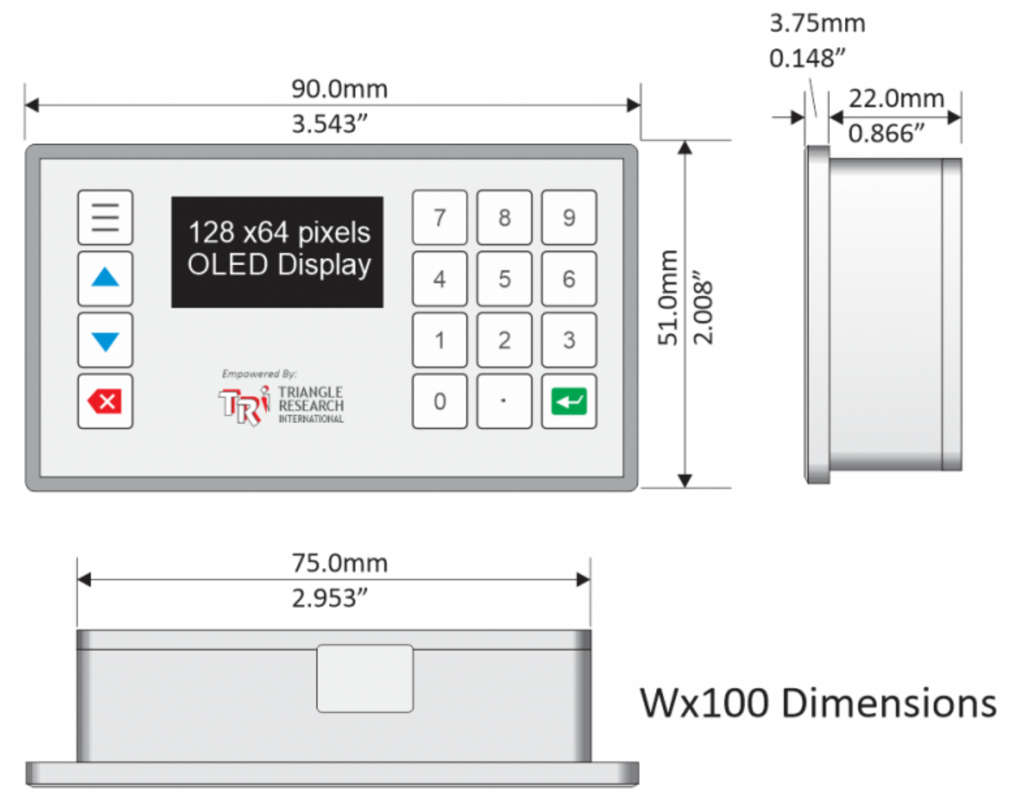
Wx100 Technical Reference

Wx100 User's Manual

Introduction

The Wx100 PLC is the main controller of the new Wx PLC family that has an integrated 16-key keypad and a 128 x 64 pixels OLED display screen. Likes its Fx series PLC cousins, Wx100 PLC supports **full floating-point math** and is programmed using the new i-TRiLOGI version 7.4 software that also support programming of its new keypad and OLED display capability.



Wx100 Dimensions

Wx100 is born as a great M2M communicator that allows it to talk and listen to other TRi PLCs or a huge range of 3rd party intelligent equipment over a LAN, the Internet or simple direct serial link, while performing its full range of control operation at the same time as described below.

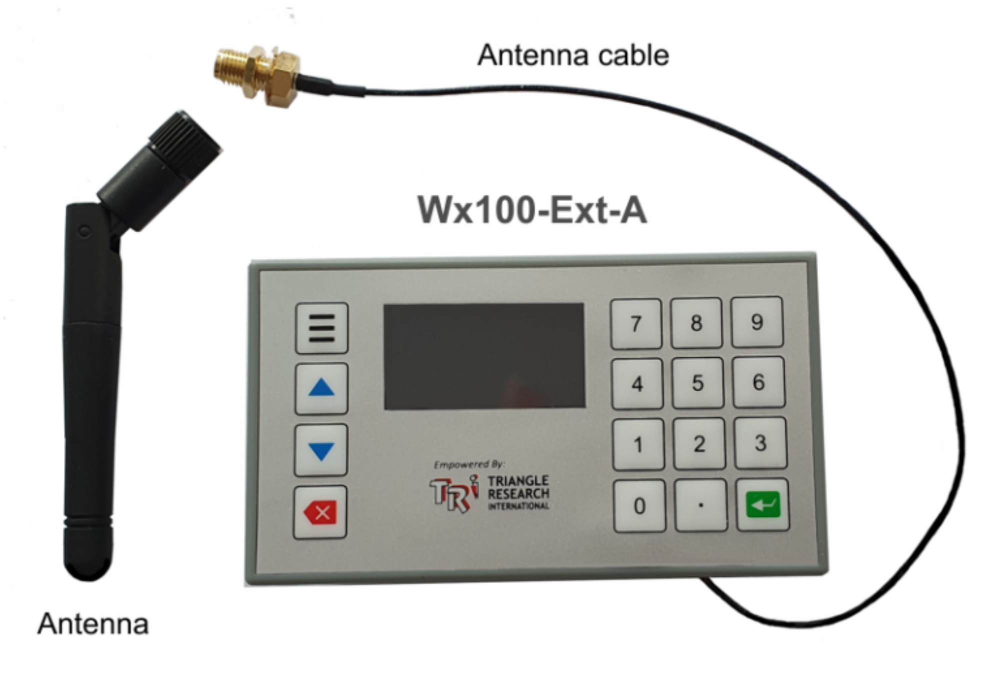
a) Model Options

There are two Wx100 models:

1. **Wx100**: This is a standard model with an internal antenna for the WiFi signal. The antenna is integrated on the module itself. It is designed mainly to be installed on exterior front cover of a control panel. It is not suitable to be installed inside a metallic control panel as the metal can block or greatly diminishes WiFi signal. Note that there is no option to add an external antenna to the standard Wx100.

OLED
Display
7
8
9
4
5
6
1
2
3
0
.
TRIANGLE RESEARCH INTERNATIONAL

2. **Wx100-Ext-A:** This model comes with an antenna cable and an external antenna as shown in the photo below. This gateway model can be installed inside a metallic control panel since the Wi-Fi signal will be captured by the external antenna (see antenna installation instructions (https://docs.triplc.com/#6971)) and conduct via the antenna cable into the CPU.



b) Wireless Networking

Every Wx100 directly supports 2.4GHz Wi-Fi (802.11b,g,n) networking capability up to 72Mbps, allowing it to be connected directly to any Wi-Fi access points or router. It supports the FServer (for remote programming or monitoring) and a Modbus/TCP server over-the-air (for access by third party devices). Each server supports multiple simultaneous connections.

The PLC can also be programmed to as a TCP client to easily connect to other PLCs or Modbus/TCP slave devices via the Internet to exchange data, or connect to the cloud to upload real time data or historical log file that it has created (via FTP). It can connect to an Internet Time Server to get the most accurate internet time to set up its own real-time clock. The Wi-Fi port supports “Web Services”, allowing enterprise software, such as a database program or MS Excel, to query for information from multiple PLCs instantaneously.

OEM can also create web apps to let user control or monitor the PLC from a web browser on PC or smartphone alike.

c) Radio Frequency Compliance

> Wx100 contains FCC-certified radio module and is in compliance with FCC Title 47 CFR Part 15 / ICES-003 Issue 7-(Class A). Click to view the Testing Laboratory Certificate of Comformity (https://triplc.com/documents/E11213-2101_CoC_Triangle Research_Wx Systems_Rev-0.1-4.pdf) and the Supplier's Declaration of Conformity (https://triplc.com/documents/FCC_SDoC_For_Wx100.pdf).

d) RS485 Networking

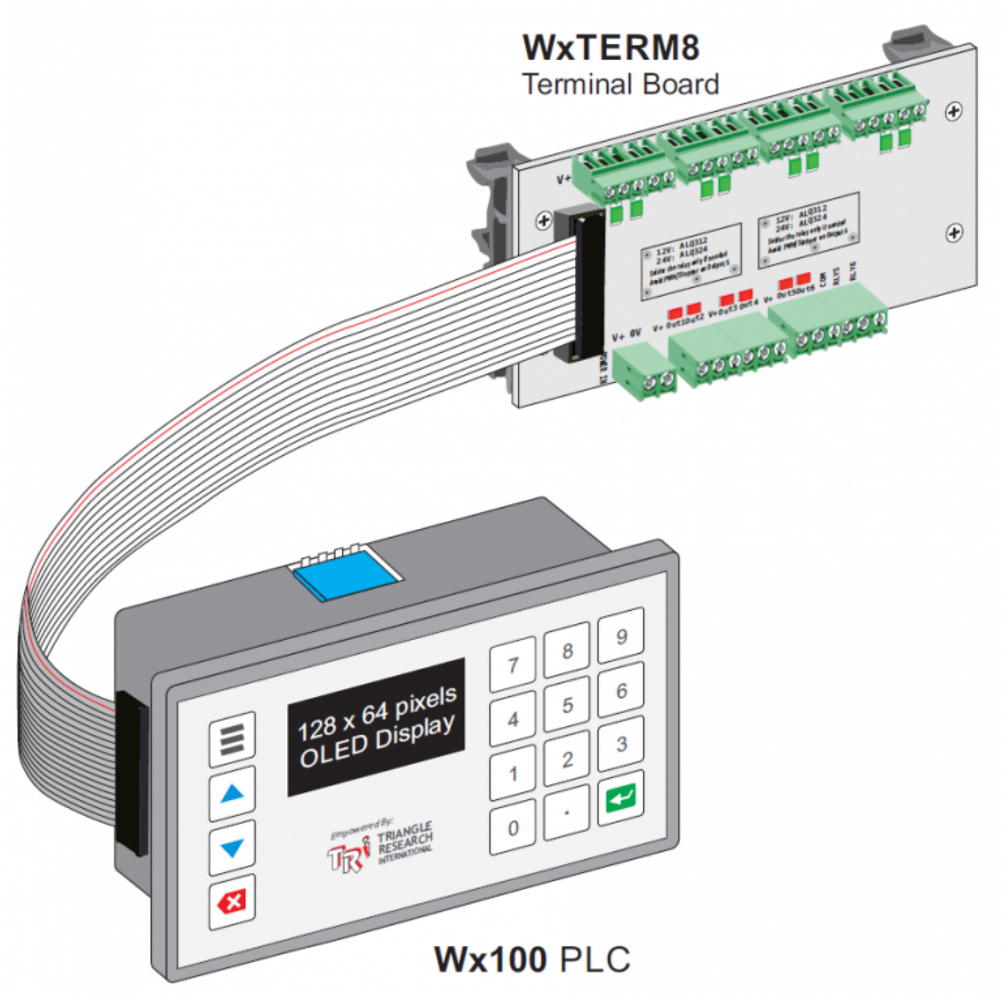
Every Wx100 has a built-in two-wire RS485 port that can communicate with another PLC or third party devices via the industry standard Modbus RTU serial communication over a wide range of baud rate and communication format. Wx100 can act as a Modbus RTU master or slave, allowing it to communicate with a large number of remote I/O boards placed at up to 1200m (4000 ft) away. In addition, Wx100 can also be setup as a **ModbusTCP to ModbusRTU Gateway**, thus enabling old Modbus RTU equipment to be connected via TCP/IP network over the air. Wx100 can therefore be used to bridge the old generation of Modbus RTU device to the new generation of IIOT capable SCADA masters.

e) Basic I/Os

The basic Wx100 PLC comprises 8 PNP/NPN configurable, 24V digital inputs , and 6 NPN digital outputs (sink up 0.5A, 9-24VDC). 6 of the digital inputs can also be used as 0.5 to 5V analog inputs. Furthermore, 6 of the digital inputs can also be used for decoding and measuring pulses received from up to 3 digital encoders, allowing you to measure position and velocity of a moving object in real time.

The 6 digital outputs on the Wx100 PLC can also double as 6 channels of PWM output, or 3 channels of stepper motor controller. 4 of the digital outputs can even be configured to drive a small stepper motor directly – saving you the expense of buying a separate stepper motor driver!

All Wx100 PLC I/Os are made available on a 20-pin, 0.1" spaced header pins, allowing the signals to be easily connected to a terminal board for quick and simple field wiring. TRi supplies a low cost PCB based terminal board model# **Wx-TERM8 (https://docs.triplc.com/wx-term8-um/)** that should satisfy most applications that do not need more than the above-mentioned basic I/Os. Using Wx-TERM8 with Wx100 PLC is the simplest way to get started using this powerful PLC for designing your small machines or sophisticated industrial automation projects.

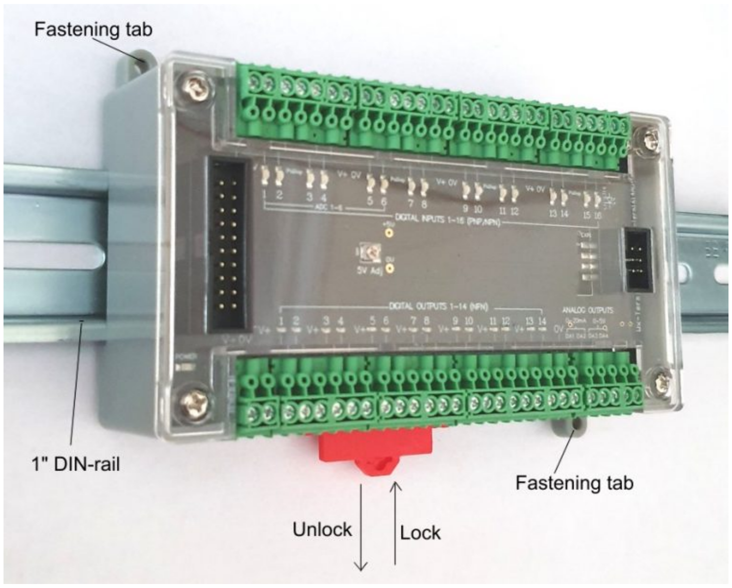


f) Expanded I/Os

1. Standard Wx-TERM16 Module:

OEM or automation system integrator can also purchase Wx100 with the off-the-shelf, Wx-TERM16 (https://docs.triplc.com/wx-term16-um/) module (instead of Wx-TERM8 board) supplied by TRi that provides:

- 16 PNP/NPN configurable Digital Inputs,
- 6 analog inputs (shared with digital inputs 1-6),
- 14 NPN digital outputs and
- 4 analog outputs (2 channel 0-20mA, 2 channel 0-5V).



(https://docs.triplc.com/wx-term16-um/)

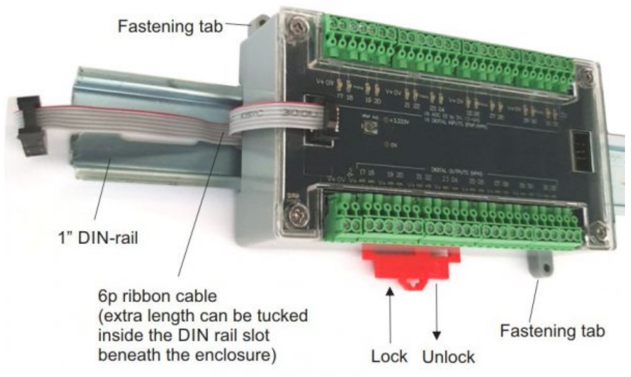
For more information please view Wx-TERM16 Installation Guide (https://docs.triplc.com/wx-term16-um/)

Wx-TERM16 also provides an expansion connector for further I/O expansion using the cascadable, Wx-M1616 expansion I/O modules (http://docs.triplc.com/wx-m1616-um/) that can increase the total number of I/Os to the following limits:

I/O Type	**Maximum Quantity**
Digital Inputs	80
Digital Outputs	78
Analog Inputs	70
Analog Outputs	32

Table 1.1

For more information about this expansion module, please refer to Wx-M1616 Installation Guide (http://docs.triplc.com/wx-m1616-um/).



