the string and integer data writing over each other space.

However, when the addresses grow larger up to a certain point, the integer and string data space will cross path and overwrite each other's space. It is therefore the programmer's responsibility to check that this does not happen. Here is how:

Assume the total data space for integer data = N words (16 bit).
Total number of data space = 2N bytes
=> Maximum number of string data space = 2N/40 (rounded down).

To determine the upper limit of one type of storage, you have to first decide how much space you want to allocate to the other type.

E.g. 1 : N = 1700, and you want use the first 510 location for integer data, that means the maximum number of string space available = (1700–500)*2/40 = 59.

E.g. 2 : N = 7700, and you want to store 200 strings. The maximum number of integer space available = (7700*2 – 200*40)/2 = 3700.

Example : save_EEP$ A$,3

The content of A$ will be stored at string space #3 of data EEPROM.

See Also : LOAD_EEP$( )

12.51 SETBAUD ch, baud_no

Purpose : To set the communication "Baud Rate" of the PLC's serial channel #ch. All of the 'Super' series PLC serial ports are defined as 8 data bit, 1 stop bit, and no parity and each has been preset to a certain default baud rate, which the PLC will assume every time its powers up. The baud rate may or may not be changed, depends on the PLC model. Please refer to the PLC's User's manual for the baud_no that represent the baud rate of each serial channel and the range of baud_no each of these serial ports may assume.

Caution should be taken when programming the baud rate of the "Host link" port because if a wrong baud value is set the host PC may not be able to communicate with it. If this happens, suspend the PLC using its hardware switch (such as DIP switch #4 on the FMD or F-series PLCs and jumper J4 on the Nano-10) and re-load the program with the correct settings. Consult chapter 1 of your PLCs reference manual for more details on regaining serial communication.

Examples : SETBAUD 3,3 ‘ Set serial port #3 to 9600.

12.52 SETBIT v,n

Purpose : To set the bit #n of the integer variable v to '1'. v may be any integer variable or a system variable such as relay[n], output[n], etc.

The range of n depends on the PLC firmware version, as follow:

PLC Firmware	**Before r78**	**r78 or later**
n	0 to 15	0 to 31

n is an integer constant or variable of value between 0 and 15 or between 0 and 31. Following digital electronics convention, bit 0 refers to the least significant bit (rightmost bit) and bit 15 the most significant bit (leftmost bit) of the 16-bit variable and bit 31 is the most significant bit of a 32-bit integer variable.

For older PLC firmware (before r78), SETBIT only works on the lower 16 bits.

Please refer to page 10-4 in "Chapter 10: TBASIC Statements, Functions, Operators and Variables" for the mapping between I/O bits and the variables.

Examples : SETBIT output[2],11

Comments : output #28 will be turned ON.
(Output channel #2 bit #11 = Output #17 +11 = 28)

See Also : CLRBIT, TESTBIT( )

12.53 SetCtrSV n, value

Purpose : Change the Set Value (S.V,) of the Counter #n to value . This statement to allow the user to modify the S.V. of the PLC internal counters without changing the source program. A TBASIC function can be written easily to make use of a few digital or analog inputs to modify the SV of these internals timers/counters.

Parameters :

N - should be between 1 and 128.
Value- should be between 0 and 9999.

Examples : SetCtrSV 10,1234
SetCtrSV 3, GetCtrSV(3)+10

Counter #10 will assume a S.V. of 1234..
S.V of Counter #3 will be increased by 10.

Comments : For F-series PLC the new S.V is stored in the onboard FRAM and hence is non-volatile. (See sample program "set_TCSV.PC4").

For Nano-10 and FMD PLC, the new S.V. are only stored on RAM-shadowed CPU flash memory and is normally volatile unless a RESET or SETSYSTEM 252, 0 command has been executed which forces a backup of the RAM shadow to the CPU flash memory. Programmer should therefore use this command sparingly. Note that TBASIC custom funciton can start a counter simply by setting its CTRPV[] variable to any integer value between -1 (reset counter) and 9999 so it is not necessary to change the counter S.V..

The present values (P.V.) of counters can be read or written directly as integer variables "CtrPV[n]". But the Set Values can only be changed by this function.

See Also : GetCtrSV( ), GetTimerSV( ) , SetTimerSV

12.54 SET_IPAddr dmindex

Purpose : To set the IP address of this PLC's built-in Ethernet port using the parameters stored in DM[dmindex] to DM[dmindex+4].

Comments : To change the IP address of a PLC's Ethernet port, you can load the new IP address and port number into the variables DM[100] to DM[104] (e.g. from a HMI panel) and then invoke a function that runs SET_IPAddr 100 to write the new IP address into the PLC.

For Nano-10 and FMD PLCs, you should soft-reset (using the RESET or SETSYSTEM 252, 0 command) and restart (cold boot) the PLC after changing the IP address so that the new IP address will be saved into the virtual EEPROM

The PLC will only assume the new IP address after it has been rebooted (via the RESTART command or power-on reset).

Example :

```
DM[100] = 192
DM[101] = 168
DM[102] = 1
DM[103] = 101
DM[104] = 9080

SET_IP 100     ‘ Set the IP address to 192.168.1.101:9080
```

Note : This command is ignored by i-TRiLOGI simulator.

12.55 SETDimmer ch, value

Note : Applicable only to F-series PLCs

Purpose : To set channel #channel of the F-series PLC's Light Dimmer Control outputs to fire a trigger pulse to opto-isolated TRIAC (or Solid State Relay).

Comments : The trigger pulse is fired after a time delay specified by the value parameter (each unit is equal to about 53 microseconds) and it is measured with respect to the zero crossing that occur at every half cycle of AC voltage.

Parameters :

ch = Channel.
Range = 1 to 12 (which maps to FPLC outputs #5 to 16)

value = Trigger Pulse Time Delay.
For 50Hz AC signal, range =1 to 188
For 60Hz AC signal, range =1 to 156

Note : when value = 1 the output is fully ON. When value is near is maximum value the output is fully OFF. Do not set value to larger than the maximum for each particular AC frequency.

Example : SETDIMMER 1,78

The PLC light dimmer output channel 1 (which corresponds to digital output #5 of F-series PLC) will send a short trigger pulse to the SSR to turn it on at 78 x (internal timer tick of 53 microsecond per unit) = 0.00413second from the last zero crossing. For a 60Hz AC this will occur at about 90 degree phase angle since the time duration of half a 60Hz wave = 1/60/2 = 0.00833 seconds.

12.56 SetTimerSV n, value

Purpose : Change the Set Value (S.V,) of the Timer #n to value . This statement to allow the user to modify the S.V. of the PLC internal timers without changing the source program. A TBASIC function can be written easily to make use of a few digital or analog inputs to modify the SV of these internals timers.

Parameters :

N - should be between 1 and 128.

Value- should be between 0 and 9999.

Examples : SetCtrSV 10,1234
SetTimerSV 3, GetTimerSV(3)+10

Counter #10 will assume a S.V. of 1234..
S.V of Timer #3 will be increased by 10.

Comments : For F-series PLC the new S.V is stored in the onboard FRAM and hence is non-volatile. (See sample program "set_TCSV.PC4").

For Nano-10 and FMD PLC, the new S.V. are only stored on RAM-shadowed CPU flash memory and is normally volatile unless a RESET or SETSYSTEM 252, 0 command has been executed which forces a backup of the RAM shadow to the CPU flash memory. Programmer should therefore use this command sparingly. Note that TBASIC custom funciton can start a timer simply by setting its TIMERPV[] variable to any integer value between -1 (reset timer) and 9999 so it is not necessary to change the timer S.V..

The present values (P.V.) of timers can be read or written directly as integer variables "TimerPV[n]". But the Set Values can only be changed by this function.

See Also : GetCtrSV( ), GetTimerSV( ) , SetCtrSV

12.57 SETDAC n, x Statement

Purpose : To set channel #n of the PLC's Digital-to-Analog Converter (DAC) with the 16-bit integer result of the expression x. n must range between 1 and 16. Once set, the DAC channel will latch the set value until the next SETDAC statement on the same channel is executed.

Examples : SETDAC 5,A+B*16

Comments : DAC channel #5 will be set with the value of A+B*16. A run-time error will result if n is less than 1 or is greater than 16. The actual number of DAC channels depends on the PLC model in use.

12.58 SETHIGH16 v, data

Purpose	:	To assign the upper 16-bit of a 32-bit integer variable v to data. The lower 16-bit of v is unaffected. This can be used to construct the value of a 32-bit integer data using two 16-bit data obtained from either the EEPROM or the DM[n].
Examples	:	A = DM[2] SETHIGH16 A,DM[1]
Comments	:	If you are constructing the value of a 32-bit variable from two 16-bit data, then SETHIGH16 should be executed only after the lower 16-bit has been assigned to the 32-bit variable, as shown in the above example. If you were to execute A = DM[2] after the SETHIGH16 A, DM[1] statement, the SETHIGH16 operation would have been lost since the assignment statement itself overwrites all previous operation (including SETHIGH16) on variable A.
See Also	:	GETHIGH16( )

12.59 SETIO labelname

Note : -- Please refer to the definition of CLRIO command

12.60 SETLCD n, offset, x$

Purpose	:	To display the string expression x$ on Line #n on built-in alphanumeric Liquid Crystal Display (LCD) or compatible Vacuum Fluorescent Display (VFD).
Comments	:	x$ may be formed by concatenation of various strings using the ‘+’ operator (e.g. “Temp =”+STR$(A,3)+CHR$(223)+” C”). Integers must be converted to string using the STR$( ) or HEX$( ) function to be accepted by this function.

Special case: if n =0 the string x$ will be sent to the LCD’s “Instruction-Register” which allows hardware-specific LCD configuration such as clear screen, set cursor ON/OFF etc. (please refer to LCD's manual for details)

The parameter offset = 1 to 40 allows you to send the string x$ beginning from the offsetth position. Only the characters position to be occupied by x$ will be written to the display, other characters of the display remain unaffected.

The PLC may support LCD display modules capable of displaying up to 4 lines x 40 characters per line of alphanumeric characters. If the display has fewer lines or fewer characters per line, the unavailable lines or characters will be ignored by the PLC. Once set, the LCD display will latch the set value until the next SETLCD statement on the same line and same offset is executed. On the TRiLOGI simulator, the result of the SETLCD is displayed together with the Special Variables view screen.

Examples : SETLCD 1,1,"This is a 1x20 LCD Display"

12.61 SETPASSWORD string

Purpose : When this statement is executed, the PLC will not properly respond to any host link commands sent to it except the command “PWxxxx…xx” which must contains the same string “xxxx…xx” (not more than 19 characters) as defined in the SETPASSWORD command. All other commands will receive a “PWER” response indicating a “password error” state. Once the correct password has been accepted the PLC will work as normal and respond to all host link commands. Execution of “PW” host link command without any string will put the password lock back in force to prevent unauthorized access.

Example : SETPASSWORD “I love TRiLOGI”

When using TRiLOGI the software will automatically prompt you to enter the password string if it encounters a PLC which has been password-locked. Note that the password is case sensitive. Password locked PLC cannot be accessed by older version of TRiLOGI.

Comments : This feature is mainly used to protect an unattended PLC which is linked to an auto-answer modem. Without password protection anybody can dial in with a TLServer or TL41.exe and have full control of the PLC, which may be a serious security problem. Within the PLC software you may also use a timer to periodically re-arm the PLC with this command for maximum protection. You can also use different passwords for different time of the day or a set of rotating passwords to provide greater security.

If you are linking the PLC to the internest via i-TRiLOGI , you will notice that the TLServer already provides username/password authentication. However, the password here works independent of the access provided via TLServer, hence this command can still be used to provide an additional layer of security against unauthorized access to the PLC.

12.62 SETPROTOCOL ch, mode

Purpose: A Nano-10, FMD series, and F-series PLC (as well as the legacy T100M series) automatically senses the type of communication protocols sent to it and responds accordingly. You may however fix the protocol type so that it does need to check the protocol type before responding. This command also allows the PLC to be defined as “No Protocol” so that it will not automatically respond to data that it receives which appears like one of the supported protocols. This may be important in some applications in which the PLC serial port is used purely to receive incoming data via INCOMM and INPUT$ and you do not want it to respond to some data that appears to it as a valid communication protocols This is also useful for implementing user’s own communication protocol without worrying about conflict with the existing protocols.

Parameters:

ch = 1, 2 or 3 (COMM port number)

mode = 0 - Auto sensing (default mode)

1 - Fixed at RTU mode
2 - Fixed at EMIT mode
3 - Fixed at MODBUS ASCII mode
4 - Fixed at OMRON C20H protocol mode
5 - Fixed at NATIVE host link command mode
10- No protocol. (For creating user own custom protocol.

IMPORTANT :

i. Please note that if you set the protocol to other than the "Native" (mode=5) or "Auto" (mode=0) that serial port will no longer respond to commands from i-TRiLOGI and you will encounter "Communication Errors" when you try to perform any communication using i-TRiLOGI with that serial port. You can still use the other unaffected serial port (e.g. COMM3, RS485) that support host link commands.

ii. For all Nano-10, FMD series, and F-series PLC (as well as the legacy T100M series PLCs with firmware revision r42 or above), setting the protocol mode to NATIVE (mode=5) disables support for point-to-point communication. Also, the wildcard FCS (i.e. 00) is not allowed in multi-point format in this mode. This is to ensure the maximum data integrity by accepting only commands that are fully formatted with correct FCS error check. Note that this means i-TRiLOGI's "Detect ID" function will not work since the PLC with protocol mode =5 will not respond to point-to-point protocol "IR*". You will have to manually enter the ID into i-TRiLOGI login dialog box to communicate with the PLC.

iii. To regain communication with the serial port that has been changed by SETPROTOCOL command, you will have to execute another SETPROTOCOL function that set it to mode 0 or 5 (assuming it has been written into the program), or you must reset the controller by turning OFF the power and then ON again. If you execute a SETPROTOCOL using the 1ST.Scan then you must turn on DIP switch #4 before powering up the PLC so that the SETPROTOCOL command will not be executed and you can regain control of the PLC using i-TRiLOGI.

12.63 SETPWM n, x, y

Purpose : To set channel #n of the PLC's Pulse-Width Modulation (PWM) output with duty cycle represented by (x/100 %) and at a frequency (in Hz) given by parameter y.

Parameters :

n = PWM channel. Range = 1 to 8.
Once set, the PWM channel will latch the set value until the next SETPWM statement on the same channel is executed.

x = Duty Cycle. Range = 0 to 10000.
If x is more than 10000, the duty cycle will be set to 100%

y = Frequency. Range = 0 to 50000.
If y is more than 50000, the frequency will be set to 50khz

Examples : SETPWM 1, 4995, 2000

Comments : PWM channel #1 will be set to operate at 49.95% duty cycle for PWM that can resolve up to 0.01%. The actual resolution will depend on the PLC's PWM resolution. The PWM frequency is set to 2000 Hz or nearest.
For a 10-bit PWM the best resolution is about 1/1024 = 0.1 %. This means that in the above example the PWM will be rounded to 50%. Please check the target PLC's manual for the actual resolution.

12.64 SETSYSTEM n, data

Purpose : Allow changing of certain default system's parameters. Currently only the following data are defined. More parameters may be defined in future. Some commands are specific to a PLC model and these will be mentioned in the relevant section of the hardware manual.

n	***data***	**Description**

1	dd dd dd dd

12.64.1 Com Response Wait Time

The data value determines the # of wait states (multiple of 0.15s for serial and 0.20s for Ethernet*) to wait for a response from a slave controller after executing a NETCMD$, READMODBUS, WRITEMODBUS, READMB2 or WRITEMB2 command.

*Note:
For PLCs with firmware before r77, the Ethernet port wait time is a multiple of 0.5s instead of 0.2s, which is for PLCs with firmware r77 or higher. The serial wait time is 0.15s for all firmware versions.

For Nano-10, FMD and F-series PLCs Only

SETSYSTEM 1, data affects the 4 COMM ports (COMM4 is Ethernet client connection) individually, where bit 0 & 1 are for COMM1, bit 2 & 3 are for COMM2 , bit 4&5 are for COMM3 and bit 6&7 are for COMM4.

By default data is set to *&H55* (binary 01 01 01 01), which means a 2 wait multiple is used for all 4 COMM ports. This is about 0.30s for all 3 serial COMM ports, and about 0.40s the Ethernet port (PLCs with r77 or higher firmware).

Bit #	76	54	32	10
data	dd	dd	dd	dd
Comm port	4	3	2	1

dd =	00	01	10	11
# of Retries	1	2	3	4

Note: maximum = 4 retries

In order to change the wait states for COMM3 you have to set the bit 4 & 5 of data.