(http://docs.triplc.com/wx-m1616-um/)

2. Fully Customized I/O Board: If you are a mid to high volume OEM equipment maker that needs more I/Os that what is available on the basic or expanded I/O boards, you can work with TRi to create a **customized I/O board** that contains any combination of digital inputs, digital outputs, up to limits shown in Table 1.1.

Note: The I/O limits in Table 1.1 are imposed by the standard PLC firmware. It is possible to expand beyond the above limits for specialized applications but may also require customized firmware development. Please consult TRi's engineer if you have special applications that goes beyond the above limits.The customized I/O board together with the Wx100 thus becomes your own **Custom PLC**!

Having your own Custom PLC is a great solution for OEM equipment maker for the following reasons:

- **Improve Product Design**: Your custom PLC can be highly optimized to match the form factor of the equipment. You can define the type of I/O terminal that suit your applications best. For example, an OEM may prefer to bundle external I/Os wiring to the PLC's I/O board via a wire harness. In such a case the best terminal would be a quick connect receptacle for the wire-harness. This enables quick connect/disconnect of all the I/Os during assembly and field service replacement.

- **Save Cost and time:** You get the exact I/O types (12V, 24V, PNP or NPN for digital I/Os, 0-5V, 0-10V or 4-20mA for analog I/Os) and the precise number of I/Os you need without paying for excess I/Os that you will never use.

- **Improved Corporate Image:** If you order the Wx100 PLC with a private-labelled face-plate/keypad (https://docs.triplc.com/wx100-um/#2571) to go with your customized I/O board, then you have essentially created your own private-labelled PLC that besides enhancing your corporate image, would also increase barrier to competition or copycat.

- **Reduce Supplier Uncertainty:** Having your own custom PLC gives you greater control over when and how to change your product design. You no longer have to worry about availability of a PLC I/O module and whether the PLC manufacturer may decide to stop supplying you a less popular I/O board down the road.

- **Reduce Troubleshooting**: A tailored, simplified PLC design makes machine maintenance easier, reducing the training required for new designers or maintenance technicians and minimizing design and maintenance errors.

Chapter 1 to 10

Chapter 1 - Installation & Wirings

1.1 Installing Wx100

A complete Wx100 PLC system comprises two parts:

- The Wx100 PLC main controller
- The WxTermXX – the wiring terminals and expanded I/O on some models.

The two parts are connected via a 20-pin ribbon cable connector which brings the I/Os from the Wx100 PLC to the wiring terminals to connect to physical devices such as sensors and loads.

With its built-in HMI that comprises a 128 x 64 pixel OLED display and a 16-key keypad, the Wx100 PLC would most commonly be installed on the front door of a control panel or on the exterior of a control box where user can interact with it directly.

Wx100 may also be deployed as the brain of a subsystem inside a bigger equipment, E.g. to control the cooling system of a MRI machine. In such applications, the Wx100 may be installed inside the control panel or on a DIN rail if the OEM prefers not to let its user to have direct access to the keypad and display on the Wx100.

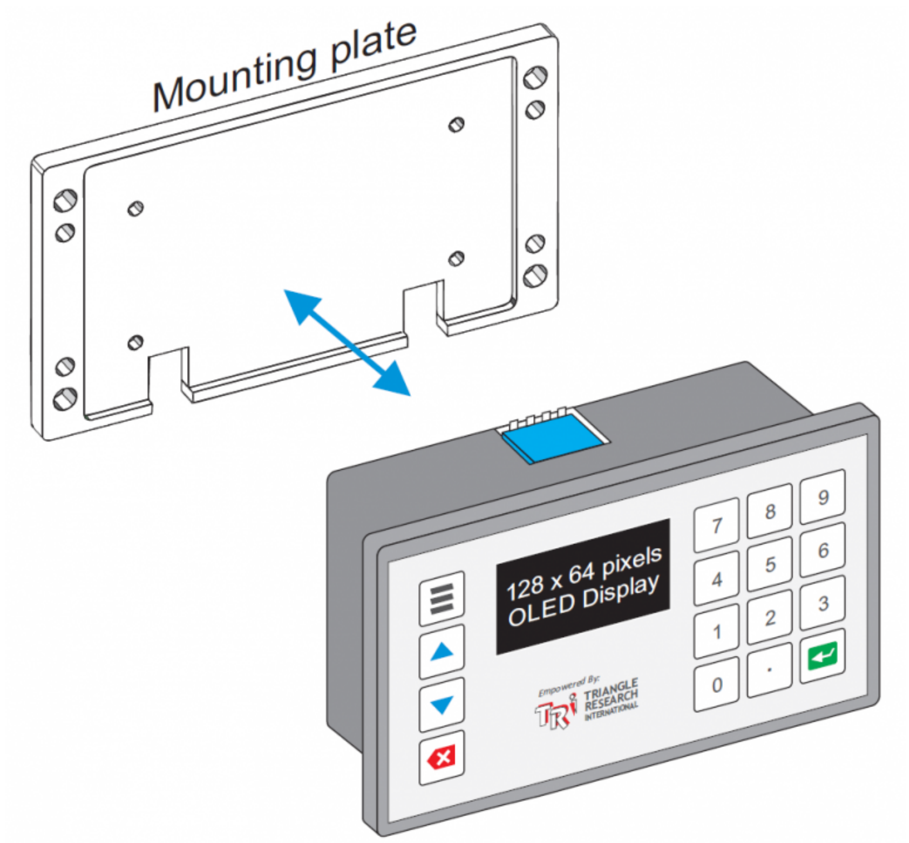
Note: When installed inside the control panel, the built-in HMI on a Wx PLC can still be an extremely effective maintenance and monitoring tool if properly designed, as it can constantly display critical performance data, charting the data trend etc and provide alert messages. OEM can easily create a menu system to allow the maintenance technician to access a wide range of data or control the operation of the subsystem with only a few key presses.

The innovative Wx100 enclosure is designed to be highly flexible which allows it to be readily installed in one of the above scenario.

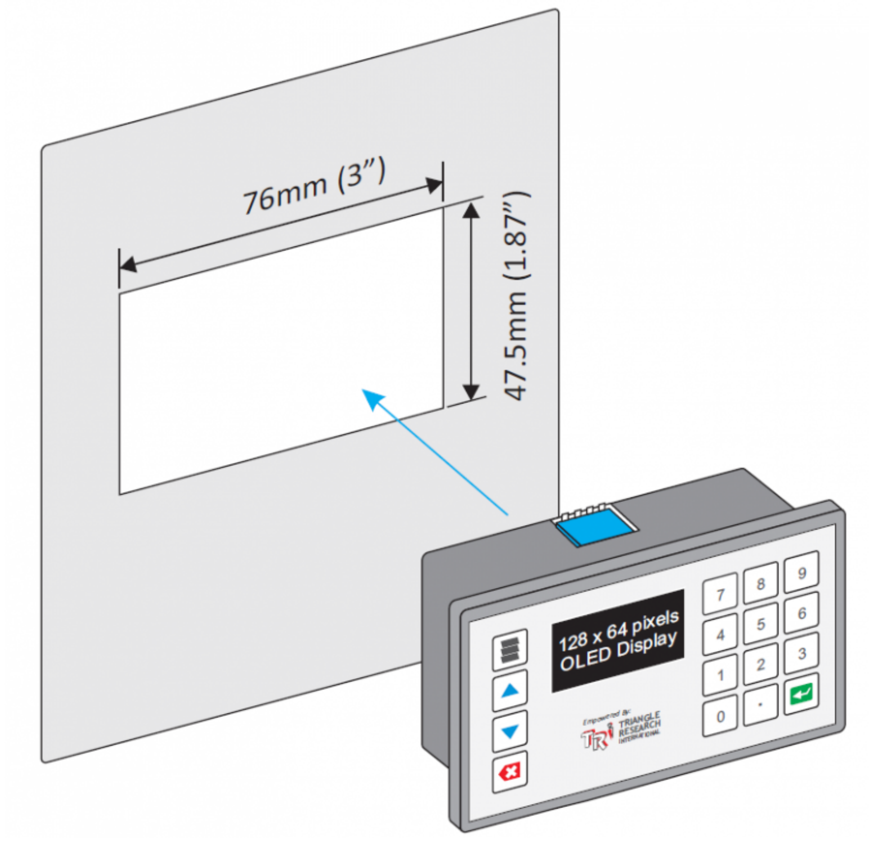
a) Front Panel Mount

Installing Wx100 PLC on the control panel front door

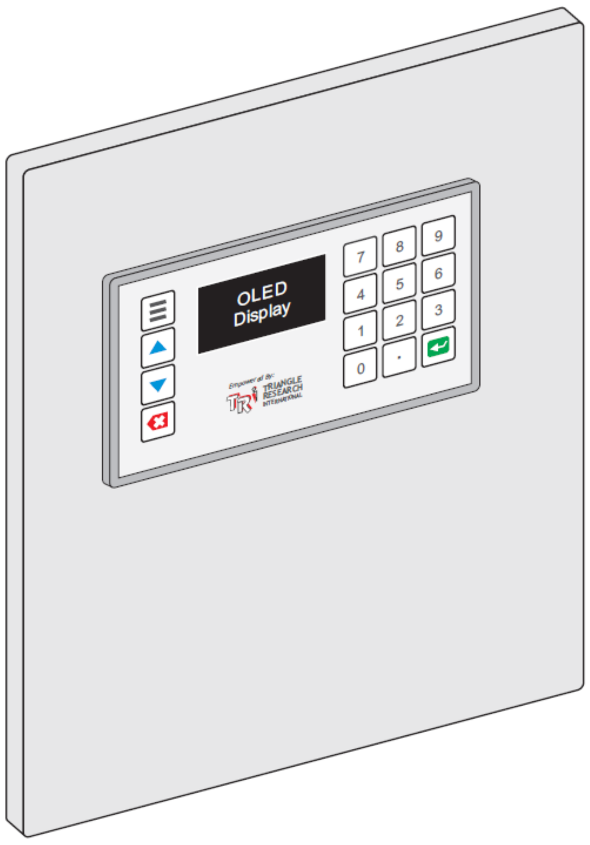
1. Detach the mounting plate from the back of the Wx100 PLC body if it is attached there. A new Wx100 may be shipped with the mounting plate loosely attached to the back of the Wx100 PLC body and it can be easily removed by hand without tools.However, if you find that the mounting plate has already been fastened to the back of the Wx100 using 4 screws, then remove those 4 screws to free it from the Wx100 body.



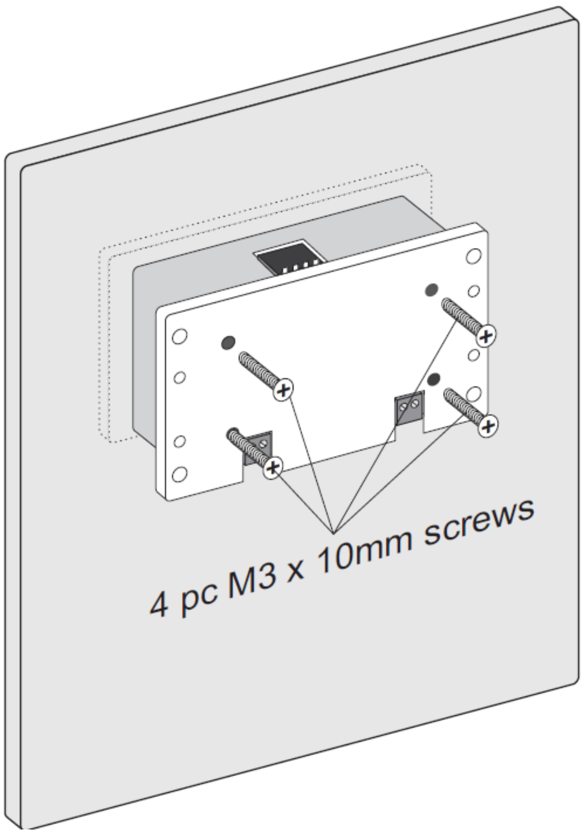
2. Cut a rectangular hole of size 76mm x 47.5mm (3” x 1.87”) on the front door. Insert the Wx100 main controller from outside of the control box through the cut-out area.



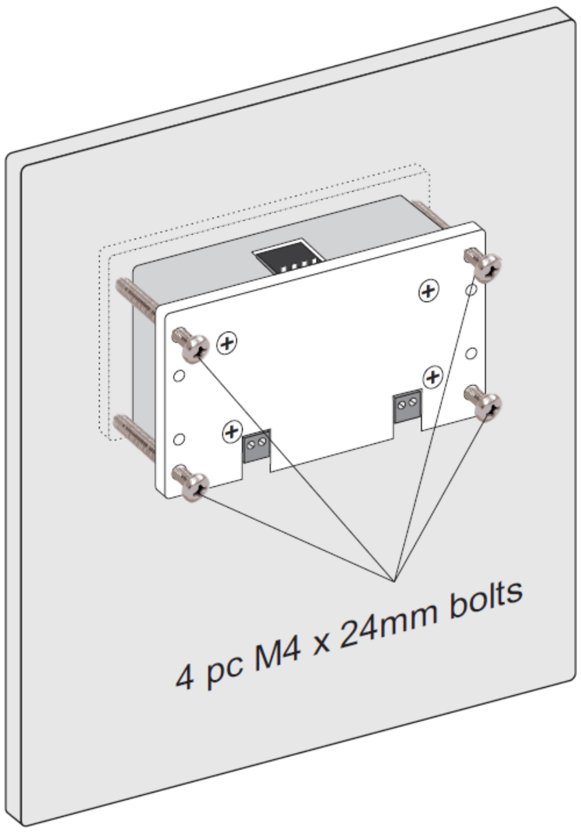
3. The Wx100 PLC should fit snugly into the cut-out area on front door. Then turn to the back of the front door.



4. Attach the mounting plate to the back of the Wx100. Fasten it using 4 pieces of M3 x 10mm screws supplied with the package.



5. Finally, insert the 4 pieces of M4 x 24mm bolts into the 4 larger holes on the mounting plate. Tighten the bolts to lock the Wx100 against the front door that it is mounted to.



b) DIN-Rail Mount

Installing Wx100 On A DIN-Rail

Wx100 can also be easily installed inside a standard industrial control panel or any control boxes to prevent end-user access to the built-in HMI on the Wx100.

Note:

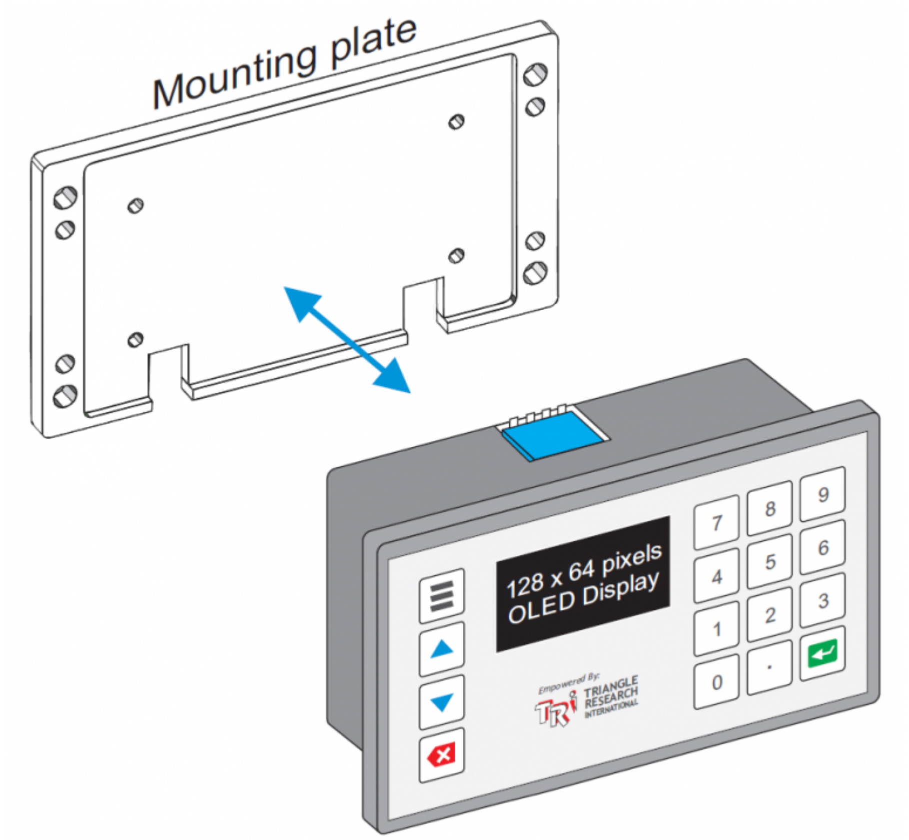
1. If you intend to install the PLC inside a metal control panel, then consider the model: **Wx100-Ext-A** (https://docs.triplc.com/#6971) which has an external antenna (https://docs.triplc.com /#6971) that can be installed outside of the control panel.