E.g SETSYSTEM 1, &HFF

This will change all 4 COMM ports to wait for 4 cycles (0.60s for serial port and 0.80s for PLCs with firmware r77 or

1	d	higher) For T100MD+ and T100MX+ *data* (d) is applied to all 3 COMM ports simultaneously and can be any value from 0 to 255 e.g. SETSYSTEM 1, 3 The PLC will wait 3 x 150ms = 450ms for a valid response from the slave controller on all 3 serial ports.
2	dd dd dd dd	**12.64.2 Number of Com Retries** # of retries if NETCMD$, READMODBUS, WRITEMODBUS, READMB2 or WRITEMB2 failed to get a valid response from slave controller. For Nano-10, FMD and F-series PLCs SETSYSTEM 2, data affects the 4 COMM port individually (COMM4 is Ethernet client connection) , where bit 0 & 1 for COMM1, bit 2 & 3 for COMM2 , bit 4&5 for COMM3 and bit 6&7 for COMM4. By default data is set to &HAA (binary 10 10 10 10). This means 2 re-tries (a total of 3 tries) for all 4 COMM ports. data : dd dd dd dd Comm port : 4 3 2 1 Where dd = 00, 01, 10, or 11 (maximum = 3 re-try) In order to change the number of retries for COMM2 to 1 retry you have to set the bit 2 & 3 of data. E.g SETSYSTEM 2, &HA5 'binary 10 10 01 10 The above example set COMM2 to 1 retry and the other 3 ports are 2 retries.
2	d	For T100MD+ and T100MX+

		data (d) is applied to all 3 COMM ports simultaneously and can be any value from 0 to 255 e.g. SETSYSTEM 2,5 The PLC will retry up to 5 times if it failed to communicate with the slave. Note that there will be longer waiting times when failures occur if you increase the number of retries.
3	0	**12.64.3 Hostlink Command Response Time** *data = 0* : Respond as fast as possible to serial port hostlink commands received from the host computer or another PLC.
	1	*data = 1* : (default) at least a 0.01s (10ms) must lapse before responding to host link commands received from the host computer or another PLC. This delay is needed for auto-switch type RS485 converter to allow time for the hardware transceiver to switch direction.
4		**12.64.4 Configure Quadrature Encoding** Set Enhanced Quadrature Encoding for HSC channels (see the PLC's User manual on High Speed Counter hardware for details)

5	0 d	**12.64.5 DM[]s Cleared During Reset** *data = 0* : All DM cleared when PLC is soft-reset. *data = d* : DM #1 to #d will not be cleared when the PLC is soft-reset. NOTE: Applicable only to PLC with firmware r78 and above.
6	3 4 6 d	**12.64.6 Modbus Function Select** *data = 3* : READMODBUS and READMB2 use Modbus Function 03. WRITEMODBUS uses default (function 16) *data = 4* : READMODBUS and READMB2 use Modbus Function 04. WRITEMODBUS uses default (function 16) The following is applicable for PLC with firmware r77 and above only: *data = 6* : WRITEMODBUS uses Modbus Function 06. READMODBUS and READMB2 uses default (function 03) *data = d* : Any other values of data (d) will set READMODBUS and WRITEMODBUS back to default (03 for READ, 16 for WRITE)
7		Reserved
8	d	**12.64.7 Change ID** Allow the PLC to change its own ID from within TBASIC temporarily to the lower 8-bit value provided in the "data" parameter for individual COMM port. The new ID is volatile. It does not overwrite the default PLC ID which can only be changed from a host program using the "IWxx" command. SETSYSTEM 8, n COMM1: d = &H0001 to &H00FF

		COMM2: d = &H0101 to &H01FF COMM3: d = &H0201 to &H02FF COMM4 (Ethernet): d = &H0301 to &H03FF ID for the respective COMM port will be set to 01 to FF after running the above command. (only applicable to PLC with firmware r76 and above)
9	1 0	**12.64.8 Fast Refresh** *data = 1* : to enable Fast I/O REFRESH ONLY *data = 0* : (default) REFRESH all i/os On FMD PLCs the fast I/Os are input 3 to 6 and output 5 to 8. On F-series PLCs, the fast I/Os are inputs 1 to 12 and outputs 5 to 16. All Nano-10 I/Os are fast I/Os so this command has no effect. By running SETSYSTEM 9,1 - only the fast I/Os are refreshed when TBASIC REFRESH command is run. This has the advantage of reducing the REFRESH latency (especially when executed within a interrupt service custom function).

11	0 1 to +127 -1 to -127	**12.64.9 Calibrate RTC** Add or subtract data seconds per day from the RTC to calibrate it for accuracy. E.g. if the clock is 3 seconds too slow everyday, then run SETSYSTEM 11, 3 once during 1st.Scan and the CPU will increase the RTC by +3 seconds every time when it past midnight. data = -127 to +127 to adjust the RTC for accuracy after midnight. *data = 0* : (default), no adjustment *data > 0* : RTC crossover from 23:59:59 to 0:0:data directly and bypasses 0:0:0. E.g. data = 5, clock is adjusted faster by 5 seconds. *data < 0* : RTC stop at 23:59:59 and wait until (-data) seconds later then cross over to 0:0:0 E.g. if data = -5, the RTC is slowed by 5 seconds. Applicable only to PLC with firmware r78 and above
16	3 to 16	**12.64.10 Number of I/O to Support** Total number of 8-bit digital I/O channel supported. *data = d* => (d x 8) inputs and (d x 8) outputs. E.G. data = 16 (default) => 16 x 8 = 128 Inputs and 128 outputs. If you don't use expansion board, and if you run SETSYSTEM 16, 3 then only inputs 1-24 and outputs 1-24 will be accessed during I/O scan. This can reduce the I/O scan time. Otherwise the default is to scan all 128 inputs and 128 outputs which consumes 0.5ms on an F-series PLC.
17	0	**12.64.11 Enable IR Remote** *data = 0* : disable IR Remote (default)

	1	*data = 1* : enable IR Remote Note: Only applicable to F-series PLCs (F2424 and F1616-BA). See users manual for more details.
20	0 1 2 to 255	**12.64.12 Pulse Measurement Resolution** *data = 0* : PULSEPERIOD and PULSEWIDTH 1 to 4 returns data with 0.1us resolution *data = 1* : PUSLEPERIOD and PULSEWIDTH 1 to 4 returns data with 1us resolution. (For PLC firmware r77 and above) *data = 2 to 255* : Number of seconds of inactive incoming pulses before PULSEWIDTH and PULSEFREQUENCY assume that incoming pulses has stopped. Default is 5 seconds.
22	0 1	**12.64.13 Dimmer Interrupt Enable** *data = 0* : Disable Dimmer interrupt *data = 1* : Enable Dimmer interrupt
24	UB LB	**12.64.14 Quadrature Encoder Config** data is defined as a two-byte integer for configuring the HSC channel to be simple, enhanced x1, enhanced x2 or enhanced x4 count. (Applicable only to PLC with firmware r72 and above). Upper byte (UB): channel number (&H01 or &H02 or &H03) Lower byte (LB) : &H00 (default) =simple ; &H01=x1; &H02=x2; &H03= x4 View the HSC chapter of your PLCs reference manual for more information.
25		**12.64.15 Analog Output Range**

		Applicable only to FMD PLCs for selecting the analog output type (0-5V or 0-10V). Please refer to Chapter 5 of the PLC's User Manual.
251	0 to 5 11	**12.64.16 Close TCP Connection** To force close an an incoming TCP server connection. *data = 0 to 5* : Force-close a FServer and Modbus/TCP server connection *data = 11* : Force-close a FTP Server connection. Applicable only to PLC with firmware r78 and above
252	0	**12.64.17 Write to Flash** *data = 0* : On Nano-10 and FMD PLC without the FRAM-RTC, this command is used to force save the emulated EEPROM content to the CPU Flash memory. **Important:** This command should be executed every time a SETTIMERSV, SETCTRSV, or SET_IPADDR command is used so that the data is properly backed up to PLC FLASH memory. This must be done for all Nano-10 and FMD series PLCs with or without any FRAM-RTC module installed.
253	0 1 2	**12.64.18 Control the RTC operation.** *data = 0* : Completely disable the RTC on the CPU *data = 1* : (default) Enable the RTC on the CPU *data = 2* : Disable hourly auto update of RTC data from FRAM-RTC (useful on systems that rely on RTC update from Internet Time Server) You can also use this command to setup the date and time for an RTC alarm interrupt to occur. The interrupt handler can be defined using

253	1hh:mm:ss	INTRDEF 19, n,1 ' where n is the custom function to handle the alarm interrupt. The time and date when the alarm interrupt occurs can be defined using SETSYSTEM 253,data as follow: *data = 1hhmmss* : set system alarm to occur when the RTC reaches the time "hh:mm:ss" E.g SETSYSTEM 253, 1130500 ' alarm at 13:05:00 When SETSYSTEM 253, 1hhmmss is run the alarm is set on the day when the RTC next reaches hh:mm:ss (so it could be the same day or the next day depending on current time). Applicable only to PLC with firmware r72 and above
254	0 1 2	**12.64.19 Ethernet Port Enable** *data = 0* : Disable Ethernet port to save power - execute only when Ethernet connector is disconnected. *data = 1* : (default) Enable Ethernet port. *data = 2* : Clear ARP cache so that it will request ARP data before connection. Applicable only to PLC with firmware r74 and above. Note - Do not disable Ethernet port when Ethernet connector is plugged into the switch or router. This is because if Ethernet is disabled while active connection is going on, the CPU can lock itself into undefined Interrupt error state.
255		**12.64.20 Watchdog Timer** This allows you to enable on-CPU Watch Dog Timer.

<table>
<tr>
<td>255</td>
<td>&HB3
&HAB
&H03
&H00</td>
<td>
<table>
<tr><th>data</th><th>All F-series</th><th>FMD & Nano-10 >= r79 firmware</th><th>FMD & Nano-10 <= r78 firmware</th></tr>
<tr><td>&HB3</td><td>Time out in 2.24s</td><td>Time out in 2.67s</td><td>Time out in 4.47s</td></tr>
<tr><td>&HAB</td><td>Time out in 0.14s</td><td>Time out in 0.17s</td><td>Time out in 0.28s</td></tr>
<tr><td>&H03</td><td>Diable WDT (default)</td><td>Diable WDT (default)</td><td>Diable WDT (default)</td></tr>
<tr><td>&H00</td><td>Reset WDT</td><td>Reset WDT</td><td>Reset WDT</td></tr>
</table>

If enabled and the CPU goes astray because of noise-induced trouble, the WDT will reset the CPU when it times out. Note that to simplify user's application the TBASIC O/S automatically resets the WDT during its normal execution except when the program is stucked inside a GOTO loop. Hence if your program is likely to spend longer time within a GOTO loop than the predefined timeout, you must run the SETSYSTEM 255, 0 command periodically to reset the WDT before it times out, to prevent the WDT from resetting the CPU. However, the WDT will not kick in if your program is stucked inside a FOR..NEXT or WHILE.. ENDWHILE loop. WDT also will not activate if your program encounters undefined interrupt error in order to give the programmer a chance to investigate the cause that leads to the Undef. Interrupt error.

Applicable only to PLC with firmware r74 and above.
</td>
</tr>
<tr>
<td>256</td>
<td>0
1</td>
<td>**12.64.21 I/O Scan Interrupt**

data = 0 : (default) do not process user interrupt during I/O scan interrupt latency determined by scan time.

data = 1 : process pending user interrupt even during I/O scan (to minimize interrupt latency).

Available to PLC with firmware r50 and above.
</td>
</tr>
</table>

12.65 SLEEP

Purpose : To put the PLC CPU into sleep mode to save power.

Comments : Upon execution of this command the CPU will enter a low power sleep mode but it can be woken up by a serial interrupt or an active I/O interrupt. However, this will only save power consumed by the CPU and does not save the static power required by other components on the PLC circuit board.

Note:

1. Command available only to PLC with firmware r67 and above.
2. Do not execute this command with active Ethernet connection as it can trigger the CPU to lock up into undefined interrupt error state. Only application that are not connected to the Ethernet should use this command.
3. This command is ignored by i-TRiLOGI simulator.

12.66 STATUS (n)

Purpose : Return the status of various system operations. Returned values may be dependent on the PLC model.

STATUS(n)	RETURNED VALUE (v)	DESCRIPTION
		Reset Status
1	0	v = 0 : Normal power on reset
	1	v = 1 : Reset by Watch Dog Timer (WDT)
		Command Status Used to check the status of a command sent to a device.
2	0	v = 0 : command failure

	1
	255
		v = 1 : command successful
		v = 255 : End of File reached
		Applies to the following TBASIC commands or tags (sent via PRINT command), except 255 will only be returned during file access. See the Extended File System chapter of the user manual for more information.
		READMODBUS, WRITEMODBUS, READMB2, WRITEMB2
		I2C_WRITE, I2C_READ
		<READ>, <WRITE>, <APPEND> tags
		Example :
		IF STATUS(2) ' MODBUS READ/WRITE OK
		  …
		ENDIF

3	0 1	v = 0 : TCP/IP connection closed or last TCP operation failed v = 1 : TCP/IP connection established or last TCP operation successful
4	0 1 2 3	**FTP transfer status** v = 0 : FTP client was idle or last FTP failed v = 1 : FTP data transfer just started v = 2 : 1st FTP segment transferred, now transferring the rest v = 3 : FTP data transfer completed. Note: only for PLC with firmware r77 and above
5	78 to xx	**Return the main firmware version** xx can be anything higher than 78 E.g.firmware r78A will return 78. Note: only for PLCs with firmware r78 and above

<table>
<tr>
<td>8</td>
<td>0 to 255</td>
<td>
<p><b>Return PLC ID</b></p>
<p>PLC’s ID address stored in FRAM/EEPROM for host communication. Returned as an integer in the range 0-255 decimal or 00-FF Hex.</p>
<p>Note: if the ID is changed, a cold boot (power cycle) is required for STATUS(8) to return the new ID. Otherwise the previous ID is returned.</p>
</td>
</tr>
<tr>
<td>14</td>
<td>0 to 64</td>
<td>
<p><b>Ethernet Connection Status</b></p>
<p>A byte is returned and each of the first 7 bits represent a connection state for an incoming or outgoing Ethernet connection. Bit7 (MSB) is not used and can be ignored.</p>
<table>
<tr><td><b>Bit#</b></td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td><b>State</b></td><td>1/0</td><td>1/0</td><td>1/0</td><td>1/0</td><td>1/0</td><td>1/0</td><td>1/0</td></tr>
<tr><td><b>Connection Type (Default)</b></td><td>Client</td><td>MDBTCP</td><td>MDBTCP</td><td>MDBTCP</td><td>FSERVER</td><td>FSERVER</td><td>FSERVER</td></tr>
</table>
<p><u>Note:</u>
Bit 6 will always represent the PLCs 1 client connection (outgoing), bits 4 and 5 will always represent Modbus TCP (incoming), and bit 0 will always represent an FSERVER connection (incoming). Bits 1,2, and 3 can be set to either type (view chapter 2 of the PLC user manual for more details). By default, there are 3 incoming connections for each type.</p>
<p>If any bit is set to ‘1’, that means the corresponding connection type is being used and is <u>not</u> available.</p>
<p>Any bits set to ‘0’ are free to be used (available connections).</p>
<p>Example:
STATUS(14) returns 11 => bits 0, 1, and 3 are on</p>
</td>
</tr>
</table>

		Comment: 2 of 3 incoming FSERVER connections are used. 1 of 3 incoming MDBTCP connections are used. Note : only for PLC with firmware r78 and above
17	IR Code	**Return IR Code** Specific only to F-series PLC with Infra-Red decoder. Returns the decoded IR remote code. See chapter 17 of the user manual for more details. Note: the *SETSYSTEM 17, 1* command must be executed first to enable the IR decoder.
18	000000 to 235959	**Return Time as Integer** Return current Real Time Clock value as a 6 digit integer from 000000 to 235959 Must be returned to a 32-bit variable. Example: At 00:12:05 => A = STATUS(18) returns 001205 to A. At 12:00:05 => A = STATUS(18) returns 120005 to A. Note: Only for PLCs with firmware r72 and above.
19		**Return File Length** Return the file length of the currently opened file. Example: The currently open is 50kb => A = STATUS(19) returns 50000 to A. See the Extended File System chapter in the PLC user manual for more information. Note: only for PLCs with firmware r77 or above
20	>= 0	**FTP transfer status** *v >= 0* : Number of bytes uploaded to FTP Server. Transfer is in progress.