2. The standard Wx100 PLC (with integrated antenna) can also be installed inside a plastic control panel, or even a metal control panel Wi-Fi connection is not required for normal operation.

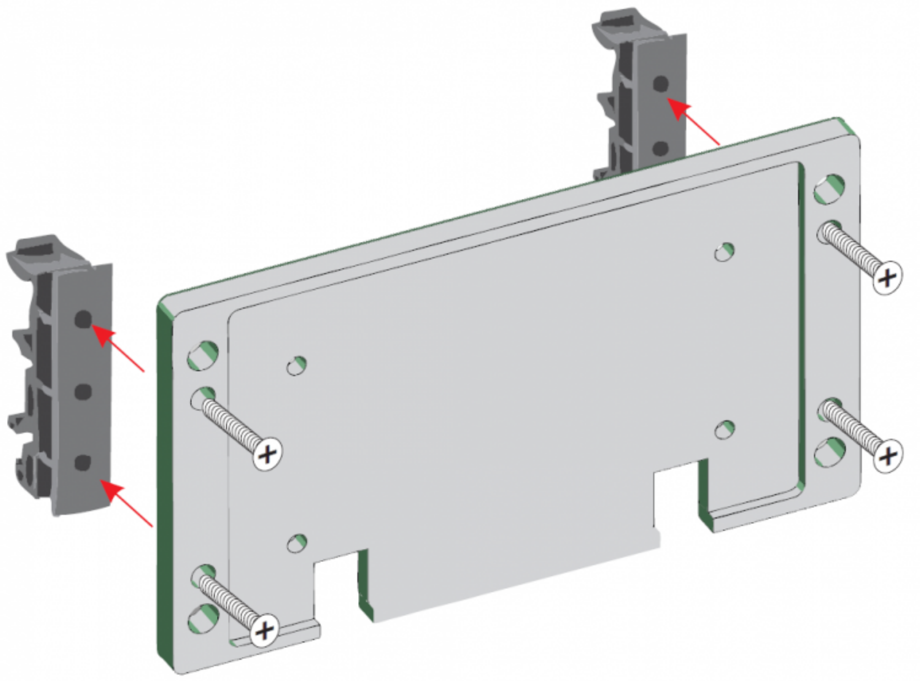
You can install Wx100 directly on a DIN-rail using the DIN-KIT-0 (purchased separately) as shown below. The two DIN clips that are attached to the mounting plate using 4 pcs of M3 screws included in the DIN-KIT-0 allows the whole assembly to be easily snapped on to or removed from an industry standard DIN rail, as illustrated in the right-hand side picture below (without the controller to more clearly show the DIN clips in action):

Please follow the procedure listed below to assemble the two pieces of DIN-CLIPs to the Wx100.

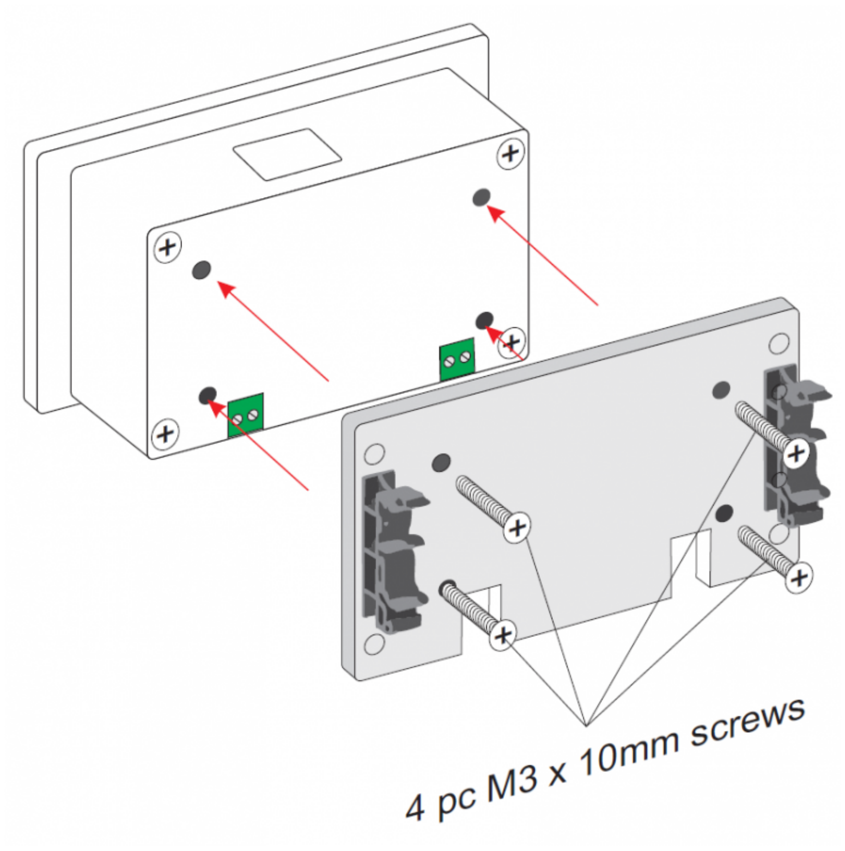
1. Detach the Wx Mounting plate from the back of the Wx100 PLC body if it is attached there. A new Wx100 may be shipped with the mounting plate loosely attached to the back of the Wx100 PLC body and it can be easily removed by hand without tools. However, if you find that the mounting plate has already been fastened to the back of the Wx100 using 4 screws, then remove those 4 screws to free it from the Wx100 body.



2. Attach the two DIN clips to the back of the Wx Mounting plate using 4 pieces of M3 screws (supplied with the DIN-KIT-0)



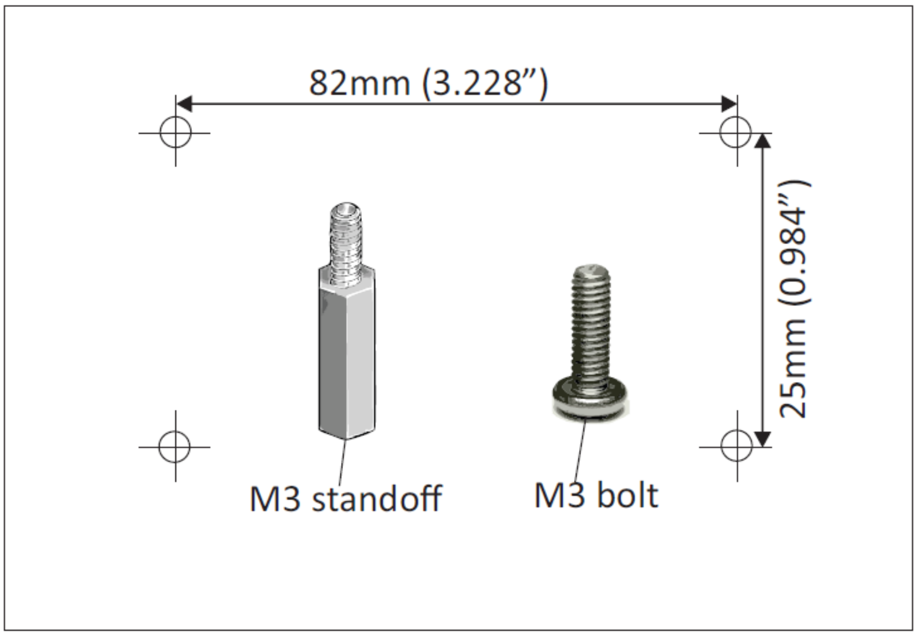
3. Turn the mounting plate over and attach to the back of the Wx100 Controller. Fasten the mounting plate to the Wx100 using 4pcs of M3 x 10mm screw (supplied with the Wx100 PLC). That completes the assembly and you can now freely clip or remove the Wx100 assembly from the DIN rail.



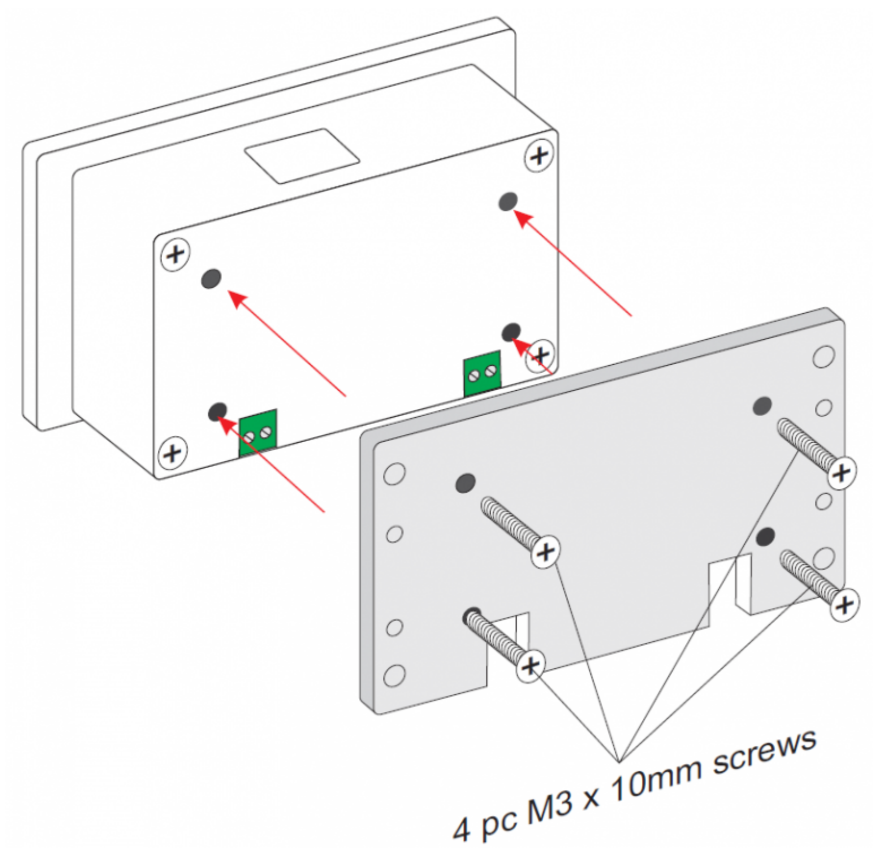
c) Direct Mount

Installing Wx100 PLC Inside A Control Box Without DIN Rail

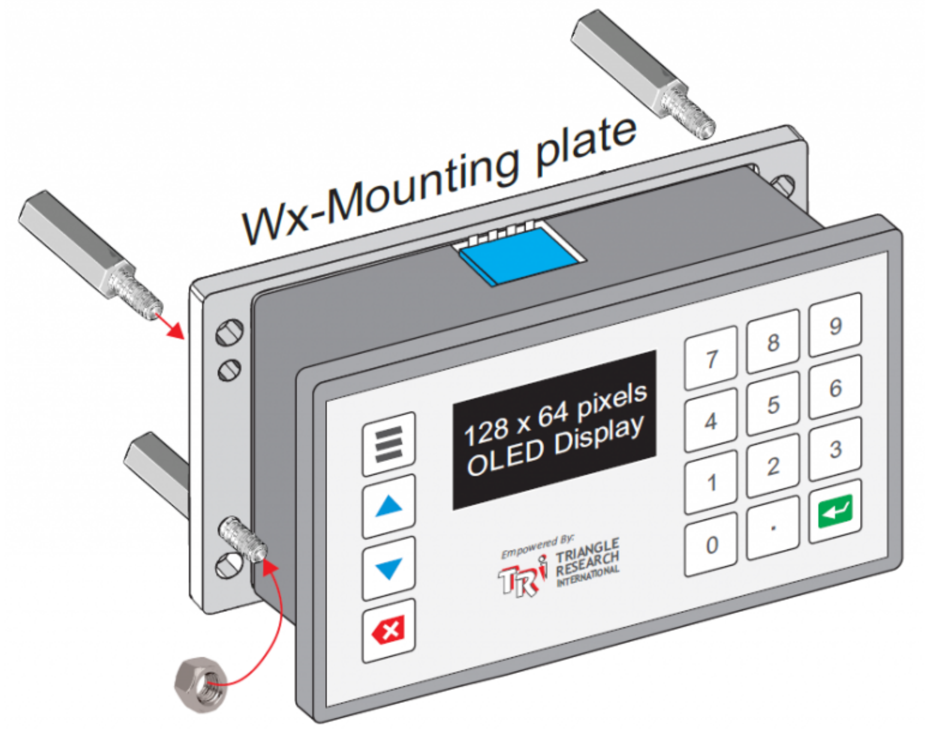
1. Install four pieces of M3 standoffs, or just 4 pieces of M3 bolts inside the control box at the following locations:



2. Next, attach the Wx Mounting-plate to the back of the Wx100 PLC and fasten it using the 4 pieces of M3 x 10mm screws (supplied with the Wx100 PLC).



3. Place the Wx100 PLC over the 4 standoffs (or 4 bolts with or without spacers) and fasten each with an M3 nut as shown below.



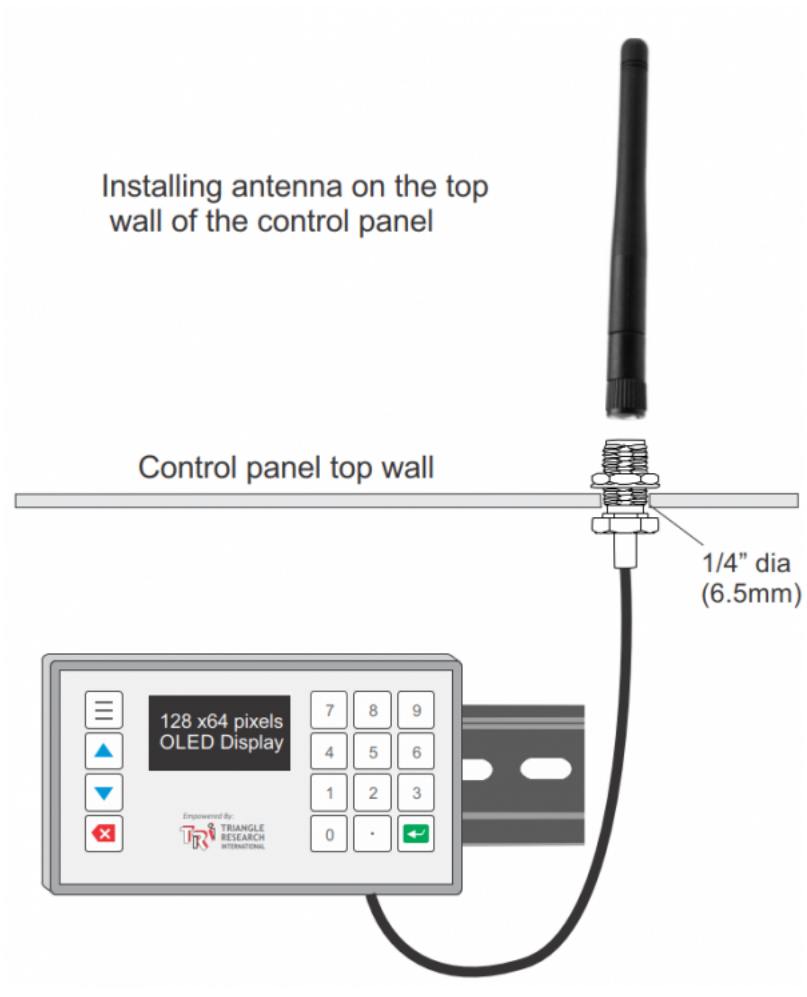
Note: Installing the Wx100 PLC directly onto the bottom surface of the control box is the most compact form of installation as it takes up very little head space, thus allowing Wx100 PLC to be deployed inside the tightest possible space. However, since the screw terminals on Wx100 is inaccessible after installing it this way, you should connect the required wires to the Wx100 power and RS485 (if used) **before** installing it using this method.

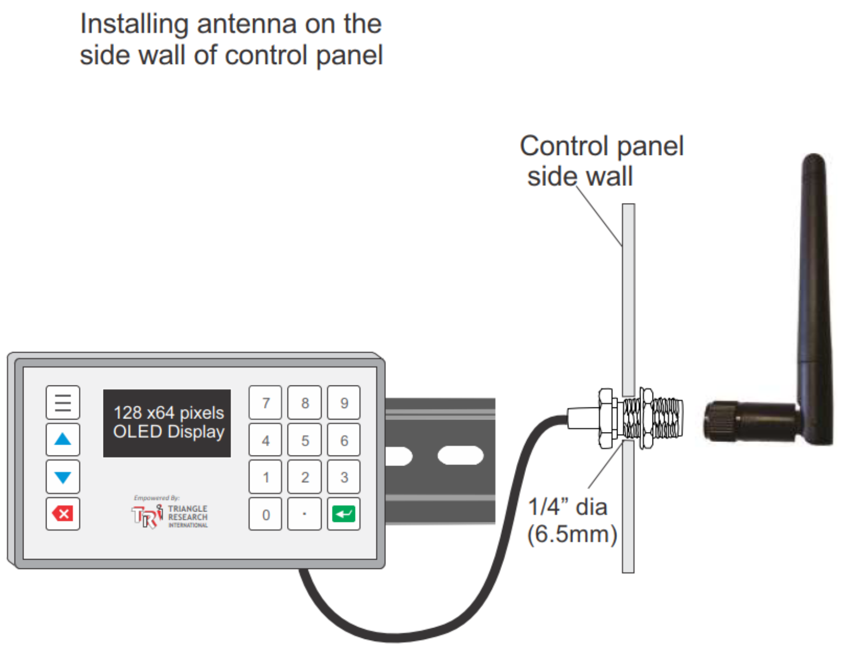
d) External Antenna Installation

The model Wx100-ExtArequires the antenna to be installed on the outer surface of the control panel. The supplied antenna can be adjusted from 0 to 90 degree, allowing flexible options to position the antenna.

Notes:

1. Use only the antenna supplied with the device to remain compliant to the FCC/ISED/CE EMC certifications.
2. If you use an alternative antenna, then you must re-certify the product to meet your local EMC standards.
3. The antenna cable attached to the Wx100-Ext-A PLC is about 7" long. This means that in order to avoid adding extension antenna cable (which would attenuate the signal strength), the Wx100-Ext-A-PLC should be installed close to the side or the top of the control panel, as illustrated in the following two installation methods:


Installing antenna on the top wall of the control panel
Control panel top wall
1/4" dia (6.5mm)
128 x64 pixels OLED Display


Installing antenna on the side wall of control panel
Control panel side wall
128 x64 pixels OLED Display
1/4" dia (6.5mm)

1.2 Installing Wx Terminal Board

TRi will be supplying several models of Wx Terminal boards with different combinations of digital and analog I/Os to fit most common applications. The main purpose of the terminal board is to provide simple wiring terminals for user to connect the Wx100 PLC to the industrial devices. For the most basic terminal board with no more than 8 inputs and 6 outputs, only mainly passive components are used on terminal along with some red and green LED indicators to indicate the ON/OFF state of the I/Os.

OEMs may like to take note that this unique wiring design of the Wx100 PLC opens the possibility for OEMs to very easily custom-design and build their own terminal boards that best fit their applications. This may be done to fit the available footprint for the control components, or to save cost.

For example, some applications may require all digital outputs to be electromechanical relays. Some applications may need to drive some AC loads using solid state relays and some may want to install pressure sensor and signal conditioners directly on board the terminal boards. Building a customized terminal board lets you eliminate the needs to wire from the PLC's terminals to yet another board that contains custom circuitry, thus saving considerable labour cost and component costs.

We recommend that first time Wx100 user purchase one of the following standard terminal boards that TRi supplies to quickly deploy the Wx100 for real world applications:

a) Low cost, PCB based **Wx-TERM8** with 8 DI/6 AI and 6 DO, no AO
Please refer to the Wx-TERM8 documentation: https://docs.triplc.com/wx-term8-um/ (https://docs.triplc.com/wx-term8-um/)

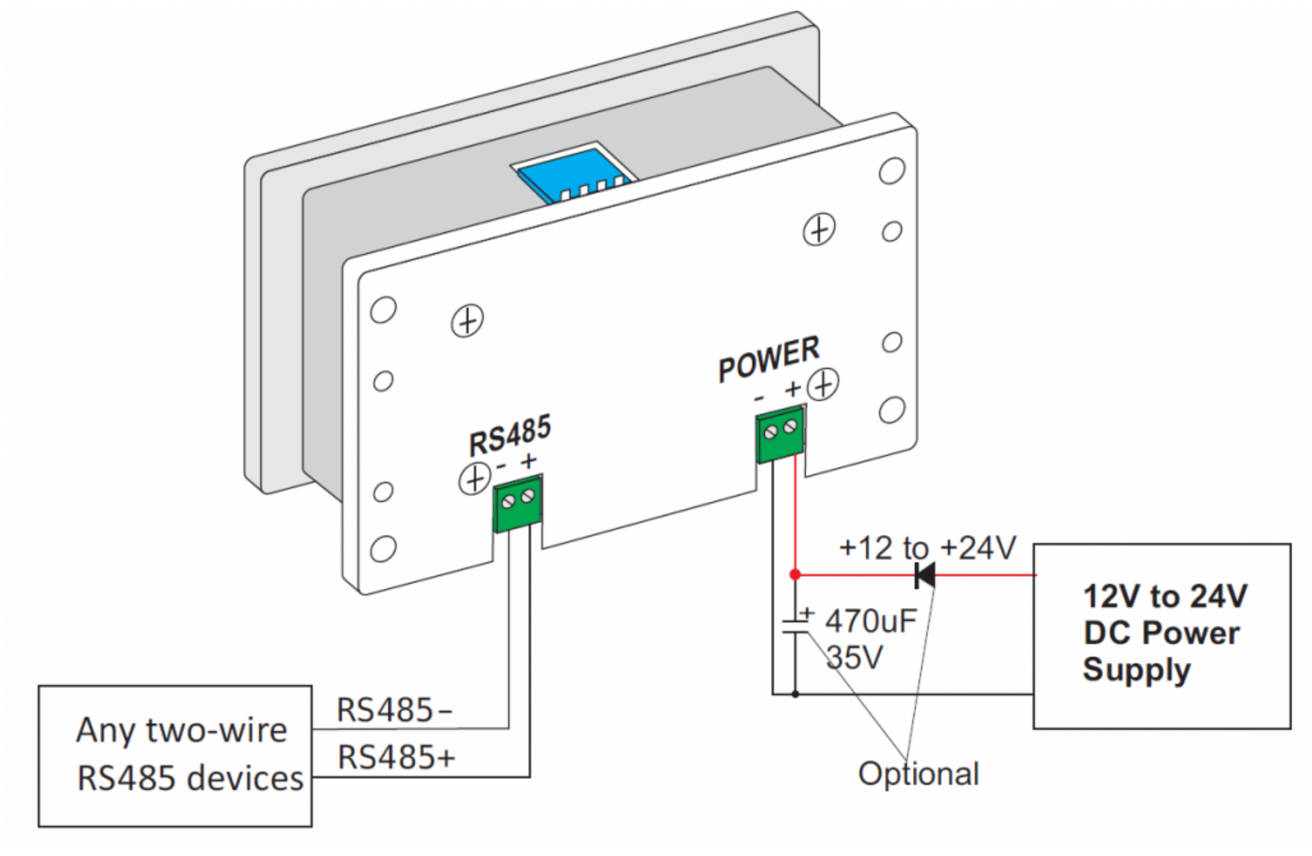
b) Standard **Wx-TERM16** with 16 DI/6AI, 14 DO and 4 AO (full part number: Wx-Term1614AO4-Std).
Documentation: https://docs.triplc.com/wx-term16-um/ (https://docs.triplc.com/wx-term16-um/)

In the rest of this chapter will focus our discussion mainly on how Wx100 + Wx-TERM8 combination as a quick start guide for user applications.

1.3 Wiring Wx100 PLC

a) Wiring Power & RS485

On the bottom side of Wx100 you will find two pieces of two-position screw terminals as shown below:



The Wx100 PLC requires a single regulated, 12 to 24V (+/- 5% ripple) DC power supply for the CPU. The PLC typically consumes less than 100mA and thus you may also use the same power supply to power the CPU and the load that are wired to the terminal board (more on that later).

Please use only an industrial grade regulated power supply from established manufacturers.

Using a poorly made switching power supply can give rise to a lot of problems if the noisy high frequency switching signals are not filtered properly.

Always place the power supply as close to the PLC as possible and use a separate pair of wires to connect the power to the PLC. Keep the power supply wires as short as possible and avoid running them alongside high current cables in the same cable conduit. The Wx100 PLC will be reset when the power supply voltage dips below 7V. It is a good idea to connect a 470mF to 1000mF, >35V electrolytic capacitor near the power supply connector to suppress any undesirable voltage glitches from conducting into the PLC. If other high current devices, such as a motor driver, were to affect the operation of the PLC, you should then also connect a diode before the capacitor to prevent reverse current which might flow back to the power supply, as shown in the above diagram.

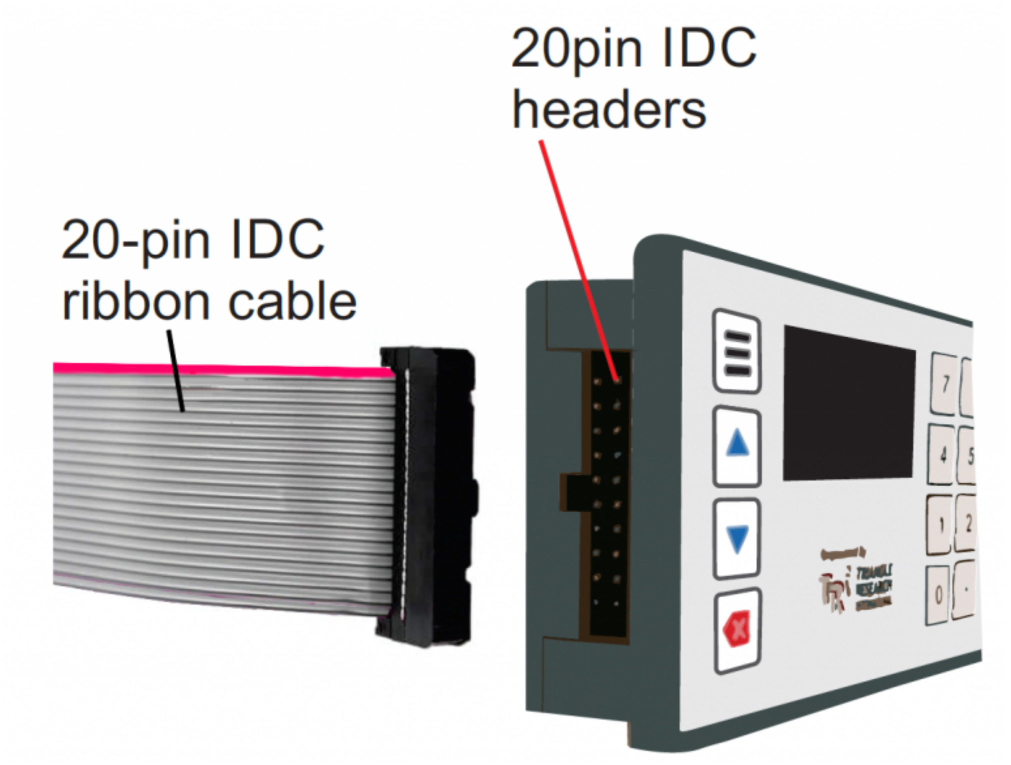
If the AC main is affected by nearby machines drawing large amounts of current (such as large three-phase motors), you should use a surge-suppressor to prevent any unwanted noise voltage from being coupled into the Wx100 power supply.

The required current rating for the power supply depends mainly on the total output current, taking into consideration the peak current demand and the duty cycle of the operation.

b) Wiring Physical I/Os

Wx100 PLC’s physical I/Os are available on a 0.1” (2.54mm) pitch, 20pin IDC header. This allows the I/O signals to be easily wired from the Wx100 PLC to a terminal board using a simple, low cost 20pin IDC ribbon cable.

After installing the Wx100 PLC, you can remove the dust cover from the side body to expose the IDC headers. Simply plug in the 20pin IDC ribbon cable supplied in the Wx terminal board package to the headers to complete the connection.



Input1/AI1	1	2	Input2/AI2
Input3/AI3	3	4	Input4/AI4
Input5/AI5	5	6	Input6/AI6
Input7	7	8	Input8
Reserved	9	10	Reserved
Reserved	11	12	Reserved
Output1	13	14	Output2
Output3	15	16	Output4
Output5	17	18	Output6
V+ IN	19	20	0V

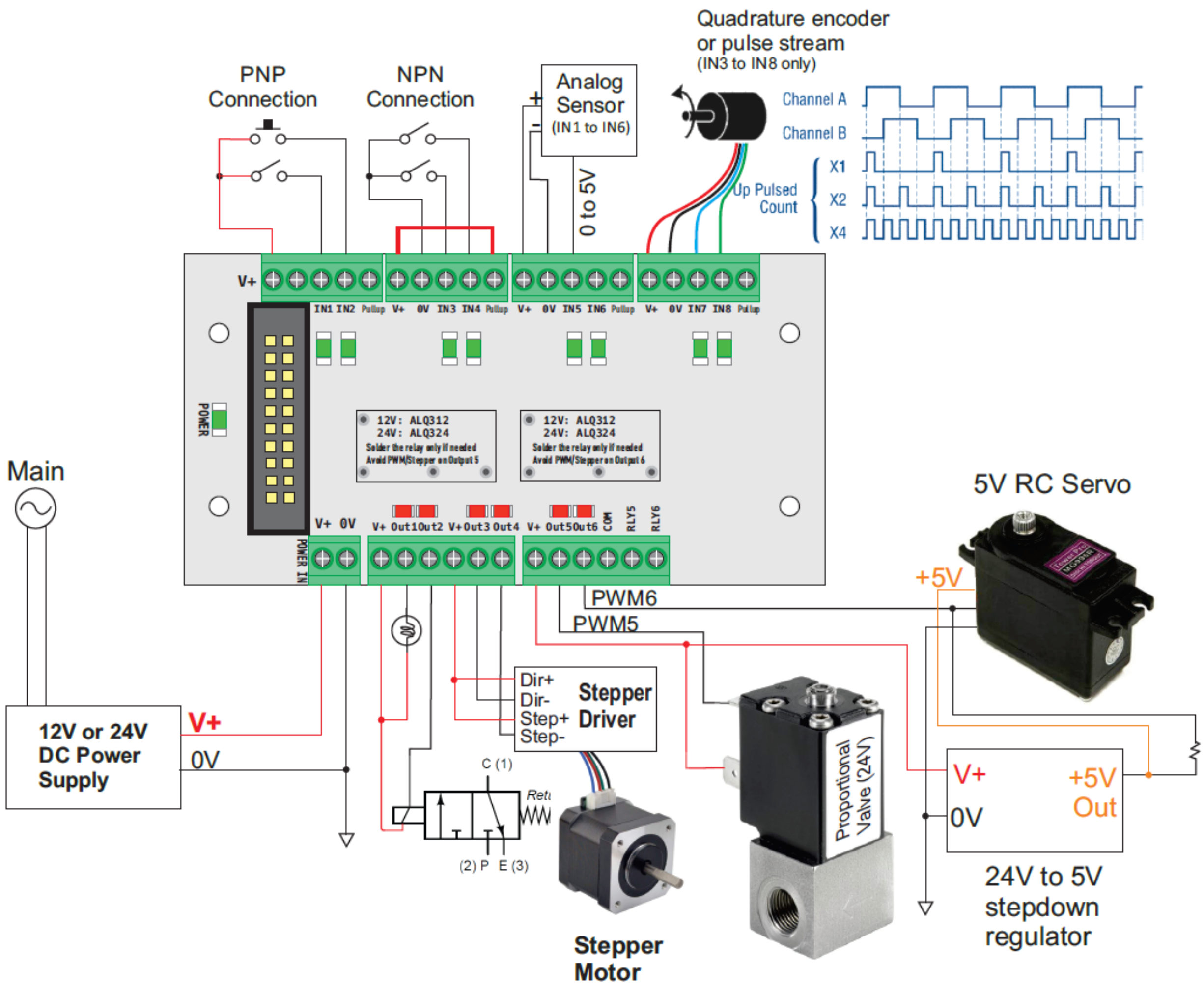
The I/Os on the Wx100 are all **multi-talented!** The inputs can be used as pure digital inputs, analog inputs (IN1 to IN6), quadrature encoder inputs (IN3 to IN8), and pulse measurement inputs (IN3 to IN8). All 6 digital outputs can also become PWM outputs, stepper motor controller or stepper motor driver outputs.