	< 0	*v < 0* : Total number of bytes uploaded. Transfer completed. Example: If 2,345 bytes have been uploaded to the server and the transfer has ended => A = STATUS(20) returns –2345 to A. See the Extended File System chapter in the PLC user manual for more information. Note: only for PLCs with firmware r77 or above
21		**Return CPU Counter Value** Can be used to measure execution time. This function returns the 32-bit value of a 10MHz free running counter in the CPU. The difference between the values from two STATUS(21) gives the elapse time of a process in 0.1us. E.g: A = STATUS(21) CALL [somefunction] B = STATUS(21) - A B is the execution time of [somefunction] in 0.1 microseconds. (i.e. if B = 10000 => 1 ms) Note: only for PLCs with firmware r78 or above

12.67 STEPCOUNT (ch)

Purpose : While the stepper motor controller is sending out pulses, this function can be used to monitor the number of stepper pulses sent to the Stepper Motor Channel #ch since the execution of the last “STEPMOVE” command. Hence this function returns the relative number of step moves.

This function can also be used to “measure” the physical size of a part if we use the stepper motor to drive a sensor and use the STEPSTOP command and the interrupt input to halt the stepper motor when the edges are detected. The physical size is then computed using the number of steps the stepper motor travels from one edge to another edge. The center position can be easily determined using such data too.

See Also : STEPMOVE, STEPMOVEABS, STEPCOUNTABS(), STEPSPEED, STEPSTOP, STEPHOME

12.68 STEPCOUNTABS (ch)

Purpose : Returns the absolute position of the stepper motor #ch. This function returns a zero if a STEPHOME command had just been executed and the stepper has not been moved since.

12.69 STEPHOME ch

Purpose : Set the current position counter of stepper # ch to zero. This indicates a new “Home” position of that stepper motor. This command should be executed only when the stepper has reached a particular position to be regarded as the home position. All STEPMOVEABS command executed subsequently will be relative to the defined home position.

12.70 STEPMOVE ch, count, r

Purpose : To activate the PLC's built-in stepper motor pulse generator channel #ch to output count number of pulses. The speed and acceleration parameters for the motion is defined by the STEPSPEED statement on the same channel # ch, which must be executed at least once before the first STEPMOVE command is issued. After executing the STEPMOVE command the PLC hardware will take over the actual pulse generation operation. The user’s program will continue to execute even though the pulse generation is not yet completed. The internal relay #r can be used to signal to the other parts of the ladder program regarding the status of pulse generation, as follow:

Parameters :

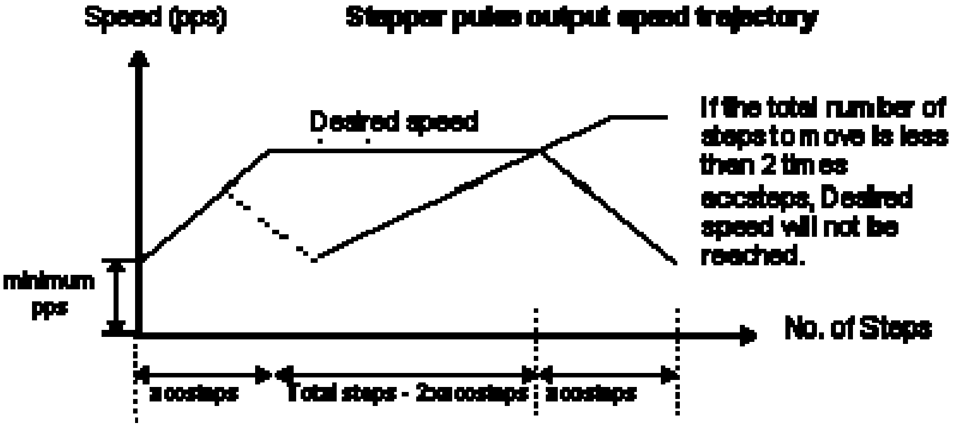
ch = Stepper Channel. Range = 1 to 8

count= Step Count. Range = -2^{31} to $+2^{31}$

r = Move Complete Relay Bit. Range = 1 to 512

Comments : When STEPMOVE command is first executed, the internal relay #r will be cleared before the first pulse is sent. After the completion of the movement (i.e. when all the pulses have already been sent), the relay #r will be set.

Count is a 32-bit integer number which allows you to program the stepper motor to move from -2^{31} to $+2^{31}$.(i.e. about ± 2 x 10^9) steps. Count can also be an integer variable A-Z. However, if you use a 16-bit variable such as DM[n] for count then the range of movement can only be between 1 to 32,767.



Pulse generation can be interrupted by issuing a STEPSTOP command in another CusFn, which may occur in cases when the hardware hits a limit-switch and must stop the motor immediately.

Important: When a stepper channel is already activated (i.e. mid-way through its pulse generation) repeat execution of STEPMOVE command on the same channel will be ignored by the PLC. Re-execution of the STEPMOVE command on this channel can only take effect after the channel's pulsing operation has been completed by itself or aborted by the STEPSTOP command.

When in TRiLOGI simulation mode, execution of the STEPMOVE command will bring up a pop-up window that displays all the parameters of the motion path.

Examples : STEPMOVE 1,5000,10

Send out 5000 pulses on channel 1 and at the end of motion turn ON relay #10.

See Also : STEPMOVEABS, STEPCOUNT( ), STEPCOUNTABS(), STEPSPEED, STEPSTOP, STEPHOME

12.71 STEPMOVEABS ch, position, r

Purpose : This command allows you to move the stepper motor # ch to an absolute position indicated by the position parameter.

Parameters :

ch = Stepper Channel. Range = 1 to 8

position= Step Count. Range = -2^{31} to $+2^{31}$

r = Move Complete Relay Bit. Range = 1 to 512

Comments : At the end of the move the relay # r will be turned ON. Position can be between -2^{31} to $+2^{31}$.(i.e. about $\pm 2 \times 10^9$). The absolute position is calculated with respect to the last move from the “HOME” position. (The HOME position is set when the STEPHOME command is executed). The speed and acceleration profile are determined by the STEPSPEED command as in the original command set.

This command automatically computes the number of pulses and direction required to move the stepper motor to the new position with respect to the current location. The current location can be determined at any time by the STEPCOUNTABS( ) function.

Once STEPMOVEABS command is executed, re-execution of this command or the STEPMOVE command will have no effect until the entire motion is completed or aborted by the STEPSTOP command.

See Also : STEPCOUNT( ), STEPCOUNTABS(), STEPSPEED, STEPMOVE, STEPSTOP, STEPHOME

12.72 STEPSPEED ch, pps, acc

Purpose : To set the speed pps and acceleration/retardation acc parameters for the PLC's stepper motor motion controller (pulse-generator) channel #ch.

Parameters :

ch = Channel. Range = 1 to 8.

pps = Speed. Range = 1 to 10000.
Set the target no. of pulses per second (pps) output by the pulse generator.

Acc = Acceleration Steps. Range = 1 to 2^{32}.
The acceleration ‘acc’ determines the total number of steps taken to reach full acceleration from standstill and the number of steps from full speed to a complete stop.

Comments : The stepper motor calculates and performs the speed trajectory according to these parameters when the command STEPMOVE is executed.

STEPSPEED command should be executed at least once before executing any subsequent STEPMOVE command to control the pulse generation. The defined parameters will be remembered until another STEPSPEED statement operating on the same stepper channel is executed again.

Examples : STEPSPEED 2,2000,20

The PLC's Stepper motor controller channel #2 is configured to send out pulses at 2000 pulses per second when STEPMOVE instruction is executed. It follows a linear acceleration trajectory which takes 20 pulses to reach the full speed of 2000 pps. This is equivalent to an acceleration of:

$$a = \frac{V^2}{2S} = \frac{2000^2}{2x20} = 100{,}000 \text{ pulse/s2}$$

12.73 STEPSTOP ch

Purpose : To abort a stepper channel #ch that is in motion due to exceptional circumstances (NOT intended for normal motor control operation).

Examples :

```
STEPSTOP 2
```

Important : Motion aborted by STEPSTOP command will not trigger the end-motion relay #r specified in the STEPMOVE command.

See Also : STEPCOUNT, STEPSPEED, STEPMOVE

12.74 STR$(n)
STR$ (n, d)

Purpose : To return a string that represents the decimal value of the numeric argument n. If the second format is used then this function will return a string of 'd' number of characters.

Examples :

```
A$ = STR$(-1234)
B$ = STR$(-1234,7)
```

Comments : A$ will contain the string : "-1234" , B$ will contain the string "-001234"

12.75 STRCMP(A$, B$)

Purpose : Perform a comparison between its two string expressions A$ and B$. IF A$ and B$ are equals, STRCMP will return a 0, if A$ is of lower order (in ASCII table order) than B$ the function will return a negative value. Otherwise it returns a positive value.

Examples :

```
IF STRCMP(A$, B$)=0 THEN
        STEPMOVE 1,1000,1
ENDIF
```

Comments : IF A$ and B$ are the same then turn on the stepper motor #1.

12.76 STRLWR$(A$)

Purpose : To return a string which is an all-lowercase copy of A$.

Examples : B$ = STRLWR$(A$)+Z$
C$ = STRLWR$(C$)

Comments : The second example shows how to convert a string to all lower case.

12.77 STRUPR$(A$)

Purpose : To return a string which is an all-uppercase copy of A$.

Examples : B$ = STRUPR$(A$)
C$ = STRUPR$(C$)

Comments : The second example shows how to convert a string to upper case.

12.78 TESTBIT (v, n)

Purpose : To return the logic state of bit #n of the variable v. The function returns 1 if the bit is '1', otherwise it returns 0.

Parameters :

v = Integer Variable or Register. May be any integer variable, however, if v is a 32-bit integer TESTBIT will only test the lower significant 16 bits.

n = Bit #. Range = 0 to 15.

Comments : Following digital electronics convention, bit 0 refers to the least significant bit (rightmost bit) and bit 15 the most significant bit. (leftmost bit) of the 16-bit integer variable. Please refer to page 10-4 in "Chapter 10: TBASIC Statements, Functions, Operators and Variables" for the mapping between I/O bits and the variables.

Examples : TESTBIT (Input[2],3)

To test whether input #20 is ON
(Input channel #2 bit #3 = Input 17 +3 = 20)

See Also : SETBIT, CLRBIT

12.79 TESTIO (labelname)

Note : Please refer to the definition of CLRIO command

12.80 TOGGLEIO labelname

Note : Please refer to the definition of CLRIO command

12.81 VAL(x$)

Purpose : To return a value of a decimal number contained in the argument x$.

Examples : B = VAL("123")*100

Comments : B should contain the value 12300

12.82 WHILE expression .... ENDWHILE

Purpose : To execute a series of statements in a loop as long as a given condition is true.

Syntax :

```
WHILE expression
. . .
. . .

ENDWHILE
```

Comments : When a WHILE statement is encountered, the expression will be evaluated and if the result is true, the statements following the expression will be executed until the ENDWHILE statement. Thereafter, execution branches back to the WHILE statement and the expression is evaluated again. The loop statements will be executed repeatedly until the expression becomes false.

It is possible to break out of a WHILE loop using the GOTO command or the EXIT command (only on PLCs with r75 or higher firmware).

Warning : Be careful that the WHILE loop will not be an endless loop as the PLC will appear to freeze up, being trapped in an endless-loop execution. TRiLOGI simulator attempts to detect this situation by giving a warning message if a loop is executed for an unduly large number of loops. Adding a break condition using GOTO or EXIT makes for more reliable WHILE loops.

Examples :

```
WHILE S = 1
        IF INPUT[1] & &H0002: S = 0 : ENDIF
ENDWHILE
```

Execution will only be terminated when input #2 is ON. WHILE loops may be nested; i.e. a WHILE loop may be placed within the context of another WHILE loop. Each Loop must have a separate ENDWHILE statement to mark the end of the loop.

12.83 WRITEMODBUS ch, DeviceID, address, data

Purpose : Automatically write the 16-bit data to a MODBUS device using the MODBUS RTU/ASCII protocol. The communication baud rate is the default baud rate of that COMM port unless it has been changed by the SETBAUD command.

Comments : The command automatically checks the response string received from the slave device for the correct LRC and the slave address. The status of the operation can be checked by the user program by testing the STATUS(2) function, which will return a ‘1’ for successful command and a ‘0’ if there is any error or if the slave device is not present.

Parameters :

ch = PLC COMM port number.
Range = 1-8 for MODBUS ASCII

Range = 11-18 for MODBUS RTU (11 is com1, 12 is com2, etc)

DeviceID = Device ID of the MODBUS device.
Range = 1 to 255

Address = Zero-offset address of the holding register in the MODBUS device.

data = the 16-bit data to be written to the MODBUS device

Example : WRITEMODBUS 3, 8, 1000, 1234

The data 1234 will be written to the MODBUS ASCII device with ID=08 at the holding register offset address 1000 (in MODBUS convention this refer to holding register #41001).

WRITEMODBUS 13, 8, 1000, 5678

The data 5678 will be written to the MODBUS RTU device with ID=08 at the holding register offset address.

See Also : READMODBUS( ), STATUS(2), NETCMD$( )

12.84 WRITEMB2 ch, ID, addr, var, count

Note : Applicable to all Nano-10, FMD Series, and F-series, but only to the legacy T100MD/MX PLC with firmware r44 or higher

Purpose : Automatically write the multiple words of 16-bit data to a MODBUS device using the MODBUS RTU/ASCII protocol. It is essentially the WRITEMODBUS command except multiple words can be sent.

Comments : The command automatically checks the response string received from the slave device for the correct LRC and the slave address. The status of the operation can be checked by the user program by testing the STATUS(2) function, which will return a '1' for successful command and a '0' if there is any error or if the slave device is not present.

Parameters :

ch - PLC COMM port number.
Range = 1-8 for MODBUS ASCII
Range = 11-18 for MODBUS RTU (11 is com1, 12 is com2, etc)

ID - Device ID of the MODBUS slave device.
Range = 1 to 255

addr - Zero-offset address of the holding register in the MODBUS slave device starting from 0 = 40001.

var - the starting variable in the master whose data is to be sent out (may be a DM or any system variable)

count - number of variables to send (max =16 in the legacy T100M Series PLCs).

Example : WRITEMB2 13,5,101,DM[10],8

The PLC will use MODBUS RTU protocol, via its Comm port #3, to write 8 words of data from DM[11] to DM[17] to the slave MODBUS device with ID = 05 and into its register offset address 101 to 108 (in MODBUS term. these refer to the #40102 to #40109 holding registers) .

See Also : READMB2, WRITEMODBUS, STATUS(2)

Appendix 1: PLC to PC Communications Setup - Hardware

A-1 Quick Start: PLC to PC Communication Guide

Part 1 of this guide will show how to quickly get connected to the PLC from a PC via serial port (RS232 and RS485).

Part 2 of this guide will show how to quickly get connected to the PLC from a PC via Ethernet port.

For a more in depth explanation see chapter 3 INTERFACING TO THE PLC

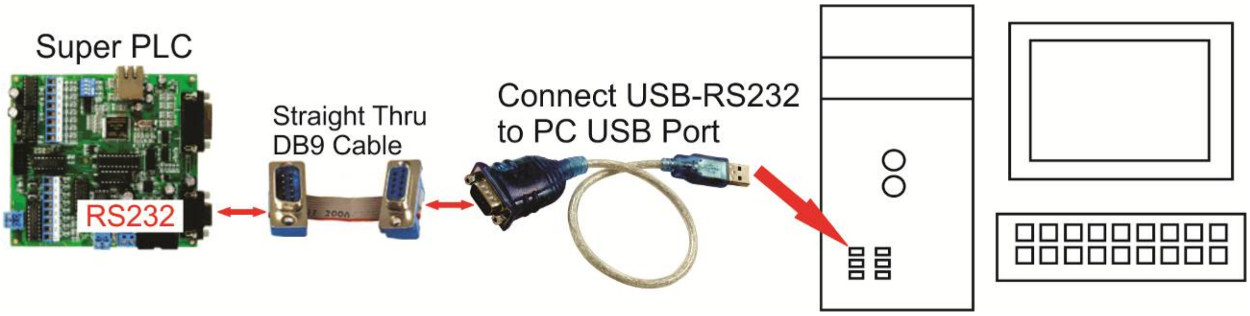
A1.1 Part 1: Serial Communication Setup Guide

1) CONNECT THE PLC TO THE PC VIA SERIAL PORT

Choose one of the following two options to connect the PLC to your PC. Note that if your PC requires a USB adapter, it should already be installed at this point.

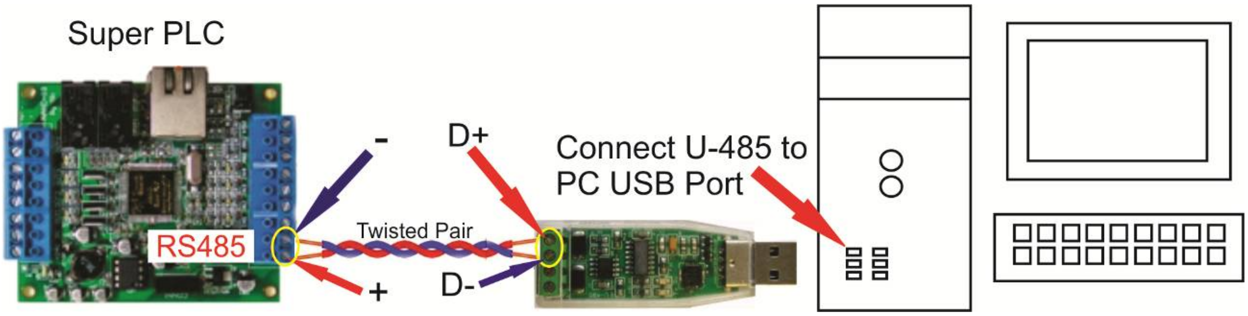
a) Connect via RS232

If you are connecting via RS232 with a USB-RS232 adapter, follow this diagram:



b) Connect via RS485

If you are connecting via RS485 with a USB-RS485 adapter, follow this diagram:



Note:

The D+ terminal of the U-485 is connected to the + terminal of the PLC RS485 port.
The D- terminal of the U-485 is connected to the – terminal of the PLC RS485 port.

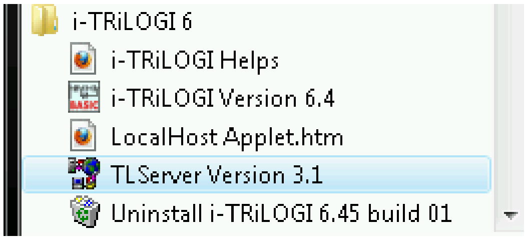
2) POWER THE PLC

Now that the communication wiring is setup, you can connect power to your PLC. Refer to your PLCs installation guide (chapter 1 of the respective user manual) for instructions on powering the PLC.

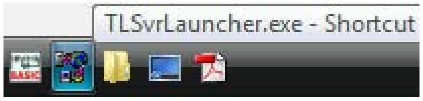
3) RUN TLSERVER

Start TLServer by selecting "TLServer Version 3.1" from "i-TRiLOGI 6" in your PCs start menu (below, top-left screenshot) or by clicking on the quick-launch symbol for TLServer (below, bottom-left screenshot). Once opened, you will see the PLC Webserver window (below, right screenshot):

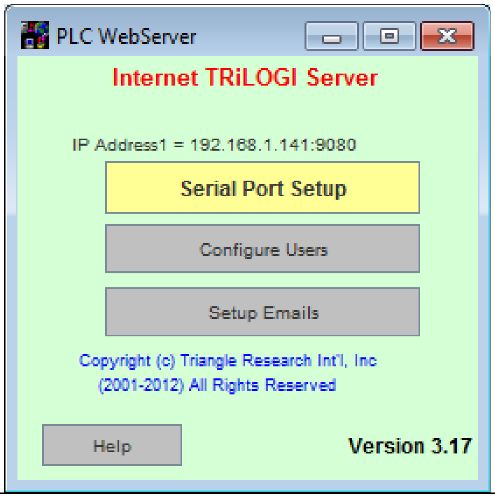
Open TLServer from Start Menu



Open TLServer from Quicklaunch



TLServer Opened Successfully



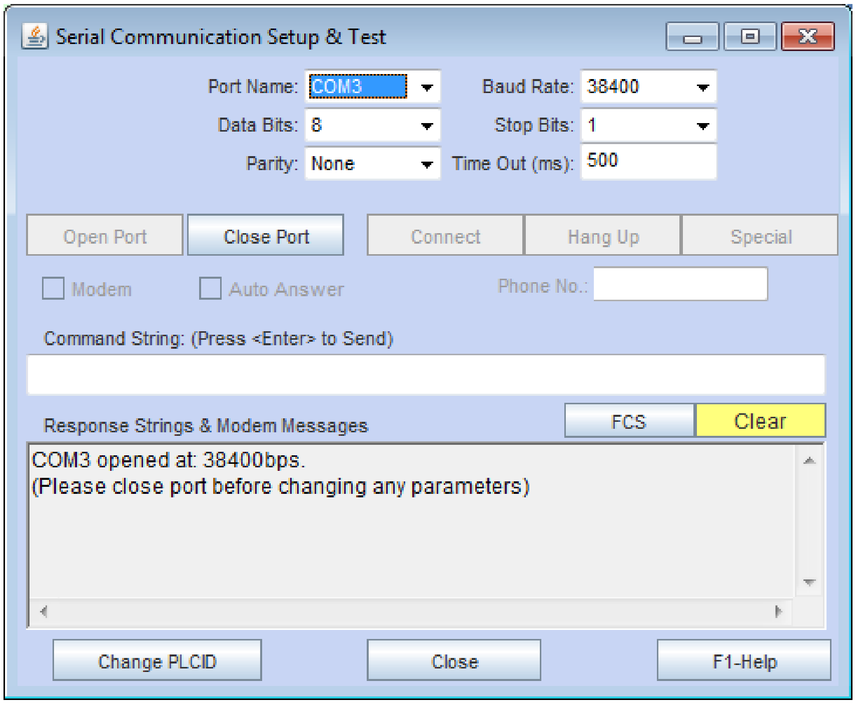
IP Address1 = 192.168.1.141:9080

Copyright (c) Triangle Research Int'l, Inc
(2001-2012) All Rights Reserved

Version 3.17

4) TLSERVER COMMUNICATION SETUP

Click on **Serial Port Setup** to open the serial port setup window (below screenshot). Once TLServer is configured, leave it running because it is needed to communicate with the PLC.

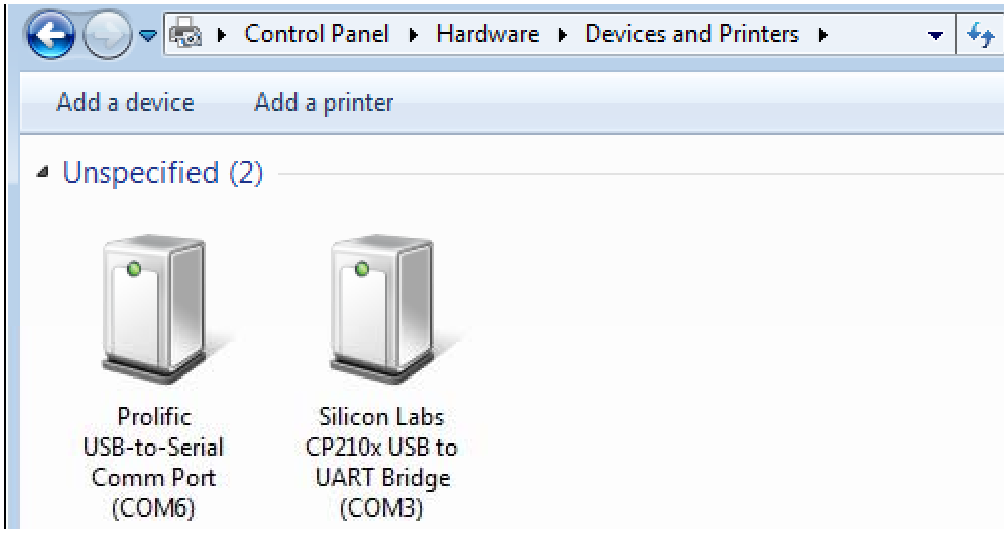

Serial Communication Setup & Test
Open Port
Close Port
Connect
Hang Up
Special
☐ Modem
☐ Auto Answer
Phone No.:
Command String: (Press <Enter> to Send)
Response Strings & Modem Messages
FCS
Clear
Change PLCID
Close
F1-Help

Port Name: COM3
Baud Rate: 38400
Data Bits: 8
Stop Bits: 1
Parity: None
Time Out (ms): 500

```
COM3 opened at: 38400bps.
(Please close port before changing any parameters)
```

If no settings have been changed in the PLC, then the default baud settings shown above can be used. You will only need to select the correct COM port from the Port Name: field.

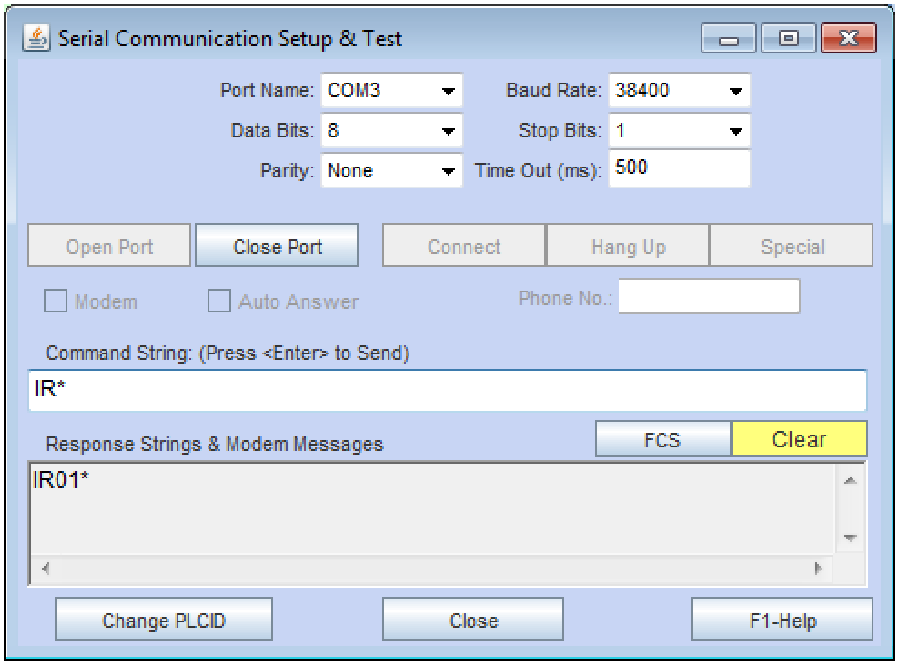
Select the port your USB adapter is installed on. If you are unsure, check the device manager in your PCs control panel. You can see from the below screenshot that the Prolific USB adapter (for RS232) is installed on COM6, so you would select COM6 from the Port Name field. Similarly, would select COM6 from the Port Name field for the Silicon Labs USB adapter (RS485).


Control Panel ▸ Hardware ▸ Devices and Printers ▸
Add a device
Add a printer
Unspecified (2)
Prolific USB-to-Serial Comm Port (COM6)
Silicon Labs CP210x USB to UART Bridge (COM3)

5) TEST COMMUNICATION FROM TLSERVER

In the same Serial Port Setup screen, type the following command in the Command String field to read the PLCs ID (default ID is 01):

Command String : **IR***
Response String : **IR01***



Port Name: COM3
Baud Rate: 38400
Data Bits: 8
Stop Bits: 1
Parity: None
Time Out (ms): 500

Command String: (Press <Enter> to Send)
IR*

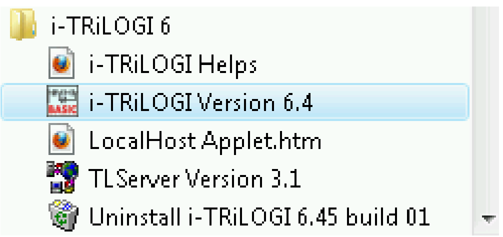
Response Strings & Modem Messages
IR01*

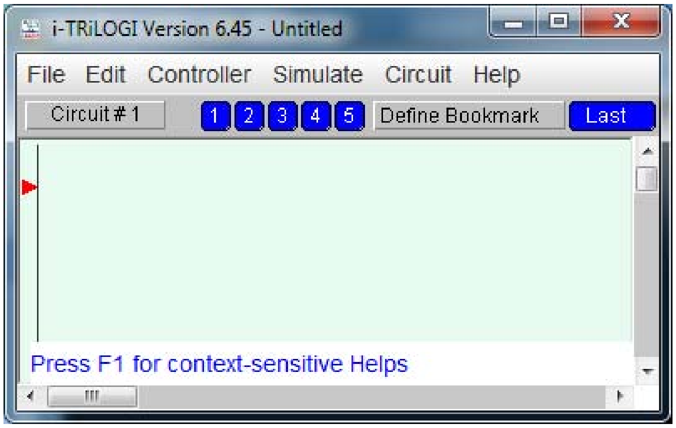
If you get the message **(Warning: No Response From PLC!)**, then communication is not setup properly. First repeat the steps from PLC connection to TLServer setup and if there is still no response, refer to the Communication Troubleshooting Appendix at the end of this document.

6) START TRILOGI

Start TRiLOGI by selecting "i-TRiLOGI Version 6.4" from "i-TRiLOGI 6" in your PCs start menu (below, left screenshot). Once opened, you will see the TRiLOGI programming environment window (below, right screenshot):

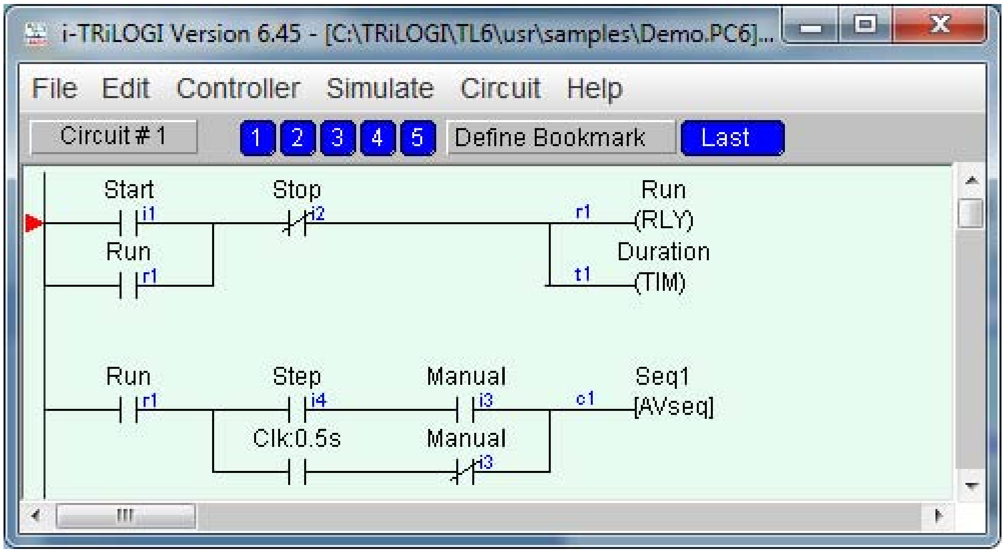
Start Menu





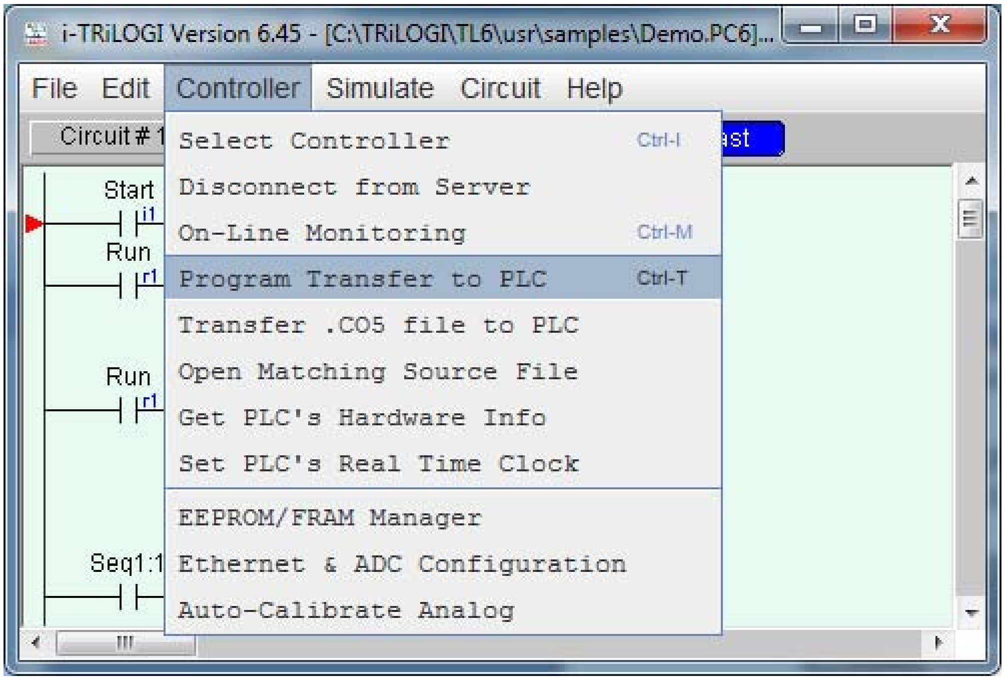
7) OPEN FILE

For the purposes of this tutorial, the Demo.PC6 program will be referenced. This is a sample program that is included in the i-TRiLOGI installation. This sample program, as well as all the other included sample programs, is accessible from the "**C:\TRiLOGI\TL6\usr\samples**" folder. Below is the Demo.PC6 program:

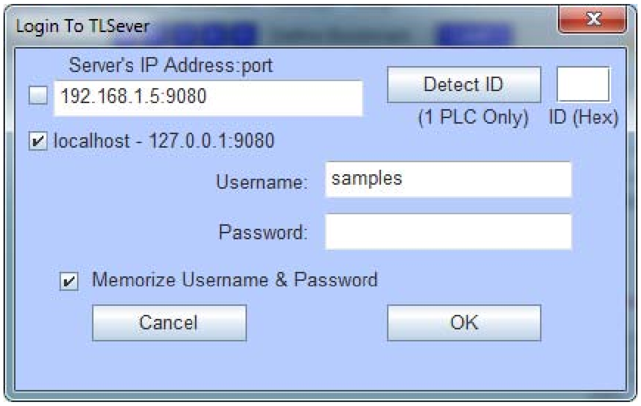


8) PROGRAM TRANSFER

a) **Select "Program Transfer to PLC"** from the "Controller" menu (or press CTRL +T on the keyboard).


i-TRiLOGI Version 6.45 - [C:\TRiLOGI\TL6\usr\samples\Demo.PC6]...
File
Edit
Controller
Simulate
Circuit
Help
Circuit # 1
Start
i1
Run
r1
Run
r1
Seq1:1
Select Controller Ctrl-I
Disconnect from Server
On-Line Monitoring Ctrl-M
Program Transfer to PLC Ctrl-T
Transfer .CO5 file to PLC
Open Matching Source File
Get PLC's Hardware Info
Set PLC's Real Time Clock
EEPROM/FRAM Manager
Ethernet & ADC Configuration
Auto-Calibrate Analog

b) The "**Login to TLServer**" window will open as shown below.