All Wx100 I/Os are industrial voltage level and can operate from 9V to 30VDC. The following are the specifications of the I/O pins when used together with the Wx-TERM8 terminal Board:

Electrical Characteristics (When Vp = 24V)

	MIN	TYP	MAX	UNIT
IN1-8 Logic ‘1’				
Voltage	8	–	30	V
Current (sink)	0.055	–	0.21	mA
IN1-8 Logic ‘0’				
Voltage	-0.3	–	3	V
Current (sink)	0	–	0.021	mA
OUT1-6 Logic ‘1’				
Voltage	0	0.1	0.8	V
Current (sink)	0.0024	0.5	2	A
OUT1-6 Logic ‘0’				
Voltage	22	–	30	V
Current (sink)	0	0	0	mA

1.4 Wiring Wx-TERM8



Quadrature encoder or pulse stream (IN3 to IN8 only)
PNP Connection
NPN Connection
Analog Sensor (IN1 to IN6)
0 to 5V
Channel A
Channel B
Up Pulsed Count
X1
X2
X4
V+
IN1 IN2 Pullup V+ 0V IN3 IN4 Pullup V+ 0V IN5 IN6 Pullup V+ 0V IN7 IN8 Pullup
POWER
12V: ALQ312 24V: ALQ324 Solder the relay only if needed Avoid PWM/Stepper on Output 5
12V: ALQ312 24V: ALQ324 Solder the relay only if needed Avoid PWM/Stepper on Output 6
V+ 0V V+ Out1 Out2 V+ Out3 Out4 V+ Out5 Out6 COM RLY5 RLY6
POWER IN
Main
5V RC Servo
+5V
PWM6
PWM5
12V or 24V DC Power Supply
V+
0V
Dir+ Dir- Step+ Step-
Stepper Driver
C (1)
(2) P E (3)
Stepper Motor
Proportional Valve (24V)
V+
0V
+5V Out
24V to 5V stepdown regulator

Note:

- In the next few sections we will describe in greater details how the power supply, the digital inputs and digital outputs should be wired to the Wx-TERM8 board.
- The specifications and programming methods for the analog I/Os are detailed in Chapter 5 of this manual. Chapter 6 shows how the special inputs and outputs such as quadrature encoder, stepper motor driver etc. are mapped to the Wx100 digital I/Os.

1.5 Power Supply

a) PLC and I/O Using Same Power Supply

A regulated, low ripple industrial grade DC power supply between 12 to 24V should be used to power a Wx100 PLC application.

Wx100 PLC itself consumes less than 50mA (0.05A) at 24VDC when all its I/Os are turned OFF. For most applications Wx100PLC can share the same 12 to 24VDC power supply as the load. The output driver is capable of sinking 2A continuously per output. Hence the power supply current output capacity is mainly determined by the maximum total amount of output current need to power the digital output loads.

You should connect a separate pair of wires (AWG24 or larger) from the power supply to the Wx100 PLC as shown in section 1.3.1. Beside powering the Wx100 PLC, this pair of connecting wires is also tasked to carry the return current that the PLC output sinks from the load back to the power supply.

The power to the I/O should be connected to the bottom left screw terminals marked as “POWER IN” on the Wx-TERM8 as shown in Section 1.3.3. The power supply input is in-turn routed by the Wx-TERM8 PCB to the INPUT and OUTPUT terminal sections for easy connection to I/O devices.

Note:

Internally, the Wx-TERM8 also routes its power supply (via a reverse blocking diode) to the Wx100 via pin19 and 20 on the ribbon cable. This means Wx100 PLC will receive power even if you don’t connect the power supply cable to its own power supply screw terminals. However, we **do not** recommend you depend solely on this connection to power the Wx100 unless your output loads consume only very small amount of current. This is because If you don’t connect separate wires to the Wx100 power connector, then the total output current that the Wx100 output drivers sink from the 6 output loads can only return to the power supply via the ribbon cable, which has limited current carrying capacity and could lead to overheating if it has to constantly carry large amount of return current.

b) Using Separate Power Supplies for the PLC and the Load

If the output loads require higher current (such as driving large DC motors) or the loads draw very high in-rush current when turned on and could dip the power supply to below 9V, then it is recommended that a separate power supply be used for the output load. Note that if you choose to use two separate power supplies, their 0V terminals should be tied together to provide a common ground reference for the two power supplies.

It is recommended that the PLC and Load power supply should be **either of the same output voltage** or the load voltage should be at a higher voltage that PLC power. If the load voltage is lower than the PLC, the output LED may light up even when the output driver is not energized. This is because the current could flow out of the Wx100 output terminal into the (lower voltage) load power supply, essentially applying a reverse voltage to the load (though it is current-limited by a 10K pull up resistor) and therefore light up the LED. This can be confusing to users even though it would not damage the output driver. Should this happen, you will need to add a series diode to the output terminal to block the output LED currents from flowing through the load and into to the load power.

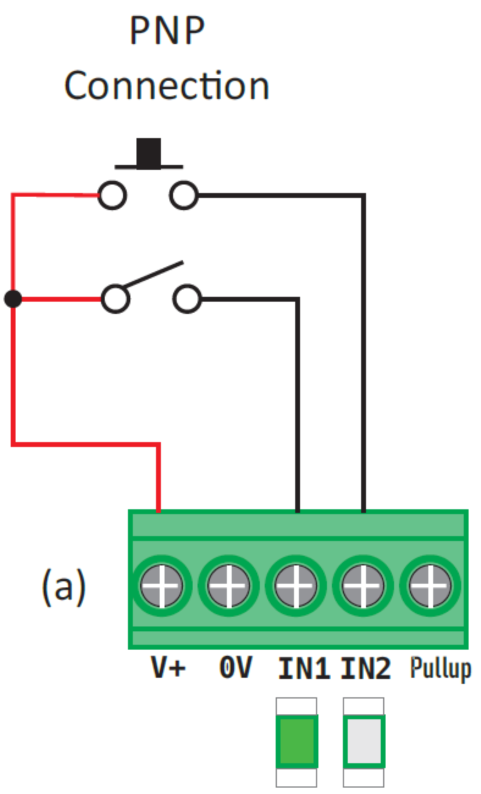
1.6 Digital Input Circuits

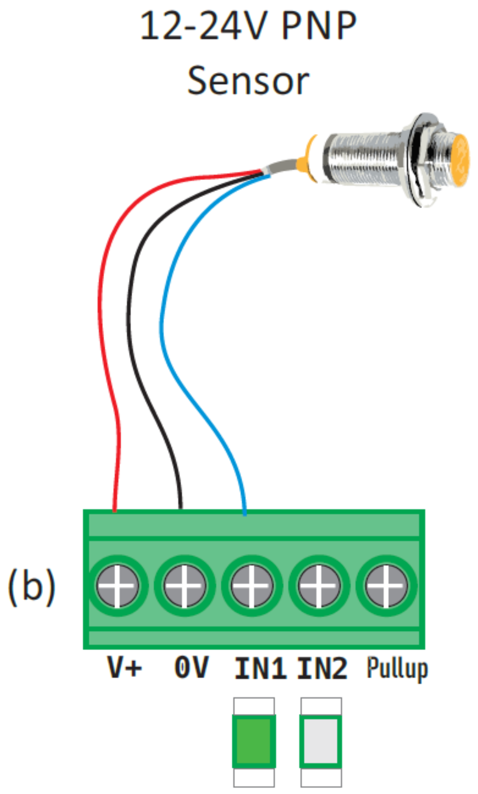
To simplify field wiring, the power input that Wx-TERM8 receives via its “POWER IN” connectors are routed by the Wx-TERM8 PCB to the input terminal section. The V+ and 0V terminals beside the IN terminals are electrically connected directly to the “POWER IN” terminals

a) PNP Connection

All Wx100 PLC digital Input are PNP (current sink) by default. This means that to turn ON the digital input you need to supply it a high positive voltage >=+8V. To turn off the digital input you need to supply it <= +3V.

Each digital input has a green LED indicator which lights up when the input is turned ON. Connecting to a mechanical switch or push button is extremely simple – it only needs to short the V+ to the IN terminal as shown in (a) below. PNP sensors that are powered from 12 to 24V DC is just as easily connected as shown as (b) in Figure 1.5





b) NPN Connection

NPN (current sink) sensors work in reverse to PNP sensors in that when it turns ON it will want to pull the input to low voltage, whereas when it turns 'OFF' it will either leave the input floating or pull it up to the power supply voltage level.

By connecting the "Pullup" terminal to V+, each digital input in the group (e.g. IN1 and IN2 are in one group, IN3 and IN4 are in another group) will be pulled up to V+ via its individual internal 3.3K 0.5W pullup resistor. This means that by default the digital input is turned ON (the green LED will light up) even if the sensor is not connected or sensor output an OFF state. When the NPN sensor turns ON it will pull the input to low, essentially turning OFF the PLC digital input. This means that the NPN sensor behaves in negative logic to the PLC.

Fortunately, you don't have to think in term of negative logic when you write the PLC program. By running the special **INVERT_INPUTS** command once during initialization of the PLC program, you can force the PLC firmware to invert the logic of specified inputs before processing. This includes the digital I/O scan as well as all interrupt-based input functions such as pulse monitoring and high-speed counter operation:

INVERT_INPUTS *ch*, *n*

ch	Every 16 digital inputs are grouped into 1 channel the same way as is defined by the system variable INPUT[ch]. Wx supports up to 5 digital inputs channel = 16 x 5 = 80 digital inputs.
n	bit within the channel to be inverted. Any bit that is a '1' in ***n*** will signal to the firmware that the corresponding physical input bit is to be inverted during I/O scan or when it is processed by an interrupt service routine.

E.g. If you are connecting only IN1 and IN2 to NPN sensors and the rest to PNP, that means you should set bit 0 and bit 1 to '1' and the rest to '0'. So you should run this command:

INVERT_INPUTS 1, &H0003 ' Only inputs 1 and 2 are inverted

If you are connecting first 8 inputs to NPN sensors, then run this command:

INVERT_INPUTS 1, &H00FF ' Inputs 1 to 8 are all inverted. Inputs 9 to 16 are not inverted.

Note:

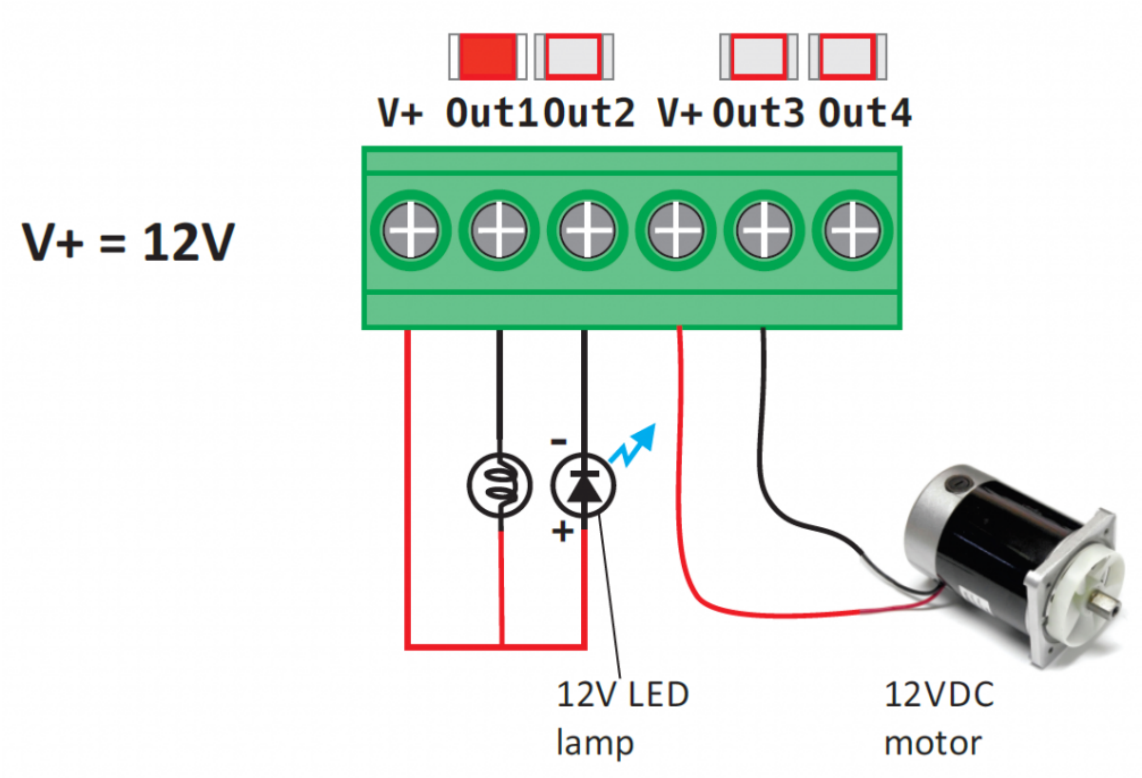
1. Run the **INVERT_INPUTS** *ch*, *n* command as early as possible in your PLC program. Ideally it should be the first statement inside an INIT custom function triggered by a **SCAN** contact at the first rung of the ladder logic.
2. An input that belongs to the input group where the "Pullup" signal is connected to V+ can no longer be configured as an analog input. Thus, if you have a system that needs to connect to analog, PNP and NPN sensors we recommend that you configure IN7 to IN16 (if available) as the NPN inputs since these inputs are not configurable to be analog inputs in the first place.
3. Although the ON/OFF input indicators on the i-TRiLOGI online monitoring screen will show correctly the input state of the input corresponding to its defined logic type (either positive or negative logic), the actual green LEDs of the digital inputs will continue to only light up in positive logic mode (i.e. when the input voltage is > 8V) even if an input has already been configured as an NPN input. This should be clearly stated in the service manual for your maintenance technicians to avoid confusion.

1.7 Digital Output Circuits

a) Wiring Digital Outputs

To simplify field wiring, the V+ power input that Wx-TERM8 receives via its "POWER IN" connector is routed by the Wx-TERM8 PCB to the output terminal section. This allows user to very easily wire to a typical output load. For example, the following shows how to easily use it to turn on a 12V lamp load. By configuring the digital output as a PWM output you can also control how much power to supply to the lamp and thus control the brightness in software.