Server's IP Address:port

☐ 192.168.1.5:9080

☑ localhost - 127.0.0.1:9080

Username: samples

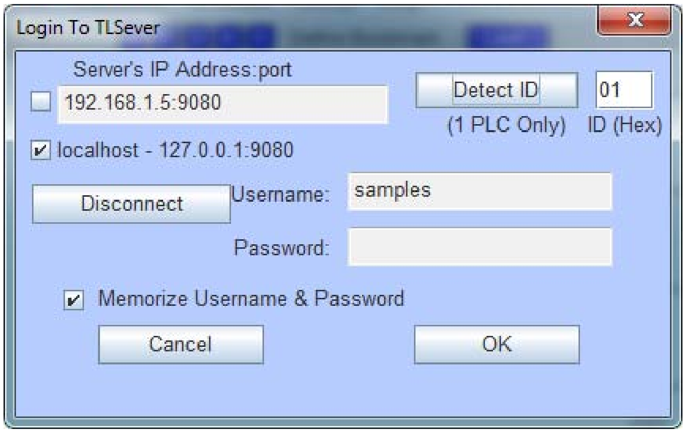
Password:

☑ Memorize Username & Password

<u>Default Settings:</u>

Server IP : localhost – 127.0.0.1:9080
Username : "samples"
Password : none (blank field)

c) **Click "Detect ID"** and the PLC ID should be returned in the adjacent field, as shown below, if communication is functioning correctly.



Login To TLSever

Server's IP Address:port

☐ 192.168.1.5:9080

☑ localhost - 127.0.0.1:9080

Detect ID: 01 (1 PLC Only) ID (Hex)

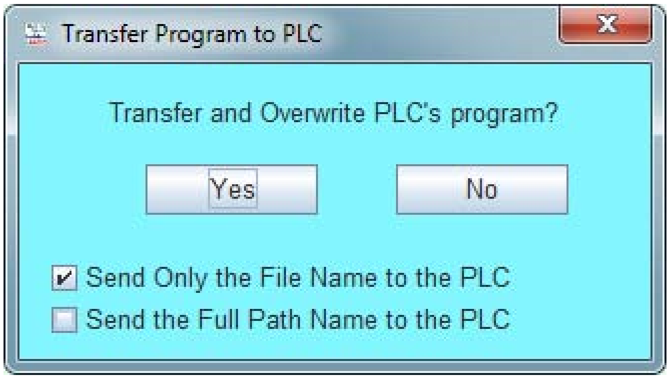
Username: samples

Password:

☑ Memorize Username & Password

d) **Click OK** to complete the server login process.

e) **Click 'Yes'** to continue when prompted with the program transfer confirmation window:



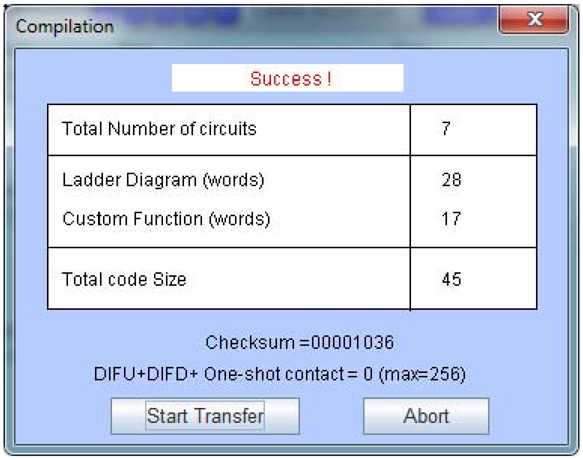
Transfer Program to PLC

Transfer and Overwrite PLC's program?

☑ Send Only the File Name to the PLC

☐ Send the Full Path Name to the PLC

f) **Click 'Start Transfer'** to initiate the actual program transfer if Success! is displayed, which indicates program compilation was successful.



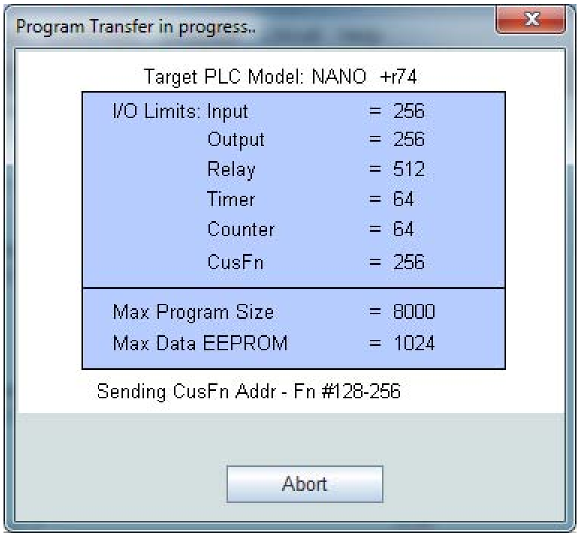
Success !

Total Number of circuits	7
Ladder Diagram (words)	28
Custom Function (words)	17
Total code Size	45

Checksum =00001036

DIFU+DIFD+ One-shot contact = 0 (max=256)

g) **Wait** while the program transfer takes place (below-left screenshot).

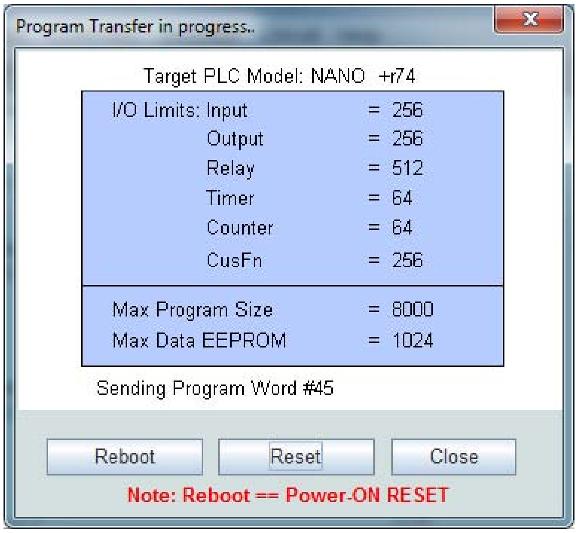


Target PLC Model: NANO +r74

I/O Limits: Input = 256
Output = 256
Relay = 512
Timer = 64
Counter = 64
CusFn = 256

Max Program Size = 8000
Max Data EEPROM = 1024

Sending CusFn Addr - Fn #128-256



Target PLC Model: NANO +r74

I/O Limits: Input = 256
Output = 256
Relay = 512
Timer = 64
Counter = 64
CusFn = 256

Max Program Size = 8000
Max Data EEPROM = 1024

Sending Program Word #45

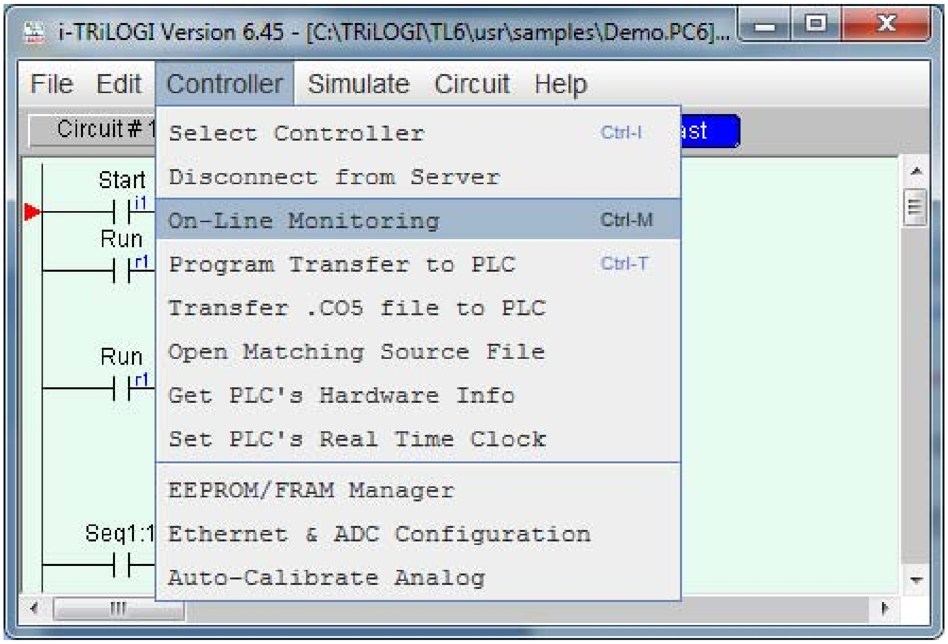
Note: Reboot == Power-ON RESET

h) **Click 'Reset'** when the program transfer is complete (above-right screenshot).

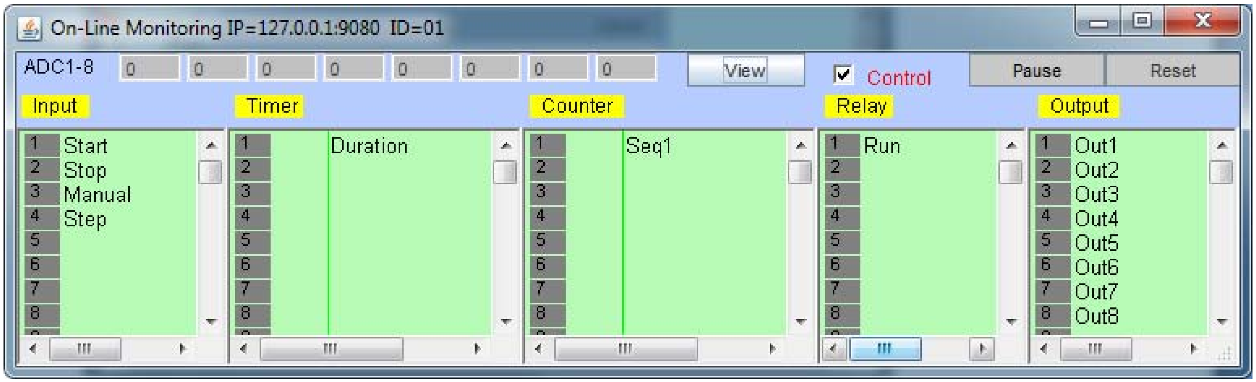
The program transfer is now complete and online monitoring can be performed.

9) ONLINE MONITORING

Select "On-Line Monitoring" from the "Controller" menu (or press CTRL + M on the keyboard).


i-TRiLOGI Version 6.45 - [C:\TRiLOGI\TL6\usr\samples\Demo.PC6]...
File
Edit
Controller
Simulate
Circuit
Help
Circuit # 1
Select Controller Ctrl-I
Disconnect from Server
On-Line Monitoring Ctrl-M
Program Transfer to PLC Ctrl-T
Transfer .CO5 file to PLC
Open Matching Source File
Get PLC's Hardware Info
Set PLC's Real Time Clock
EEPROM/FRAM Manager
Ethernet & ADC Configuration
Auto-Calibrate Analog
Start
i1
Run
r1
Run
r1
Seq1:1

The online monitoring window should open immediately if you are still connected to the server.

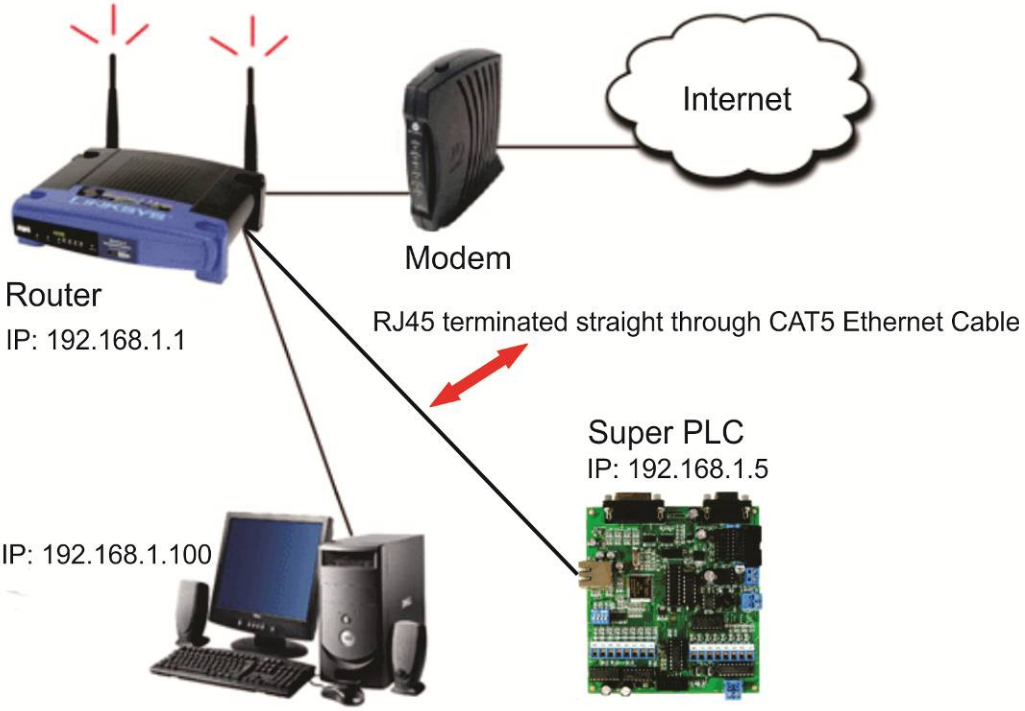


If not, then **follow steps b-d** from the serial Program Transfer section above to login to TLServer, or the Ethernet Program Transfer section below to login to FServer (PLC). The above monitoring window will appear after successful login.

A1.2 Part 2: Ethernet Communication Setup Guide

1) NETWORK WIRING

Below is an example of the most common network setup, which assumes the default PLC IP address (192.168.1.5) is able to work on your network. If this IP will not work, then follow the serial port setup instructions in part 1 of this guide and then refer to chapter 2 of your PLCs user manual for details on configuring the network settings via serial port connection.



2) START TRILOGI

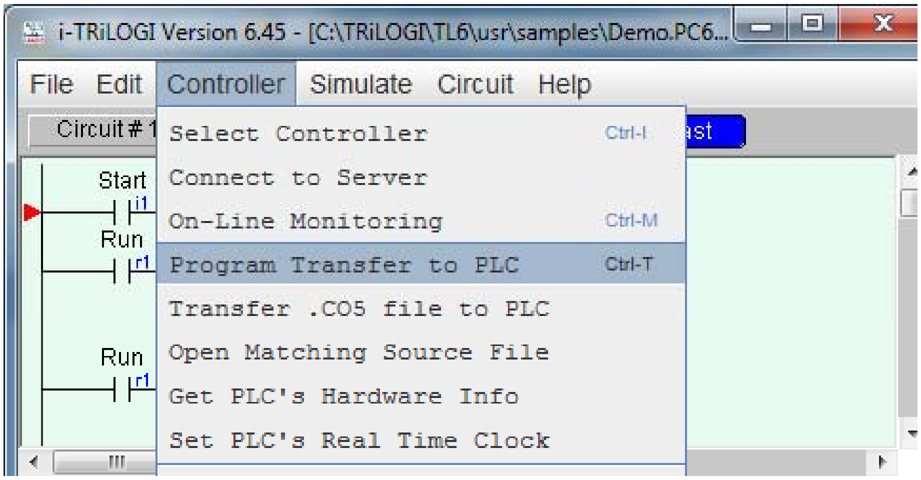
Refer to the START TRILOGI section in Part 1 of this guide. TLServer is not needed for Ethernet communication, so there is no need to start it or configure it.

3) OPEN FILE

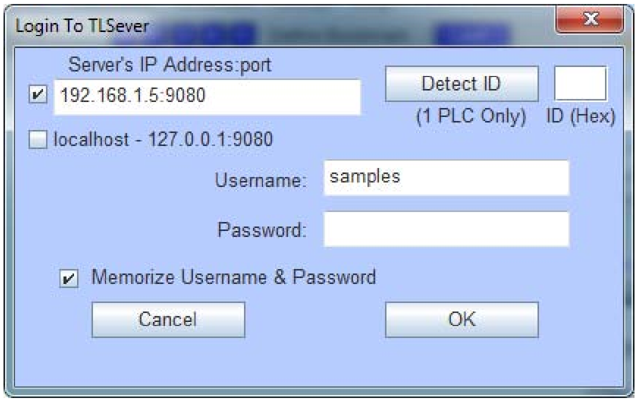
Refer to the OPEN FILE section in Part 1 of this guide.

4) PROGRAM TRANSFER

a) **Select “Program Transfer to PLC”** from the “Controller” menu (or press CTRL +T on the keyboard).


i-TRiLOGI Version 6.45 - [C:\TRiLOGI\TL6\usr\samples\Demo.PC6...
File
Edit
Controller
Simulate
Circuit
Help
Circuit # 1
Start
i1
Run
r1
Run
r1
Select Controller Ctrl-I
Connect to Server
On-Line Monitoring Ctrl-M
Program Transfer to PLC Ctrl-T
Transfer .CO5 file to PLC
Open Matching Source File
Get PLC's Hardware Info
Set PLC's Real Time Clock

b) The "**Login to TLServer**" window will open as shown below. Note that in this case TRiLOGI will login to the PLCs FServer, even though it displays Login to TLServer.



Server's IP Address:port

☑ 192.168.1.5:9080

☐ localhost - 127.0.0.1:9080

Username: samples

Password:

☑ Memorize Username & Password

Default Settings:

PLC IP : 192.168.1.5:9080
Username : none (ignored)
Password : none (ignored)

c) **Enter the IP address and port number** of your PLC if it was changed from the default: 192.168.1.5:9080

d) **Follow steps c-h** in the Program Transfer section from Part 1 of this guide.

5) ONLINE MONITORING

Follow the steps in the ONLINE MONITORING section from Part 1 of this guide.

Appendix 2: Application Notes & Programming Examples

A-2 Appendix 2 : Application Notes & Programming Examples

A2.1 Important Notes to Programmers of TRiLOGI Version 6.x

A2.1.1 Understanding Ladder Logic Execution Process

Like all industrial PLCs, the CPU of the 'Super' PLC first checks the logic states of the physical inputs and copies them into memory. During the ladder logic scan the actual logic states of the physical Inputs (except for interrupt inputs) are ignored by the PLC. The CPU uses the memory copy of the inputs to execute the ladder program.

The CPU executes its ladder logic program starting from the top rung of the program to the bottom rung. When the CPU reaches a ladder rung that activates a {CusFn} or {dCusF} that custom function will be executed. The CPU will only continue to scan the rest of the ladder program when the current custom function ends normally. Hence the order in which a ladder rung is placed within a ladder program can have an effect on the behavior of the program.

Output bits which are changed as a result of the program execution will only be updated to the physical outputs at the end of the ladder logic scan. One scan time is defined as the time it takes to execute the 3 steps (read physical inputs, execute program, update physical outputs). The CPU repeats these 3 steps continuously all the time, known as "Ladder Logic Scanning".

Hence, it is important to note that the variables INPUT[n] s and OUTPUT[n] in TBASIC are not the actual physical I/Os of the PLC, but only a memory representation of the actual I/Os which will be updated only during the I/O update cycles. The logic states of physical inputs are copied into the INPUT[n] variables during input scan and the physical outputs are set to the logic states contained in the OUTPUT[n] variables during output updates.

Therefore, one potential error that traditional BASIC programmers tend to commit is to attempt to poll for a change in the variable INPUT[n] within TBASIC such as the following:

✕

```
WHILE  INPUT[1] = 0
    ..
ENDWHILE
```

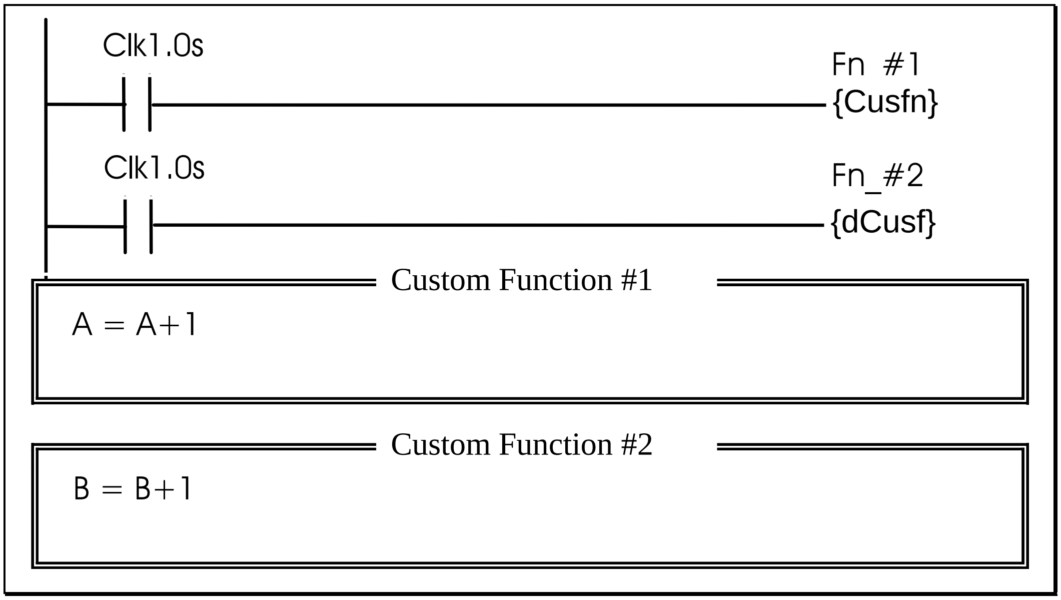
This will result in an endless loop since the value of the variable INPUT[1] will never change during execution of the custom function regardless of the actual logic states of physical input #1 to #8. The only way to force upon a physical I/O update is to use the **REFRESH** command, but it is not a good practice for ladder logic programming to update physical I/Os in the midst of a program execution. The REFRESH command is meant more for forcing an immediate output to be turned ON or OFF during time-critical situations.

Hence it is important to allow a ladder logic program to finish its scan so that the physical I/Os can be updated. You should never hog the CPU within a particular custom function as this will mean the rest of the ladder program don’t have a chance to be executed in a timely manner.

A2.1.2 The Difference Between {CusFn} and {dCusF}

It is very important to understand the difference between the two formats of the custom functions once you understand how the ladder logic scanning process works as described in the last section. If you use the **{CusFn},** the custom function will be executed **EVERY SCAN** of the ladder logic program as long as its execution condition is ON.

On the other hand, the **{dCusF}** (known as the differentiated or one-shot format) is executed only **ONCE** when its execution condition goes from OFF to ON. The execution condition must go OFF and then ON again for the function to be executed again. It is not difficult to see that the differentiated format is used far more frequently than the other one since most custom functions involve arithmetic and when a condition is ON you most likely want the computation to be performed ONCE and not repeatedly in every scan of the ladder logic. You can easily understand the difference between the two formats if you run the following sample program:



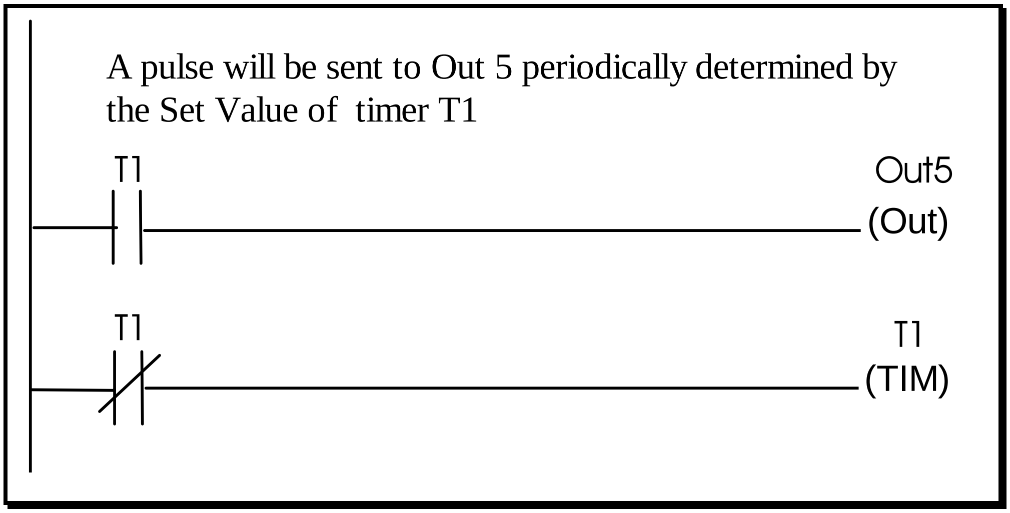
Run the program in simulator and press the <V> key to view the changes in the variables A and B. You will see that B is incremented by one every second, while A is incremented wildly for 0.5s and then stops for 0.5s. Try it! It can be very educational!

If you want to periodically check the status of an analog input or the real time clock, you should use a clock pulse (0.1s, 1.0s etc as shown in the example) and connect to a **{dCusF}**. Connecting to non-differentiated version would mean checking thousands of times for half the period and not at all for the other half period -- certainly not the intended outcome.

A2.1.3Timers Contact Updating Process

All the timers' contacts of the PLC, like the inputs and outputs, are updated simultaneously at the beginning of every ladder logic scan and not at the rung that contains the (TIM) coil. So if you are using self-reset timer, please note that if a timer times out its contact will be ON from the beginning of the ladder logic rung until the rung that contains the self-reset circuit. Thereafter the timer contact will be OPEN since the coil has been self-reset.

Hence please note that you should place the self-reset timer rung after all the ladder rungs that utilize the said timer contact. This allows those ladder rungs which use the timer contact to have a chance of being executed before the self-resetting rung clears the timer.

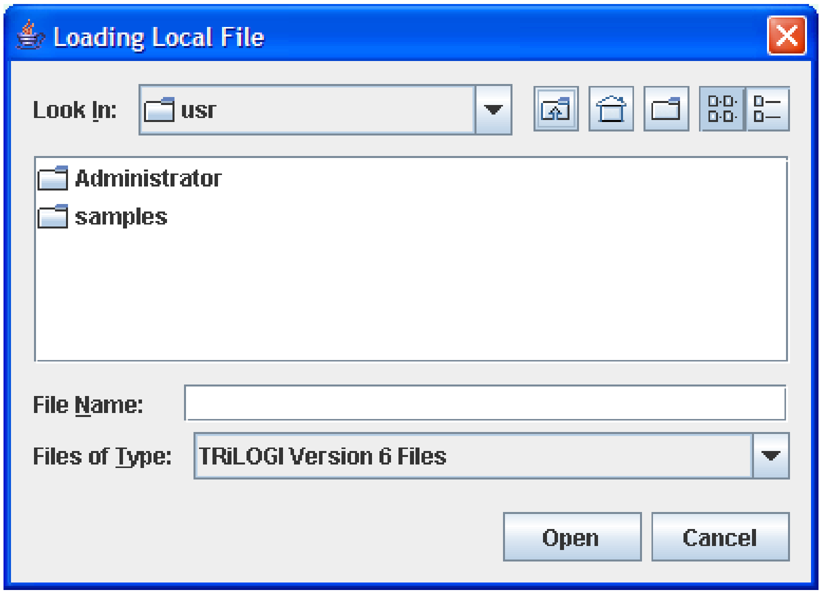

A pulse will be sent to Out 5 periodically determined by the Set Value of timer T1

A2.2 TRiLOGI Sample programs

There are many well-documented demos, as well as practical TRiLOGI program examples, included in the following TRiLOGI installation folder:

<TRiLOGI installation folder>\usr\samples

When you click on TRiLOGI's "File -> Open (Local Drive)" command, you will be able select the user's folders where program files are stored. By default, only two users are defined: "Administrator" and "samples" as follow:


Loading Local File

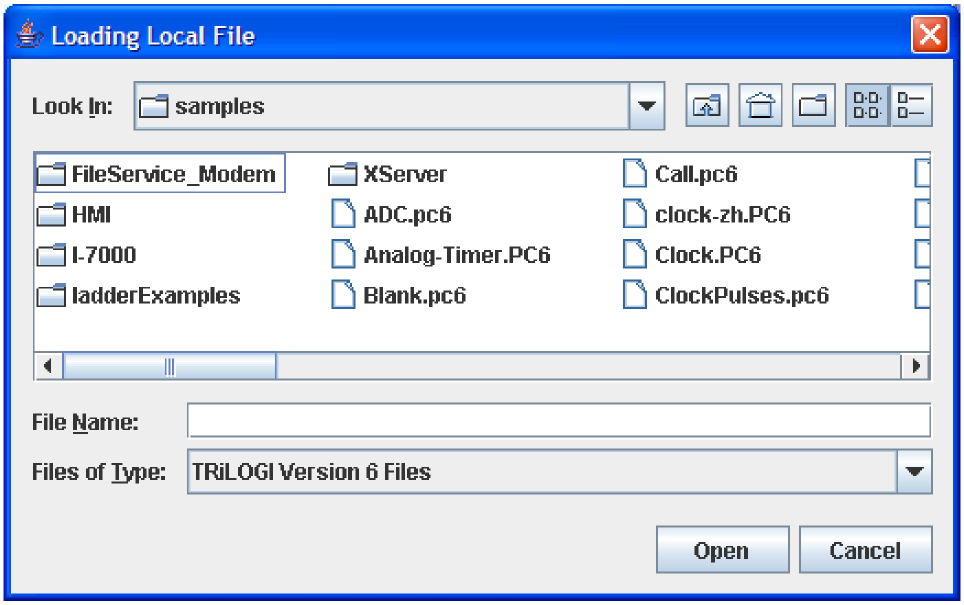
Look In: usr

Administrator
samples

File Name:

Files of Type: TRiLOGI Version 6 Files

You should open the “samples” folder and select any files with “.PC6” extension for viewing.



Loading Local File

Look In: samples

File Name:

Files of Type: TRiLOGI Version 6 Files

There are also sub-folders within the “samples” folder where sample programs that relate to a particular topic or device are stored, such as those relates to using the MD-HMI. We strongly encourage you to open these example programs to see how these programs are structured. Most of these programs can be run in the simulator except those that involve communication with other devices or high speed counters.

A2.2.1Display Alphanumeric Messages on built-in LCD Display

FMD series and F-series PLCs all come with a built-in LCD display port that allows low cost connection to industry standard LCD display module. For such PLCs, programming of the LCD display is via the SETLCD statement supported by TBASIC language.