The load can really be any kind of DC load as long as the working load voltage is compatible to that of V+ and the load current is < 2A continuous. It could be as small as a LED indicator lamp, a solenoid valve, the coil of a 12 or 24V relay or contactor, or a DC motor.



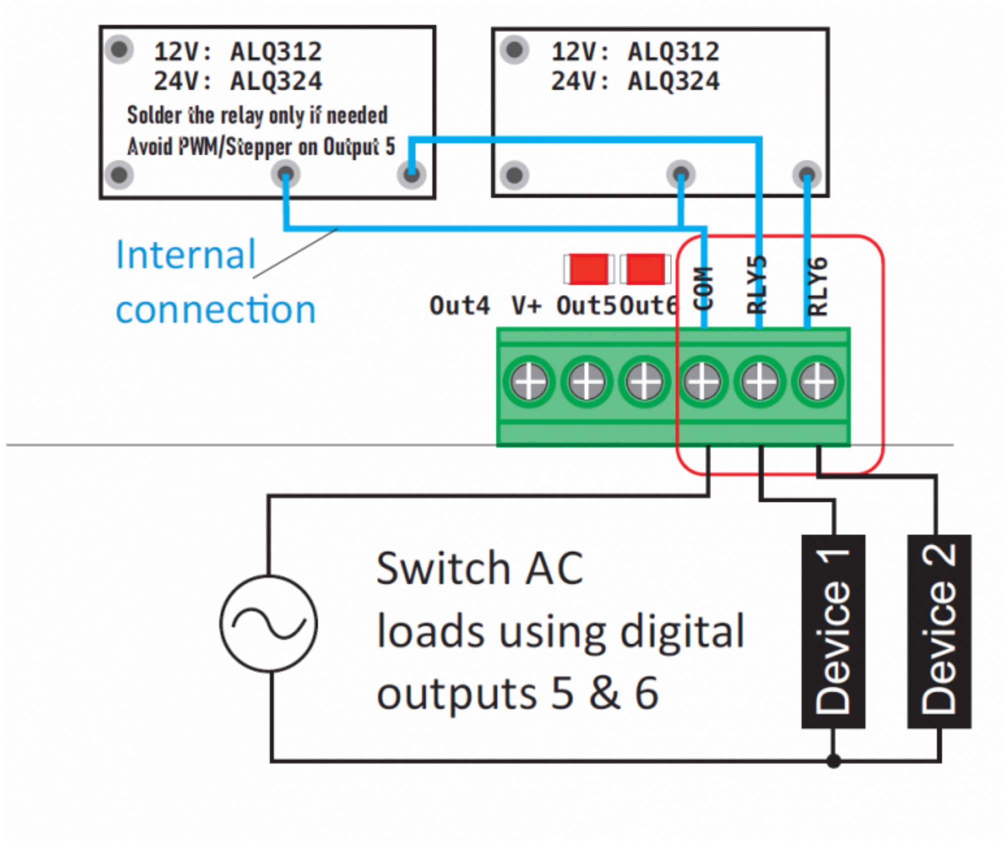
All digital outputs are directly programmable in Ladder Logic as well as in TBASIC custom functions. Some **programming examples** are detailed in “Chapter 3 –Digital I/Os and Internal Relays”

b) Relay Outputs

If you need to switch output loads of different voltages (including AC load) then you may need to use a relay to provide the galvanic isolation between the load voltage and the PLC power supply. To simplify using relay output, Wx-TERM8 provides solder pads for you to solder up to two Panasonic ALQ relays, as shown in the following diagram.

The contacts of these two relays are routed to the terminal marked “COM, RLY5 and RLY6” as shown in the figure to the left.

Note that soldering these two relays does not give you two additional outputs since the coils for these two relays are driven by output 5 and output 6. You must choose relays whose coil voltage match your power supply. For 12V system, solder a ALQ312 relay and for 24V system, solder a ALQ324 relay. You can purchase these relays from www.digikey.com (http://www.digikey.com) or any electronic components retailers.



Although the relays, the screw terminals and the PCBs are able to handle in rush current up to 10A, we strongly recommend keeping the steady state current carried by these relays to <= 5A each. If you need to switch high voltage, high current load, then you should consider installing an external contactor whose coil can be driven by either the DC output 5 or 6, or the AC power routed via RLY5 or RLY6 terminals.

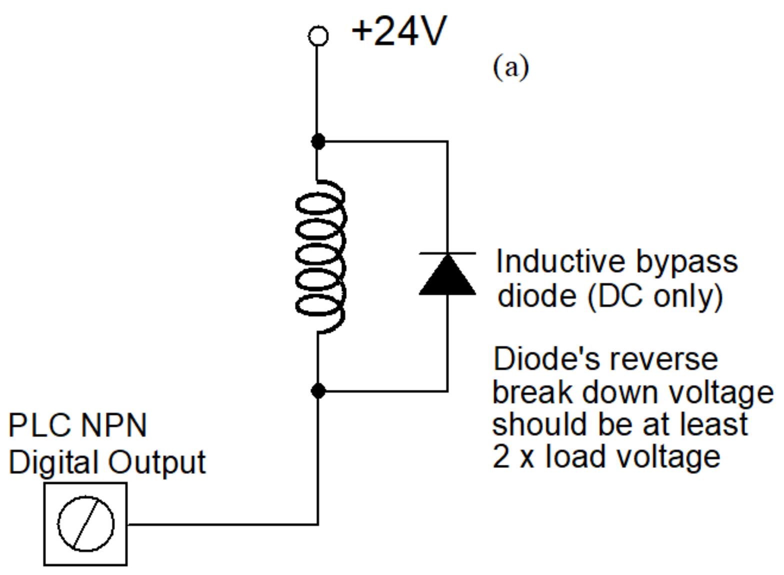
Note that Out5 & Out6 can still be used to drive DC load of the same power supply so you could potentially use them to switch 12 or 24VDC ON/OFF indicator lamps while at the same time the relay terminals are used to drive the AC load.

One reason we don’t install these two relays as standard is because once you install the relay you must avoid sending high speed switching current such as PWM or stepper motor controller/driver to the relay coils or it can cause the relay to chatter or buzz and can damage the relay contacts. Hence you should only install the relay(s) if you are really using these two outputs for low speed ON/OFF switching of external loads.

Since output 5 & 6 are fully capable of becoming PWM or stepper motor controller outputs, installing the output relay 5 or 6 will limit their flexibility and hence we elected to leave them as user-installable option.

c) Inductive Load

When switching inductive loads such as a solenoid or a motor, always ensure that a bypass diode is connected to absorb inductive kicks, which will occur whenever the output driver is turned OFF. Although all the PLC digital outputs already incorporate either internal diodes or intrinsic Zener bypass diodes to protect the driver, some may only activate when the inductive kick voltage rises above 100V DC. This can result in a large dose of noise being introduced into the system and may have undesirable effects. We recommend using a fast recovery diode such as UF4001 to UF4007 connected as shown in the following diagram to absorb the inductive noise:


+24V
(a)
Inductive bypass diode (DC only)
Diode's reverse break down voltage should be at least 2 x load voltage
PLC NPN Digital Output

d) Digital Outputs Specs

Output #1 to #6

Output Driver type	N-Channel power MOSFET with low r_{DS} < 0.05 Ohm
Maximum Breakdown Voltage	40V
Peak Output Current:	2A
Continuous Output Current	0.5A
Output Voltage when OFF	Pulled up to V+ via 3K3 resistor
Output Voltage when ON:	< 0.1V @2A

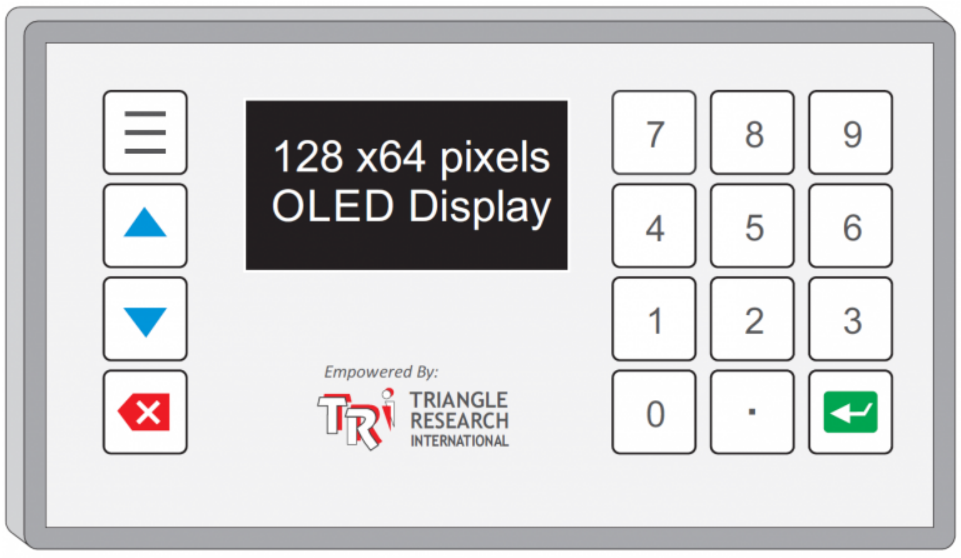
Inductive Back EMF Bypass	Yes (Intrinsic Zener)

All digital outputs have red LED indicators on the Wx-TERM8 and Wx-TERM16. The Wx100 PLC employs “sinking” (NPN) type power MOSFET outputs that turn ON by sinking current from the load to the 0V terminal.

Notes:

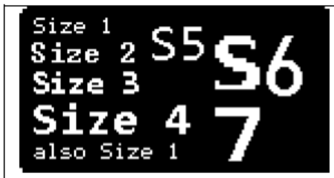
1. All 6 digital outputs can be configured as 6 PWM outputs using the SETPWM command (See Chapter 11)
2. Output 1 to 4 may be configured as a single full-step or half-step stepper motor driver, allowing it to directly drive a small stepper motor (See Section 10.2) .
3. Each pair of output 1&2, output 3&4 and output 5&6 can be configured as stepper motor controller to supply pulse and directional signals to up to 3 external stepper motor drivers. (See Section 10.3)

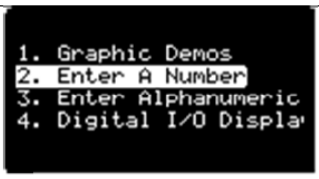
1.8 Keypad and OLED Display

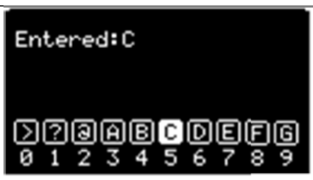


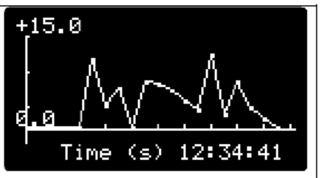
The most outstanding feature of the new Wx100 PLC is the integrated HMI that comprises a 16-key tactile keypad and a 128 x 64 pixels graphical OLED display. This provides users essentially with a zero cost HMI to interact with the PLC.

The built-in HMI is fully integrated into the new i-TRiLOGI version 7.4 software and with full simulation support on the i-TRiLOGI software itself. Many new keywords have been added to TBASIC to let you easily display a text string anywhere on the OLED display with up to 7 font sizes, or draw or fill basic shapes such as line, rectangle, triangle, circle. There is also a built-in **MENU_DEFINE** and **MENU_SHOW** commands that lets you very easily create multiple onscreen menus to help user to navigate the menu structure to control the equipment or view performance data, as shown in some sample screen shots below:



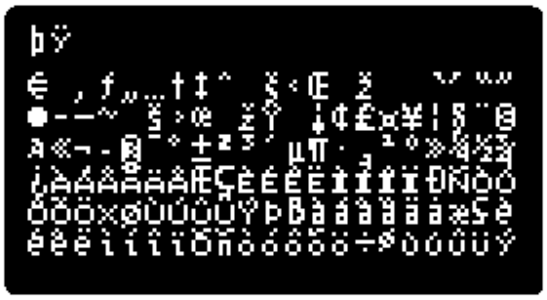




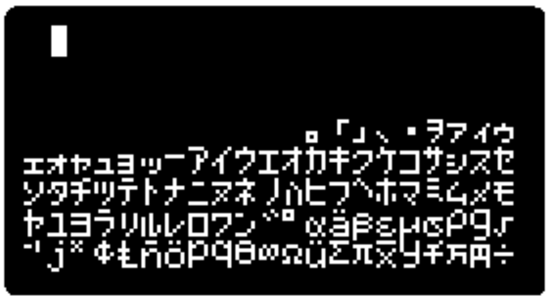


Extended ASCII Characters

TBASIC can display beyond the 32 to 127 basic ASCII characters to include extended ASCII character set. By default TBASIC displays extended ASCII based on Windows-1252 (https://www.ascii-code.com/) (code page 1252) character set, which includes many European characters. In addition, TBASIC also supports the Hitachi HD44780A00 character set (https://triplc.com/documents/LCDasiiTable.pdf) that includes a full set of Japanese Kana characters, by specifying text size 11 and display at 6 x 8 pixels only. See example below:



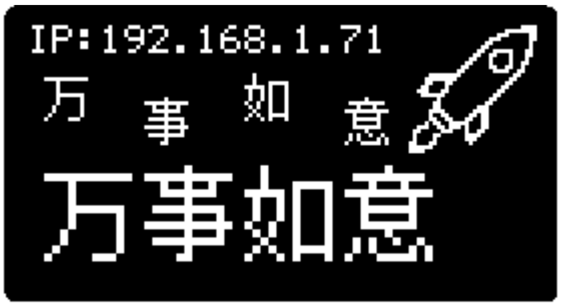
Extended ASCII – CP1252



Extended ASCII – HD44780A00

Custom Bitmap Icon

In addition, even though Wx100 does not directly support the thousands of characters available in all the world’s languages, it is still possible to create custom bitmap characters using the “DRAW_BITMAP” command, which allows TBASIC program to create and display custom bitmap graphics block of up to 32 x 32 pixels in size and support multiple magnification factor (1 to 4). This allows some limited display of messages in other languages or icons. Some examples are shown below:



Programming the HMI

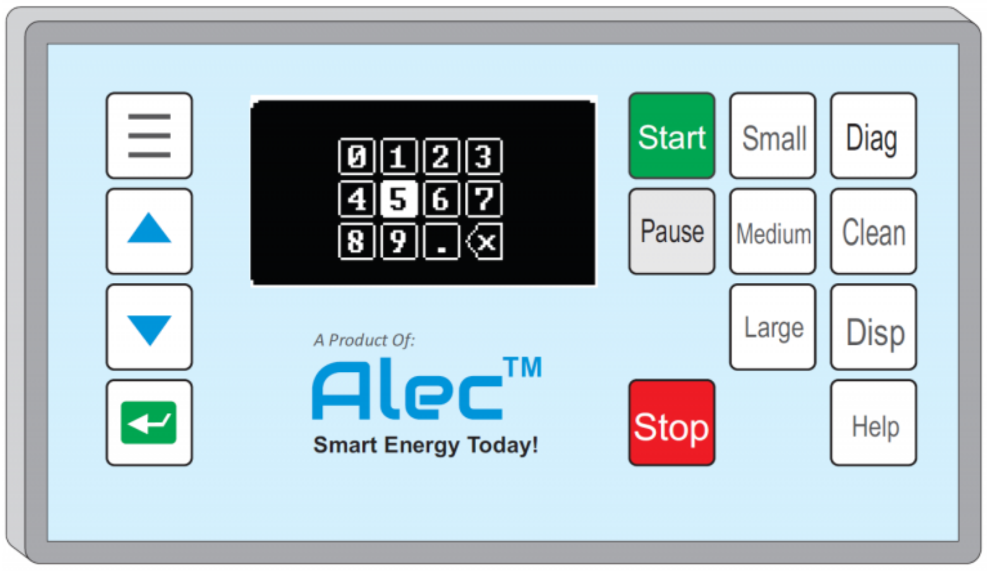
Some programming examples for the LCD display are presented in Chapter 13 (https://docs.triplc.com/#3639).