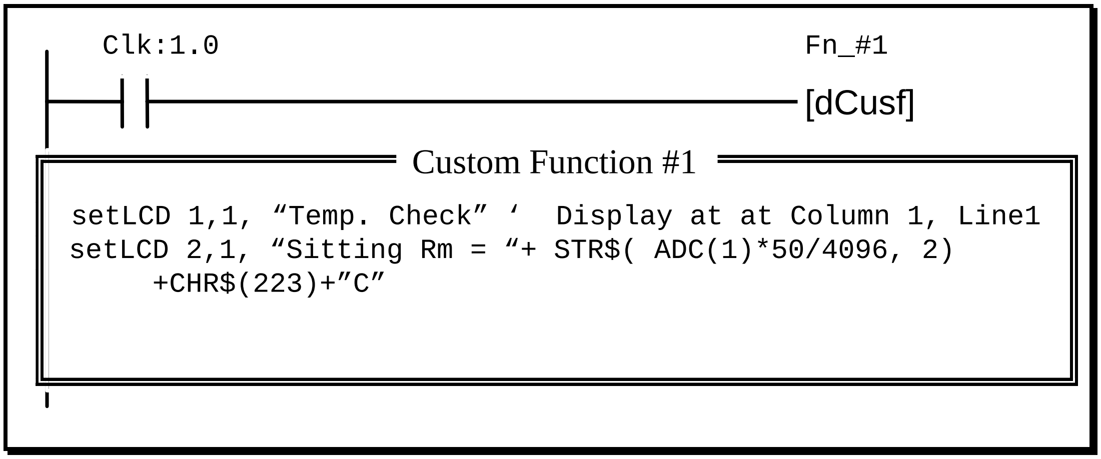
Assignment:

Every 1 second, display a message as follow:

Temp. Check
Sitting Rm = xx ^{o}C.

Where xx depends on reading of A/D #1 which is returned by function ADC(1).

Full scale A/D is 4096.
A/D range (0 to 4096) ⟹ Temperature 0 to 50°C



```
setLCD 1,1, “Temp. Check” ‘  Display at at Column 1, Line1
setLCD 2,1, “Sitting Rm = “+ STR$( ADC(1)*50/4096, 2)
     +CHR$(223)+”C”
```

Comments:

Every one second, the special bit Clk:1.0s closes and activates Function #1. Within the Custom Function #1, ADC(1) reads the A/D converter #1 and converts it into degrees. The integer value is then converted into a two-digit string using the STR$ function and concatenated to the rest of the text string for display using the SETLCD command.

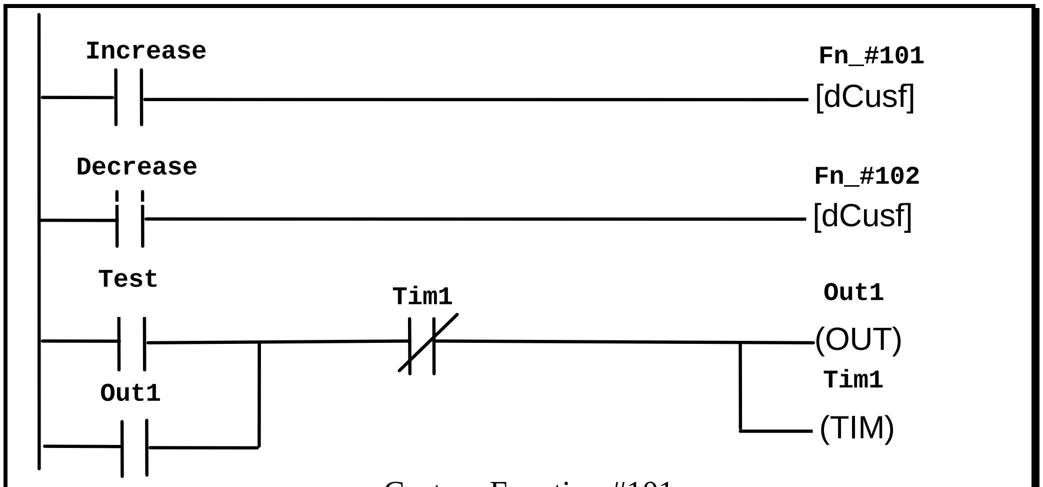
Simulation of the display string to built-in LCD is supported on TRiLOGI Version 6.x. When in Simulation mode, press <V> key to view the Special Variables and the messages will appear in an LCD Simulation window.

A2.2.2 Setting Timer/Counter Set Values (S.V.) Using LCD Display

If you have an LCD display, then you can use two push-buttons inputs to change the Set Values (SV) of any selected timers or counters with visual feedback.

Assignment:

1. Press push-button “Increase” increment the SV of timer #1 by 0.5s. The upper limit for timer #1 SV is 10s (SV <=100)
2. Press push-button “Decrease” decrement the SV of timer #1 by 0.5s
3. Press “test” button turns ON output #1 for a duration given by timer #1 and then turns it OFF.



Custom Function #101

```
Z = getTimerSV(1)
IF Z > 100 RETURN: ENDIF      ‘ MAXIMUM 10s
setTimerSV  1, Z+5  ‘Increase the current SV by 5 (0.5s)
SETLCD 1,1,”T1-SV=”+STR$(getTimerSV(1),4)
```

Custom Function #102

```
setTimerSV 1,getTimerSV(1)-5    ‘Decrease SV by 5
SETLCD 1,1,”T1-SV=”+STR$(getTimerSV(1),4)
```

Comments:

The **getTimerSV(1)** function returns the current set value of the Timer #1. This value is read into variable Z in CusFn #101 but used directly in CusFn #102 for changing the Set Value of Timer #1. The **setTimerSV** statement uses the value of its second argument to update Timer #1’s SV accordingly.

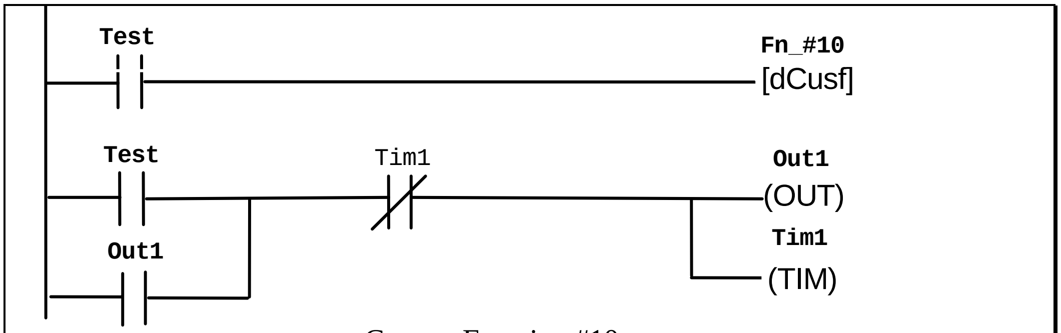
Note that changes to the set value SV will be updated in the program EEPROM memory and is non-volatile. However, EEPROM has a typical life-span of about 100,000 to 1,000,000 erase-write cycle. Exceeding this limit will “wear out” the EEPROM and resulting in a read error when the PLC operates. Hence, you should **NEVER** write a program that excessively changes the set value of the timer or counter (e.g. put it in a non-differentiated form of [CusFn] which executes every scan of the ladder program and continuously changes the content of the EEPROM).

A2.2.3Using a Potentiometer As An Analog Timer

A cheap potentiometer can be connected to the PLC A/D input and provide a user-adjustable “knob” as an analog “Set Point” input device. A scale can be drawn around the potentiometer to provide visual indication of set point value.

Assignment:

1. A potentiometer is connected to A/D #5. Use it to provide a timing range of 0 to 10.00 seconds.
2. Pressing the “test” input turns ON output #1 for a duration determined by the potentiometer reading, after that turns output #1 OFF.



Custom Function #10

```
HSTIMER 1    ‘ Define Timer #1 as High Speed Timer (0.01s base)
TimerPV[1] = ADC(1)*1000/4096  ‘ Set the timer running with
                               ‘ value proportional to A/D value.
```

Comments:

To take full advantage of the resolution of the A/D converter, the timing range of 0-10 seconds is more finely divided when timer is defined as high-speed timer using the HSTIMER command. The time base is now 0.01s. This means that for maximum value of 10.00s, the timer should count down from 1000.

The next statement in CusFn #10 computes the ratio of the A/D input with respect to its full scale value of 4096 and multiplies it to the maximum timing value of 1000. I.e., if the potentiometer wiper is at half way, the A/D reading will be around 2048, the computation will results in a timing value = 2048*1000/4096 = 500, or 5.00 second. Note that TRiLOGI 6.x does not support floating-point arithmetic. Hence <u>the multiplication must be carried out before the division</u>. Otherwise, if you compute 2048/4096 *1000, the result of the integer division of 2048/4096 = 0 and the whole expression yields a ‘0’, which is clearly wrong!

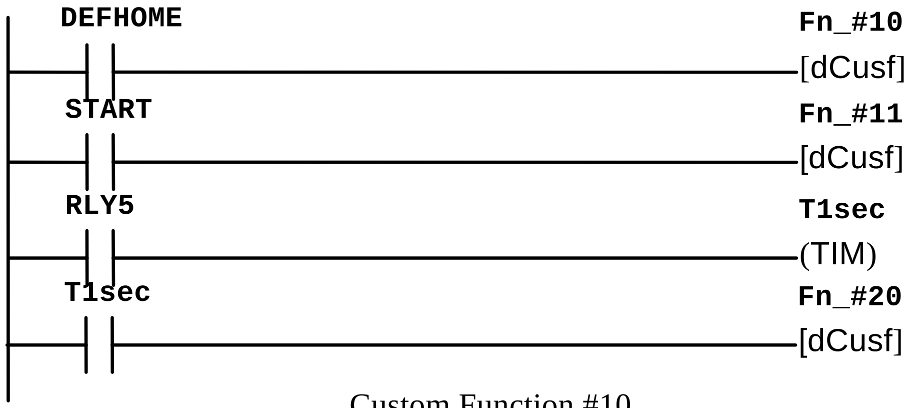
The timer #1's Present Value (P.V) register is loaded with this number, which will start the timer countdown. In the next logic rung, the timer coil connected to the latched "OUT1" is necessary to prevent the timer from resetting itself. But It will not overwrite the PV with its own Set Value (SV), which will not be used at all in this case. This is because the previous ladder program has already started the timer with a value determined by the position of the potentiometer "knob".

A2.2.4Motion Control of Stepper Motor

The 'Super' PLC can generate pulses to feed to stepper motor driver. The maximum speed, acceleration, deceleration and total number of pulses to generate are definable using TBASIC. Both absolute positioning commands and relative move commands are supported.

Assignment:

1. A "DEFHOME" input define the current location as home position.
2. Press the "START" input to begin Indexing the stepper motor to position at 1500, -2000, 4500 and 9000 steps with respect to the HOME position. Pause for 1 seconds at each position. Return to home at the end of the cycle.
3. Maximum speed = 5000 pps, Acceleration=100 steps to full speed.



Custom Function #10

```
STEPHOME(1)       ‘Define the HOME position for stepper 1
```

Custom Function #11

```
DM[1] = 1500: DM[2]= -2000: DM[3]=4500   ‘Store index position
DM[4]=9000: DM[5]=0
N = 1
STEPSPEED 1, 5000,100      ‘Stepper1: Max 5000pps, Acc:100
STEPMOVEABS 1, DM[N], 5    ‘ Move to position stored in DM[1]
                           ‘ at the end, turns ON relay 5
```

Custom Function #20

```
 N = N+1
IF N <= 5
  STEPMOVEABS  1, DM[N], 5  ‘Move to next position in DM[N]
ENDIF                      ‘ at the end, turns ON relay 5
```

Comments:

RLY5 is the label for internal relay #5. T1sec is a timer with preset value of 10. At the end of the pulse generation, RLY5 will be activated. Ladder logic senses RLY5 and executes the T1sec timer to cause a 1 second delay, after which custom function #20 is executed which triggers another STEPMOVEABS command and the process repeats for the other four indexing positions.

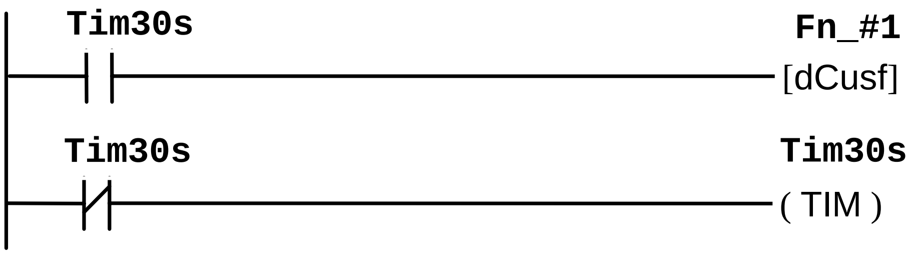
A2.2.5 Activate Events at Scheduled Date and Time

All ‘Super’ PLCs have built-in Real Time Clock which keeps track of Date and Time and can be used to activate events at scheduled time.

Assignment:

1. Every day turn on output #1 (label name: Out1) at 19:00.
2. Turn OFF output #1 at 7:00

3. On 1st Jan 2000 at 12:00 turn ON output #5.
4. On the same day at 18:00 turn OFF output #5



Custom Function #1

```
IF TIME[1]=19 AND TIME[2]=0  ‘ Hour hand at 19
    SETIO OUT1            ‘ Minute hand at 00
ELSE
    IF TIME[1]=7 AND TIME[2]=0
       CLRIO OUT1
    ENDIF
ENDIF

IF DATE[1]=2000 AND DATE[2]=1 ‘Jan, yr 2000
   IF DATE[3]=1
      IF TIME[1]=12 SETBIT OUTPUT[1],4:ENDIF
      IF TIME[1]=18 CLRBIT OUTPUT[1],4:ENDIF
   ENDIF
ENDIF
```

Comments:

1) Tim30s should have a Set Value = 300 and it activates Function #1 every 30 seconds. It is not necessary to check the clock too often as checking consume CPU execution cycles.
2) We used SETIO to control Output #1, but as a demonstration we use SETBIT to control Output #5 which is bit #4 of the variable OUTPUT[1]. The statement SETBIT output[1],4 turns ON output 5.
3) Actually it may not be necessary to check the minute hand since when the RTC turns from 18:59 to 19:00, the output will be turned ON as long as TIME[1]=19. Only when TIME[1]=7, then output #1 needs to be changed.

A2.2.6HVAC (Heating, Ventilation and Air-Conditioning) Control

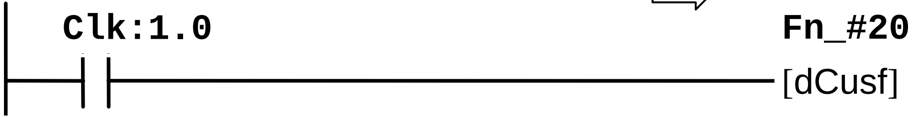
Assignment:

1. Read desired temperature setting (S) from a potentiometer connected to A/D #5.
2. Read current air temperature (T) from sensor attached to A/D #1 (T)
3. Turn ON cold air-conditioner (output #1 with labelname: OUT1)
   if T > S by more than 1.5 oC.
4. ON heater if (output #2)
   if S > T by more than 1.5 oC.
5. Turn OFF both heater and cold air-conditioner if T is within + 1.5 oC of S.

Parameters:

Full scale A/D is 4096

Range of Set Point:	A/D #5 = 0	⇨	+16.0 °C
	A/D #5 = 4096	⇨	+30.0 °C
Range of Sensor:	ADC#1 = 0	⇨	-10.0 °C
	ADC#1 = 4096	⇨	+50.0 °C



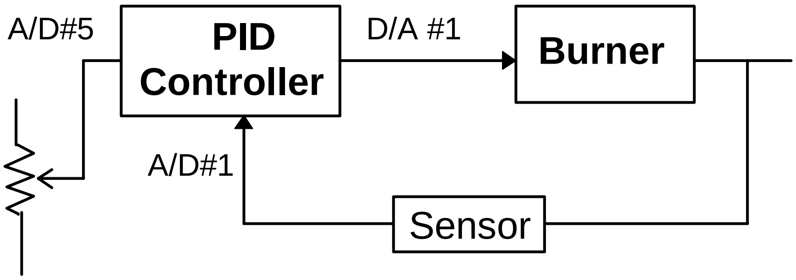
Custom Function #20

```
S = ADC(5)*(300-160)/4096 +160   ‘Convert to ºCx10
T = ADC(1)*(500+100)/4096 -100   ‘Convert to ºCx10

IF S -T > 15
   SETIO OUT1  ‘Cold Air-conditioning ON
ELSE
   CLRIO OUT1  ‘if T is hotter by 1.5 ºC
ENDIF

IF S-T < -15
   SETBIT OUTPUT[1],1   ‘ Heater ON
ELSE
   CLRBIT OUTPUT[1],1   ‘if T is colder by 1.5 ºC
ENDIF
```

Comments:

Since TRiLOGI Version 6.x does not support floating point computation, in order to handle decimal value (±1.5° C) in this application we use a unit integer to represent 0.1 quantity. All temperature readings are x10 times. Hence 16.0°C is represented by 160, -10.0°C is represented by -100. This method, known as fixed-point computation is quite commonly used in industrial control program.

A2.2.7 Closed-Loop PID Control of Heating Process



E.g. Implementing Closed-loop Digital Control with PID computation function

PID Controller Transfer Function:

$$G(s) = KP + \frac{K_I}{s} + KD\, s$$

K_P = Proportional Gain $= \dfrac{1}{\text{Proportional Band}}$

K_I = Integral Gain $= \dfrac{1}{\text{Integral Time Constant}}$

Assignment:

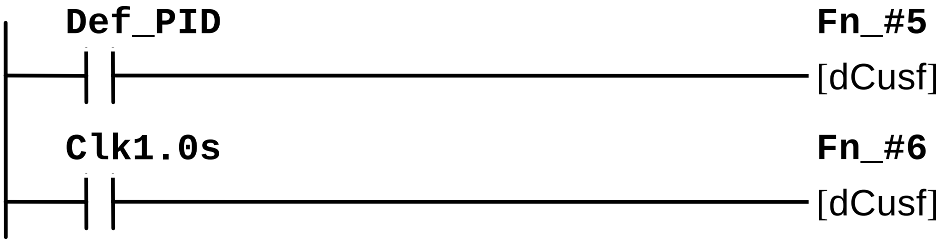
1. Read desired set-point temperature from a potentiometer connected to A/D #5 (S) with temperature range between 50 oC - 200 oC
2. Measure the process temperature from a thermocouple + signal conditioner attached to A/D #1(T)
3. Compute the Error = S - T. Apply Proportional + Integral + Derivative (P.I.D) algorithm to compute output X.
4. Apply output X to Digital-to-Analog converter D/A #1 to control a variable position valve that feed fuel to the flame.
5. Sample and compute every 1 second.

<u>Full scale A/D range is 4096</u>

Range of Set Point: A/D #5 = 0 ⇨ +50.0 °C

A/D #5 = 4096 ⇨ +200.0 °C

Range of Sensor: ADC#1 = 0 ⇨ 0 °C
ADC#1 = 4096 ⇨ +300.0 °C



Custom Function #5

```
P = 500: I = 50: D = 0
PIDDEF 1, 2048*100 ,P,I,D ‘ Use PID Engine #1, max limit
                          ‘  = +/- 50% of full scale
```

Custom Function #6

```
S = ADC(5) * (200-50)/4096 + 50    ‘Convert to ºC
T = ADC(1) * (300 - 0)/4096

X = PIDcompute(1, S - T)/100 + 2048  ‘X can vary + 50%
setDAC 1, X             ‘ Write to analog D/A output #1
```

Comments:

1) We use two decimal places to represent the gains K_P, K_I and K_D. Each integer unit represents 0.01. Proportional gain KP = 5 is represented by variable P = 500. Likewise, Integral gains KI = 0.5 is represented by I = 50 and Differential gains = 0 means Differential term is not used (P.I. only). The integrator limits of + 2048 for the PIDDEF statement must be multiplied by 100 to be put on the same scale as the P,I and D parameters.

   Note that since TRiLOGI does not support floating point arithmetic, the multiplication must be carried out before the division. Otherwise, if you compute 150/4096 *ADC(5), the result of the integer division of 150/4096 = 0 and the whole expression yields a ‘0’, which is clearly wrong!

2) The value returned by PIDcompute() function is then divided by 100 to get the real value of controller output. PIDcompute() returns a signed value

which can vary from -limit to + limit. We choose the 50% D/A output (4096/2 = 2048) as the mean control point so that negative values from PIDcompute() means D/A output will be < 2048, positive values means D/A output will be > 2048.

Appendix 3: A-3 Troubleshooting Serial Communication

A-3 Troubleshooting Serial Communication

If you are unable to communicate with the PLC from TRiLOGI, first check the following:

1. **TLServer is Running**. This is required for serial communication to work. Refer to chapter 1.5 Run TRiLOGI and TLServer for instructions on starting TLServer

2. **The PLC is powered on and properly connected to the PC**. Refer to A1.1 Part 1: Serial Communication Setup Guide for serial port connection info. Refer to your PLCs installation guide (chapter 1 of the respective user manual) for instructions on powering the PLC.

3. **The Correct COM port is selected in TLServer**. Refer to section 4) TLSERVER COMMUNICATION SETUP in the A1.1 Part 1: Serial Communication Setup Guide for instructions on selecting the correct com port.

4. **The USB to serial converter is properly installed**. If the adapter is not listed in your PCs device manager, as shown in the TLSERVER COMMUNICATION SETUP section referenced above, then the adapter driver may still need to be installed. Refer to the CD/installation instructions provided with the USB adapter. Once installed, make sure to restart TLServer if it was left open.

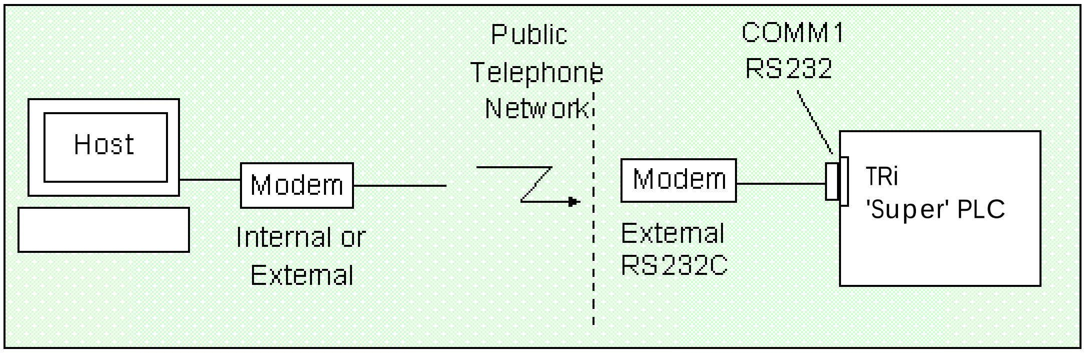
5. **The PLC program is not using the serial port**. If the PLC program is trying to send data out of the serial port, then it will corrupt communication attempts from TRiLOGI/TLServer and would not work. It is possible to pause the PLC using DIP Switch #4 on the blue switch box that is physically located on the PLC (Note that it is jumper J4 for the Nano-10 PLC). The PLC should be power cycled after being paused, which will bypass any program initialization function that could affect the default PLC communication settings. TLServer can be set to the default com settings: **38,400 bps, 8 data bits, 1 stop bit, no parity** and communication can be reattempted. Refer to chapter 1.8.1 of your PLCs user manual for more information on this.

If communication is still not possible at this point, please contact tech support by phone at 877 874 7527 or by email at support@tri-plc.com.

Appendix 4: A-4 PLC-to-Modem Communication Setup

A-4 PLC-to-Modem Communication Setup

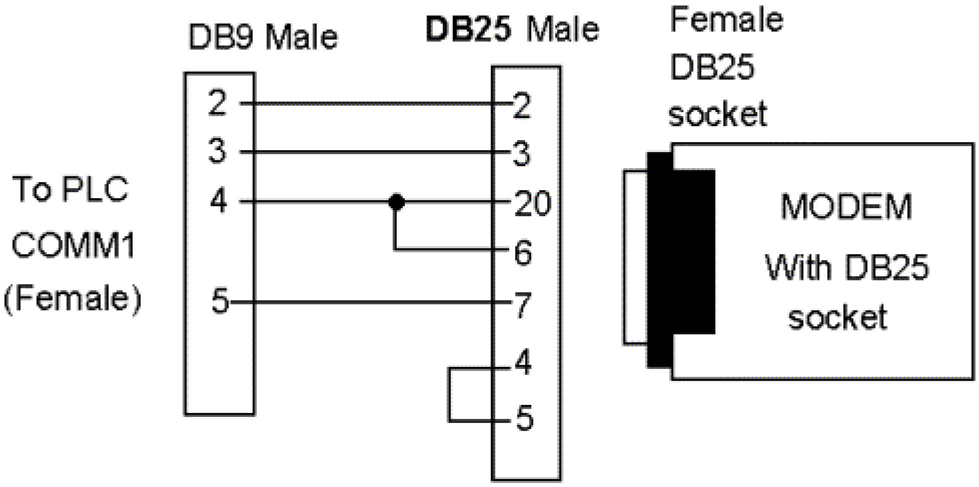
A remotely located Nano-10, FMD series, or F-series PLC can be connected to a host PC via public-switch telephone network (PSTN), radio or cellular phone network. This can be accomplished by using two analog modems, one connected to the PLC's RS232 serial port, and another modem connected to the remote host PC as follow:



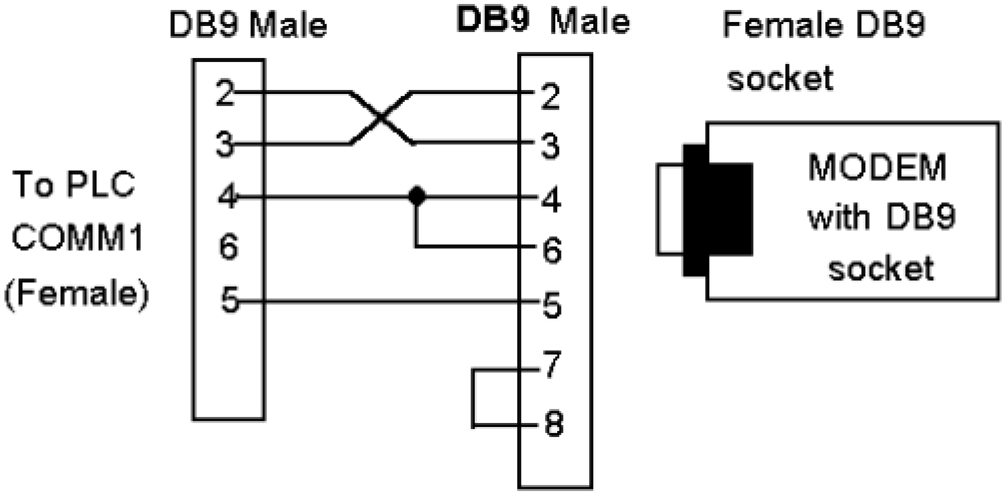
There are a some technical issues that need to be handled carefully in order to successfully implement the modem-linked host communications as described in the following sections.

A4.1 Modem Connection

Modem 1: The host PC may use any internal or external modem that can communicate at 2400 bps or faster. Connect the modem to the PC as instructed in the modem's manual and connect the phone line to the phone jack on the back of the modem marked "WALL" or "Line".



Modem 2: The modem to be attached to the PLC (modem2) must be an external modem with an RS232 connection port. Since modem are DCE type device, they most likely come with a female type DB25 or DB9 socket meant for plugging into the PC's RS232 port. Since the PLC's host link port is also a female DB9, we need to construct a DB9-male-to-DB25-male cable or DB9-male-to-DB9-male cable to link the PLC to the modem, as follow:



A4.2 Communication Speed

When communicating via modems, there are two different definitions of communication speeds that you should be aware of:

- The "DTE Speed" or "line rate" is the serial communication speed between the modem and the device connected to its RS232 port. Most modems can automatically detect the RS232 speed of the device and can assume any speed from 1200, 2400 all the way to 115,200 bps. The first ASCII character they receive from the device will determine the DTE speed that the modem will use to communicate with the device.

- The "modem-to-modem communication speed" is what you read on the modem specifications, such as 33.6Kbps, 56Kbps etc. When two modems are connected, they automatically negotiate for the best speed to communicate between the two of them based on the quality of the phone connection and the maximum speed that both modems are able to achieve. We usually have no control of what speed they choose to communicate. But one thing is for sure, which is that the modem-to-modem speed is always lower than the DTE speed.

Since the default communication baud rate of the 'Super' PLC's RS232 serial port is 38,400 bps, the PLC should send a modem initialization string to the modem on the first scan pulse so that the modem can recognize its default DTE speed (i.e. 38,400 bps) in order to talk to the PLC. E.g. To reset the modem, you just have to send an ASCII string "ATZ" to the modem using the following TBASIC command:

PRINT #1 "ATZ"

If you want your modem to automatically answer to an incoming call (e.g. using TLServer 2.x/3.x modem dialing capability), then you should execute the following TBASIC statement:

PRINT #1 "ATS0=1"

The above statement will tell the modem to answer on first ring, you can also change the number 1 to other numbers, E.g. if ATS0=3 it will answer on the 3rd ring of the phone.

A4.3 Software and Programming

The TLServer 3.0, which is part of the Internet TRiLOGI 6.x software suite, already includes built-in support for dialing a modem. Hence if you are using the PLC in passive answer mode only, all the PLC needs to do is to send a modem initialization string "ATS0=1" using the "1st.Scan" pulse to put the modem in auto-answer mode whenever the PLC is powered up. The PLC does not need to issue any more commands to the modem. Whenever a user wants to communicate with the PLC, he/she will first use the TLServer to dial and connect to the PLC's modem and when the connection is established, he/she will then be able to use the TRiLOGI client or the TRi-ExcelLink program to communicate with the PLC. The fact that the PLC is connected via modem and not via direct RS232 is totally transparent to the client programs. To prevent unauthorized access to the PLC, you may need to use the TBASIC command "SETPASSWORD" to set a protective password.

The great flexibility of these PLCs becomes even more apparent when you realize that you can easily program the PLC to automatically dial in to the TLServer to perform a number of tasks, such as using the PLC's File Service to save or append data to hard disk files, send email to anyone via the Internet or even synchronize its real time clock with the host PC!

A number of examples have been included in the "C:\TRiLOGI\TL6\usr\samples\FileService_Modem" folder when you installed TRiLOGI version 6.x. All these examples make use of a powerful yet easy to use custom function that was written entirely using the standard TBASIC commands (see source code listing in the text box below). You only need to create the following simple ladder circuit to use this function (assuming it is function #10):

```
D$ = "ATDT*802"   ' store the phone number
IF TESTIO(Connected) THEN ' already connected.
    IF TESTIO(DialModem)=0 ' connection no longer needed
      IF DM[3991]=0      ' used as timer for modem attention.
         PRINT #1       ' clear serial-out buffer.
         WHILE INCOMM(1)<> -1 ' clear whatever data in serial-in buffer
         ENDWHILE
      ELSE
         IF DM[3991]=5
            PRINT #1 "+++";      'get modem attention
         ELSE
            IF DM[3991]>=10      'Wait 5 second to gain attention.
                PRINT #1 "ATH" 'hang up modem command.
                CLRIO CONNECTED
              DM[3991]=0
             ENDIF
         ENDIF
      ENDIF
      DM[3991]=DM[3991]+1     'increment the timer
    ENDIF
    RETURN
ENDIF
IF TESTIO(dialModem)=0 RETURN: ENDIF
' If DM[3990] > 0 it means a dialing action has started.
' If DM[3990] > 30 it means more than 30 seconds has passed
' and connection still not established, then retry.
IF DM[3990]=0                  ' Use this DM as a flag
   WHILE INCOMM(1)<> -1 ' clear whatever data in serial buffer first.
   ENDWHILE
   PRINT #1 D$                 ' Dial the number
   DM[3990]=1
   RETURN
ENDIF
A$ = INPUT$(1)
IF LEN(A$) = 0
   DM[3990]=DM[3990]+1     ' also use it to track the time-out
   IF DM[3990] = 28          ' 28 seconds has lapsed.
     PRINT #1 "ATH"
   ENDIF
   IF DM[3990]>=30: DM[3990]=0: ENDIF
   RETURN
ENDIF
SETLCD 4,1,A$
IF STRCMP(MID$(A$,2,7),"CONNECT")=0    ' is connected
   DM[3990] = 0                                ' for next round of connection
   DM[3991] = 0                      ' reset timer for hang-up modem use
   SETIO Connected                  ' set an I/O bit to indicate connection
ENDIF
```

All you need to do is to copy and paste this custom function to your own Ladder+BASIC program, then create an I/O with label name "DialModem" – this may be an input, output, relay, timer or counter contact. The moment this I/O bit "DialModem" is turned on, the PLC will begin to execute the sequence of dialing the remote modem, waiting for a successful connection and then turning on an I/O bit with the label name "Connected". If the dialing cannot be completed within 30 seconds, this custom function will hang up and then re-dial. The process will be repeated indefinitely until either a successful connection is made or if the "DialModem" i/o has been turned OFF.

To disconnect from the modem (hang up), your PLC program just have to turn off the "DialModem" I/O bit and the abovementioned custom function will automatically perform the action of hanging up the modem.

Note: Since the PLC does not have a carrier detect (CD) connection to the modem, therefore if the connection is lost after a successful initial connection, the PLC would have no way of knowing it immediately. Your program would have to detect this condition (e.g. if it sends a file service command and does not receive a "<OK>" acknowledgement string from the host). Once the PLC notes that the connection is lost, it can re-establish the connection by simply turning off the I/O bit with label name "Connected". (say, by executing the "CLRIO Connected" statement). As long as the "DialModem" I/O bit is on, the custom function will re-dial and attempt to make another connection if it notices that the "Connected" bit has been turned OFF for whatever reason.