Customizing The HMI Keypad Legends

Although the 16 keys in the integrated keypad are already labelled as shown in Figure 1.7, they are not fixed to serve only as a numeric key, a menu key or backspace key, etc. The new TBASIC function “KEYPRESS(n)” actually receives a keycode from a keypress action and can use it for whatever purpose the program decides.

This means that **the 16 keys can be assigned to any purpose**. For example, if your application doesn’t need the user to enter numeric value frequently, TBASIC can display an on screen numeric or alphanumeric keypad if need be, thus still allowing you to occasionally enter data using only a few keys. The unused numeric keys can then be assigned as “Start” “Stop” “Jog” or whatever that you need for your application.

OEM with moderate to large volume can work with TRi to design a customized keypad that is specific to the equipment they build and inscribed with their own brand and logo instead of using the standard keypad legend. For example, it is conceivable to customize a keypad as follow:



The OEM will then be able to order the Wx100 PLC with their own branded, customized keypad to be used with their equipment.

1.9 Program and Data Memory

a) Program Memory

The Wx100 has 24K words (16-bit) of program memory stored in the CPU Flash memory area. Each ladder logic element (contacts or coils) takes up 1 word of memory. A TBASIC statement or function takes up half a word to four or five words, depending on the number of parameters the statement or function has.

The program memory can be erased and reprogrammed more than ten thousand times, which is a limit that you are unlikely to ever reach.

b) Non Volatile EEPROM Memory

Wx100 provides 14000 (16-bit) words of EEPROM memory that PLC program can use to store 8, 16 or 32 bit integer data or 32-bit floating point data. The EEPROM memory can be erased and write more than 1 million times.

c) Volatile Data Memory

All the TBASIC variables used in the Wx PLC: A to Z, A# to Z#, FP[1] to FP[1000], DM[1] to DM[4000] and string A$ to Z$, EMINT[1] to EMINT[16] and EMLINT[1] to EMLINT[16] are in the category of “volatile data memory” – meaning when you turn off power to the PLC the memory content will be lost and they will be reset to zero when the PLC is powered up again.

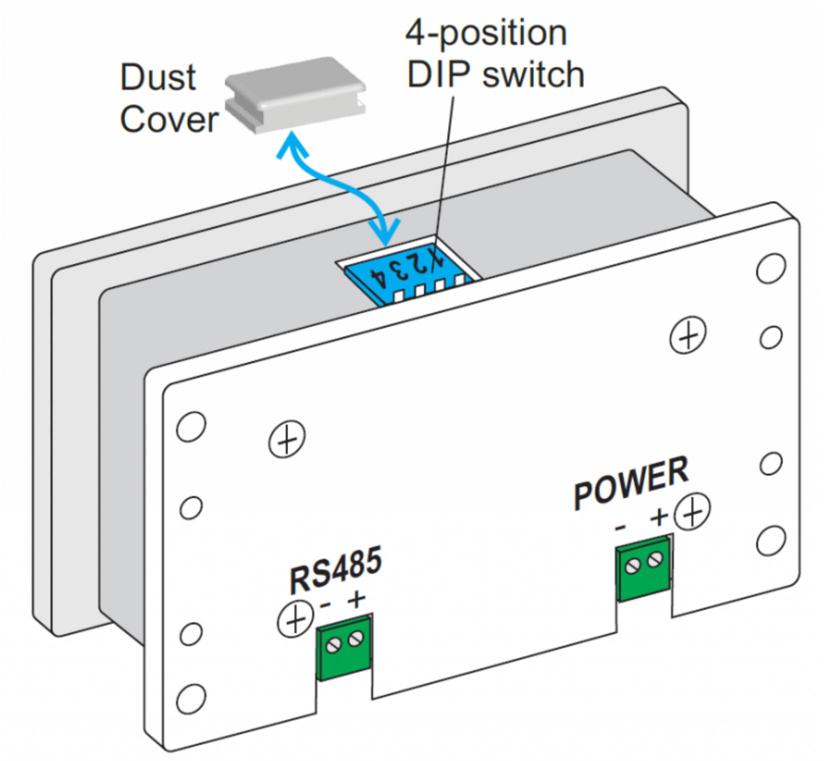
Unlike the Fx or FMD PLC, Wx PLC does not feature FRAM memory. This means that all TBASIC variables cannot be made non-volatile. However, you can still store the volatile memory data content inside EEPROM memory using the SAVE_EEP, SAVE_EEP# or SAVE_EEP$ before power shutdown and reload them back into volatile memory upon power up.

If you need volatile data to be automatically backup to EEPROM memory before power failure, then you need to do two things:

1. Build a voltage drop detector circuit and attach it to an interrupt input to detect imminent loss of power. The interrupt service routine must backup the crucial data before power drops below an acceptable level.
2. Use a large capacitor or a backup battery on the power circuit to the Wx100 PLC to allow voltage to decay gradually so that there is sufficient time for the CPU to save the data to EEPROM before power is completely lost.

1.10 DIP SWITCHES

There is a 4-position "piano" style DIP switch accessible from the upper side body of the Wx100 PLC. It may be covered by a silicone rubber dust cover. To access the DIP switch, simply remove the dust cover to expose the DIP switch and use a pen tip or a small screw driver to turn ON/OFF the switches.



DIP Switch	OFF	ON
SW1-1	–	–
SW1-2	–	–
SW1-3	Regular Wi-Fi networking station mode (if Wi-Fi is enabled)	Setup the Wx100 PLC to become an Access Point (AP) with an SSID. The PC can then connect to this AP to access the PLC and setup its WiFi parameters. This also disable the use of username/password and Trusted IP for FServer and Modbus/TCP Server.
SW1-4	Normal Run mode	Suspends execution of the ladder logic program. But host and Modbus TCP communication remains active.

Note: Wx PLC firmware now allows user program to read the positions of the DIP switches using the *n* = STATUS(23) function. DIP switch #1 ON/OFF status is represented by bit 0 in the returned data in *n*. DIP switch #2 by bit 1 and DIP switch #3 by bit 2. Since DIP switch #4 pauses the PLC, no command will run and thus its logic state will not be returned by the STATUS(23) function.

Usefulness of SW1-4

We have taken every effort to ensure that the host communication is always available even when the user-program ends up in a dead-loop. This allows the user to re-transfer a new program to the PLC and overwrite the bad program. However, you may still encounter a situation whereby after transferring a new program to the PLC, you keep encountering communication errors and you are unable to erase the bad program. This is especially common if you have been experimenting with the communication commands such as SETBAUD, SETPROTOCOL, PRINT or OUTCOMM. These commands may modify the communication baud rate, format, or protocol or set the PLC to send data out of a COMM port that conflicts with i-TRiLOGI. In such cases, you can turn ON DIP Switch SW1-4 and perform a power-on reset for the PLC. The PLC will not execute the bad program that causes communication problems and you can then transfer a new program into the PLC to clear up the problem.

Note that when the PLC has been power-reset with DIP Switch #4 set to ON, the serial port will boot up with default baud rate and communication format of 38,400, 8,n,1.

1.11 Real Time Clock

The Wx100 PLC has a built-in Real-Time clock (RTC) but does not have battery backup. The RTC will be reset to a factory date when first powered on. If the PLC is connected to the Wi-Fi network then it can automatically update its RTC after power on by making a TCP connection to an Internet time server to obtain Internet time (See Chapter 2.4.3 – Sync RTC to NTP Server (https://docs.triplc.com/#4740)). If the PLC is connected to the WiFi, but does not have access to the Internet , it can also obtain the time from a PC on the same LAN that provides the NTP time service, or a PC that runs the TLServer (part of the i-TRiLOGI suite) software.

Alternatively, if there is a PLC connected to the same LAN and has a battery-backed RTC (e.g. Fx2424), the Wx100 can connect to the Fx2424 to obtain its RTC data and update its own RTC.

Depending on demand, in future TRi may also supply a battery-back RTC module that the Wx100 can access to update its own RTC if the application does not enable the PLC to connect to the Wi-Fi network.

Chapter 2 - Wireless Networking

Overview

Every Wx100 PLC is integrated with Wi-Fi networking capability. This allows the Wx100 to connect to any 802.11 (a,g or n) network readily.

Once connected to the network, the TRiLOGI programming software and works instantaneously with the PLC, allowing remote programming and process monitoring over the air or via the Internet. In addition, the Ethernet facility at this port can host web pages and Java applets so that users of the equipment can control/monitor their equipment using their web browser from anywhere.

Like all its Super PLC cousins (Nano-10, FMD and Fx PLCs) Wx100 PLC supports Modbus TCP communication both as a client and a server, allowing it to easily perform machine-to-machine (M2M) communications with any other industrial equipment that support Modbus TCP protocols. Additionally, Wx100 can also be used as a Modbus to RS485 gateway, allowing it to become an intermediary between Modbus TCP capable software or equipment and older Modbus RTU over RS485 network.

2.1 Connect Wx100 to WiFi

A Wx100 PLC can work standalone with or without connection to the WiFi network. For example you can program the PLC either over the air via the WiFi network , or via the built-in RS485 port on the Wx100 with the help of a USB-RS485 serial port adapter connected to a PC.

However, even if the end application does not require network connection, you may still want to connect to it via WiFi network during program development since program transfer over the WiFi network is very much faster than over the slower RS485 serial connection.

You must setup two parameters before the Wx100 PLC can connect to your Wi-Fi network: i) SSID of your network and ii) The WiFi security key (WPA or WPA2). You can setup these two parameters using one of the following methods:

a) Setup the Wx100 PLC as a WiFi Access Point, then connect a PC or smartphone to it and use a web browser to access a webpage.

b) Use the onscreen keypad on the Wx100 OLED screen to enter the parameters.

c) Connect a PC to the Wx100 via RS485 serial port and run a configuration software.

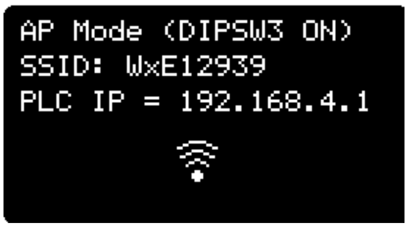
> Please note that both the SSID and the Passkey are CASE-SENSITIVE you must enter in exactly the same number of characters, without any leading or trailing white spaces.

We shall describe these 3 methods in the next few sections

a) Setup Using Wx100 AP Mode

1. Remove the DIP Switch dust cover (refer to section 1.10 (https://docs.triplc.com/wx100-um/#2586)) and turn ON DIP switch #3 on the Wx100 PLC.
2. Power cycle the Wx100 PLC.
3. Once power up with DIP switch #3 turned ON, the PLC will setup its own mini WiFi network and broadcast its own SSID in the format “WxXXXXXX”, where XXXXXX are 6 hexadecimal digits that represent the lower 24 bits of the Wx100 AP mode MACID. This SSID is shown on the OLED

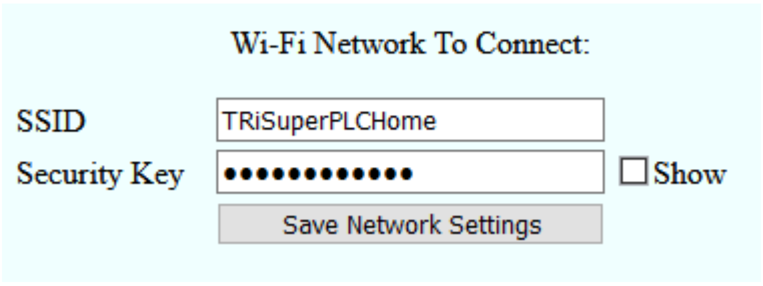
screen automatically. When powered on in AP mode the PLC will always be assigned a static IP address: 192.168.4.1, as shown below:


```
AP Mode (DIPSW3 ON)
SSID: WxE12939
PLC IP = 192.168.4.1
```

4. You can now use any PC or smartphone that has 2.4GHz WiFi (b, g or n) connection to connect to the SSID. First disconnect from the existing WiFi that the PC or phone is currently connected. Next search the available WiFi SSID when you attempt to make a WiFi connection and select the SSID that match what is shown on the OLED screen.

5. The security password to connect to the AP is "SuperPLC".

6. Once your PC or smartphone is connected to the AP (there will be no access to the internet when connected this way as it is just a local WiFi network connection), open a web browser and go the the follow web page: http://192.168.4.1:9080/netconfig.htm and the following webpage will appear

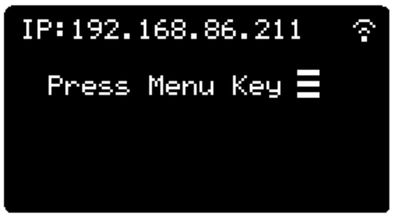

Wi-Fi Network To Connect:

SSID: TRiSuperPLCHome

Security Key: ●●●●●●●●●●●● ☐ Show

7. You can now enter the SSID and the security key of the **ACTUAL WiFi network** that you want the PLC to connect to. Then click the "Save Network Settings" button to save the settings.

8. Now turn off the DIP switch #3 and power cycle the PLC again. The PLC will now attempt to connect to the WiFi network with the SSID you entered above. If successful, the actual IP address of the PLC that is connected to the WiFi network will be displayed on the OLED screen, as follow:


```
IP:192.168.86.211
Press Menu Key
```

9. If the PLC failed to connect to the Wi-Fi network, please check the Wi-Fi credentials and then repeat procedure 1-8 and try again.

10. Note that once your PC is connected to the PLC SSID, you can also use the iTRiLOGI software to access the PLC using the IP address 192.168.4.1 and configure all the other network

parameters. However, we recommend that the AP mode be used only for changing the Wi-Fi SSID, security key and to enable/disable the WiFi.

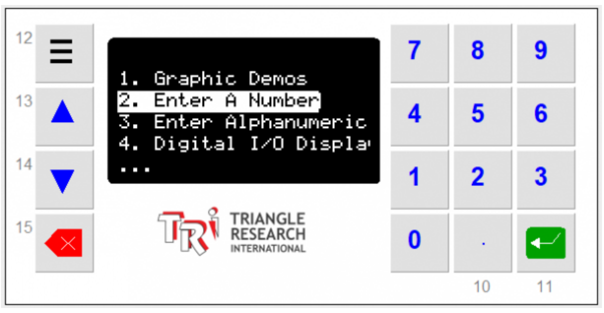
11. To setup all other network parameters, or to program the PLC, or accessing its file system via FTP server, please boot up the PLC in regular WiFi station mode (i.e. with DIP Switch #3 in OFF position when power ON). Ensure that the PLC is connected properly to the WiFi network and has a valid IP address. The PC can then perform all other network communication task with the PLC over the actual WiFi network.

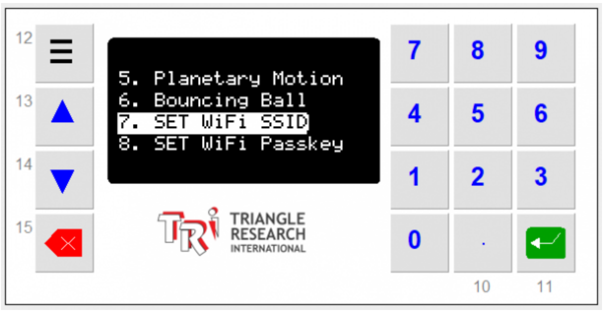
b) Setup Using Wx100's HMI

If you have received a new Wx100 that is loaded with the WxDEMO.PC7 program, then you can setup the WiFi parameters for the Wx100 PLC using its built-in keypad and OLED screen.

1. First, connect power to the PLC either via its dedicated power connector as described in Section 1.3 (https://docs.triplc.com/wx100-um/#2440): or via the Wx-TERM8 as described in Section 1.4 (https://docs.triplc.com/wx100-um/#2462) if the Wx100 PLC is already connected to the Wx-TERM8.

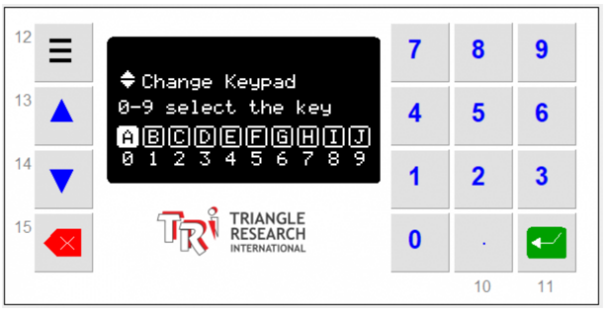
2. Once power on, the WxPLC will go through a logon procedure. You can then press the menu button (3-bar) on the keypad and a menu will appear on the OLED screen as follow:



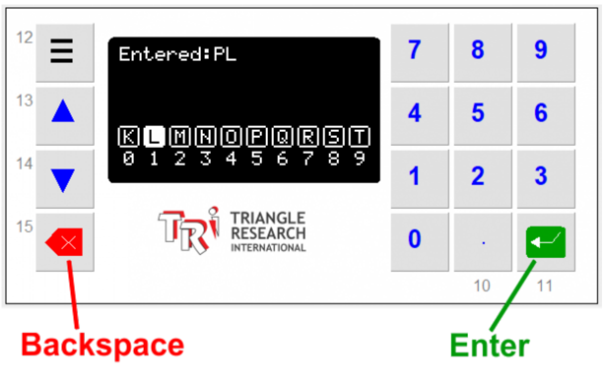


3. The menu spans two pages. You can select the menu item by pressing the up/down arrow key to move the highlight bar to the desired menu item, or simply by pressing the numeric key that corresponds to the menu item number. Item 7 & 8 in the default demo program allows you to activate an onscreen alphanumeric keyboard to set the two WiFi parameters as follow:

4. To set the WiFi SSID, select item “7. SET WiFi SSID” and press the Enter key. An on-screen alphanumeric keyboard will appear as shown below:



5. Due to small screen size only 10 keys are displayed at a time. By pressing the <up/down> arrow keys you can switch the keyboard to another set of alphanumeric characters. To select a character, press the numeric key number 0 to 9. For example if you press the “1” key on the following keypad the character “L” will be entered.



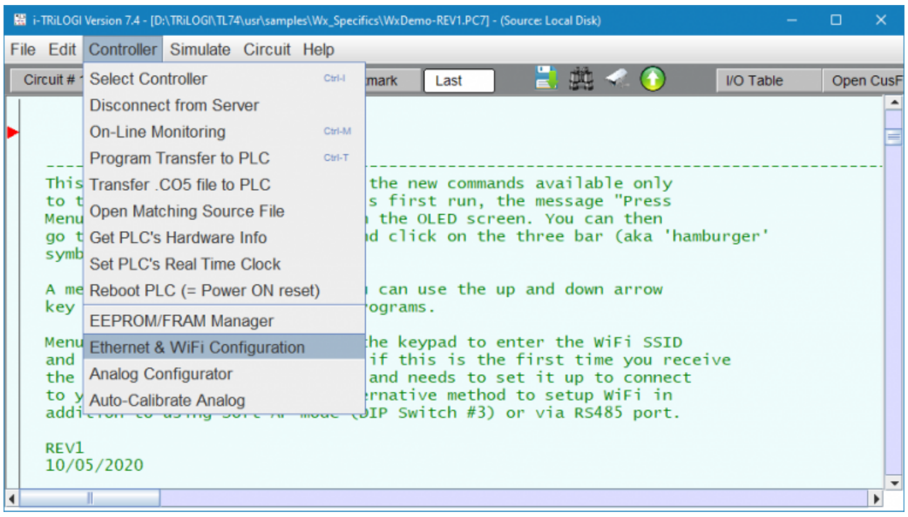
6. Enter one character at a time, all the characters (case sensitive) of the SSID of the WiFi network that the PLC is to be connected . If you have entered a wrong character, press the red “Backspace” key to delete the last entered character. When all the characters have been entered, press the green “Enter” key to save the new SSID to the PLC. You will then be offered the opportunity to also set the WiFi Key, to reboot the PLC, or simply quit the program. Note that the newly entered SSID and WiFi security key will only take effect after the PLC has been rebooted.

7. The WiFi Key can also be independently entered using menu item “8. SET WiFi Passkey”. Please follow the on screen instruction to enter the WiFi security key.

c) Setup WiFi Via RS485

Wx100 PLC’s built-in RS485 port can be used to program and configure all its parameter with the help of a USB-to-RS485 (e.g. U-485 (https://triplc.com/u485.htm)) adapter. This could be the quickest way to setup the Wi-Fi parameters for multiple PLCs. Using the i-TRiLOGI software you can also setup the Wi-Fi with either dynamic or static IP address. (If you are the end user and do not have the i-TRiLOGI software, please contact the equipment maker to obtain a license copy of configuration software if you wish to connect to the Wx100 PLC via RS485).

1. Connect the USB end of the U-485 to a PC’s USB port.
2. Connect the “+” and “-” terminal from the U-485 to the RS485 screw terminal on the Wx100 turn ON power to the PLC.
3. Run the “i-TRiLOGI” version 7.4, then select “Ethernet & WiFi Configuration” from the “Controller” menu.


i-TRiLOGI Version 7.4 - [D:\TRiLOGI\TL74\usr\samples\Wx_Specifics\WxDemo-REV1.PC7] - (Source: Local Disk)
File Edit Controller Simulate Circuit Help
Circuit #
mark
Last
I/O Table
Open CusF
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
Reboot PLC (= Power ON reset)
EEPROM/FRAM Manager
Ethernet & WiFi Configuration
Analog Configurator
Auto-Calibrate Analog

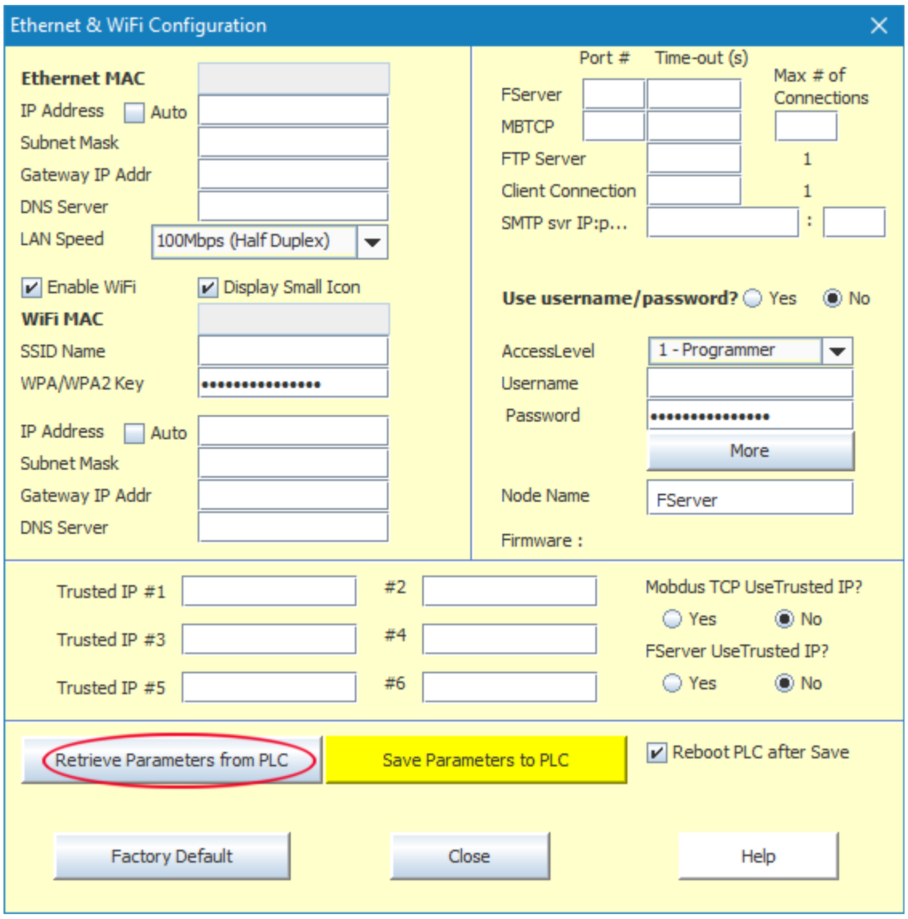
```
This                           the new commands available only
to t                           s first run, the message "Press
Menu                           the OLED screen. You can then
go t                           d click on the three bar (aka 'hamburger'
symb

A me                           can use the up and down arrow
key                            ograms.

Menu                           he keypad to enter the WiFi SSID
and                            if this is the first time you receive
the                            and needs to set it up to connect
to y                           rnative method to setup WiFi in
addi                          (DIP Switch #3) or via RS485 port.

REV1
10/05/2020
```

4. Next, click the “Retrieve Parameters from PLC” button



Ethernet MAC	
IP Address ☐ Auto	
Subnet Mask	
Gateway IP Addr	
DNS Server	
LAN Speed	100Mbps (Half Duplex)
☑ Enable WiFi	☑ Display Small Icon
WiFi MAC	
SSID Name	
WPA/WPA2 Key	•••••••••••••••
IP Address ☐ Auto	
Subnet Mask	
Gateway IP Addr	
DNS Server	

	Port #	Time-out (s)	Max # of Connections
FServer			
MBTCP			
FTP Server			1
Client Connection			1
SMTP svr IP:p...		:	

Use username/password? Yes [✓ No]

AccessLevel	1 - Programmer
Username	
Password	•••••••••••••••
Node Name	FServer
Firmware :	

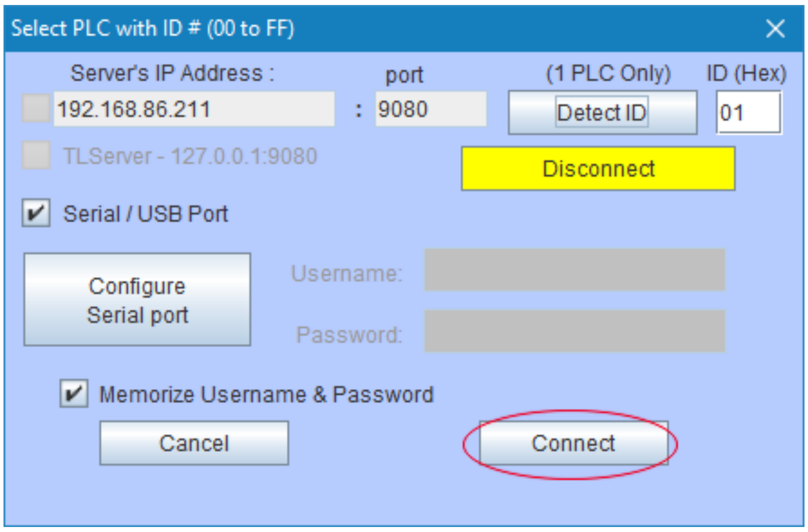
Trusted IP #1		#2	
Trusted IP #3		#4	
Trusted IP #5		#6	

Mobdus TCP UseTrusted IP? Yes [✓ No]

FServer UseTrusted IP? Yes [✓ No]

☑ Reboot PLC after Save

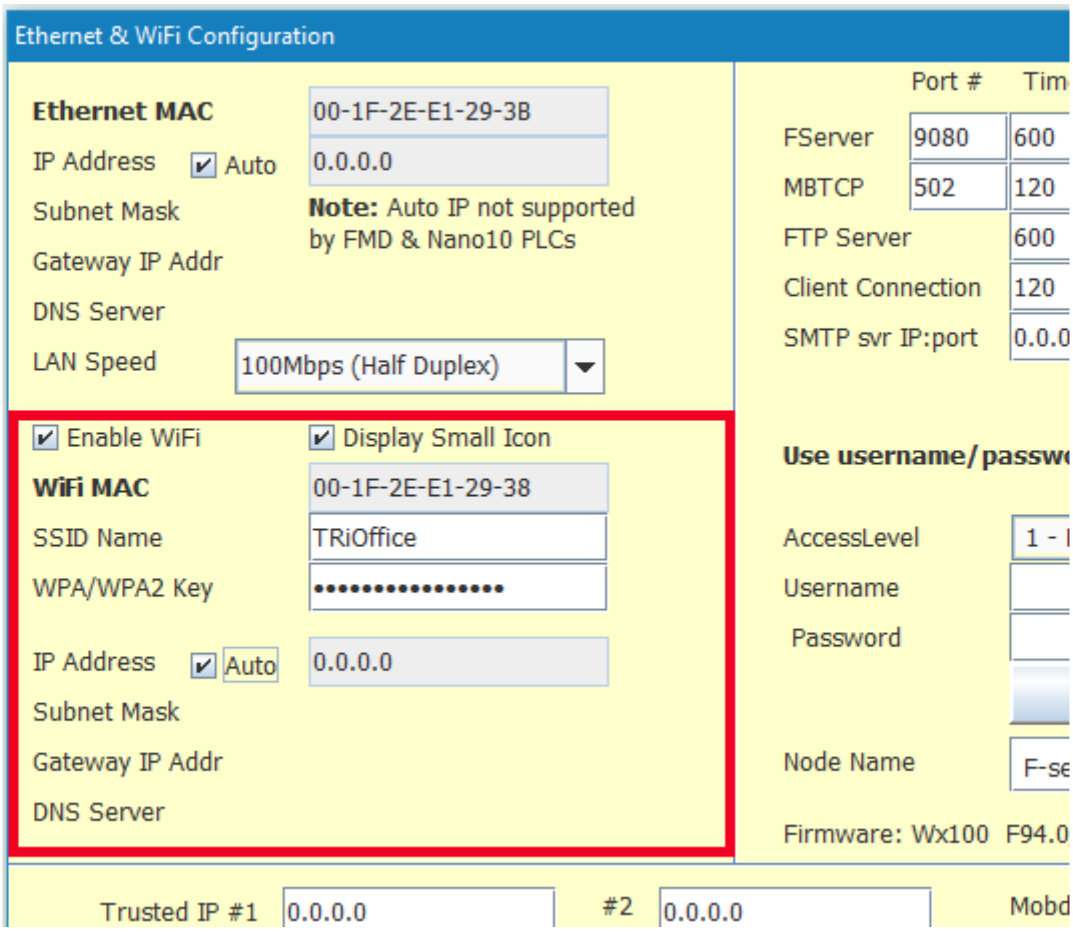
5. Since i-TRiLOGI is not yet connected to the PLC, a login screen will appear. Select the “Serial / USB Port” checkbox and click the “Detect ID” button. If the PLC returns its ID (default is 01) then the PC is correctly connected to the PLC via serial port and a yellow “Disconnect” button will appear as shown below.



Select PLC with ID # (00 to FF)

Server's IP Address :	192.168.86.211	
port	9080	
(1 PLC Only)	Detect ID	
ID (Hex)	01	
TLServer - 127.0.0.1:9080		
[✓] Serial / USB Port		
Username:		
Password:		
[✓] Memorize Username & Password		

6. Next, click the “Connect” button to complete the login and the previously stored configuration will be retrieved from the PLC. The Wi-Fi settings are shown in the highlighted section below:



Ethernet & WiFi Configuration

Ethernet MAC	00-1F-2E-E1-29-3B
IP Address [✓] Auto	0.0.0.0
Subnet Mask	**Note:** Auto IP not supported by FMD & Nano10 PLCs
Gateway IP Addr	
DNS Server	
LAN Speed	100Mbps (Half Duplex)
[✓] Enable WiFi	[✓] Display Small Icon
WiFi MAC	00-1F-2E-E1-29-38
SSID Name	TRiOffice
WPA/WPA2 Key	••••••••••••••••
IP Address [✓] Auto	0.0.0.0
Subnet Mask	
Gateway IP Addr	
DNS Server	

	Port #	Tim
FServer	9080	600
MBTCP	502	120
FTP Server		600
Client Connection		120
SMTP svr IP:port		0.0.0

Use username/passw

AccessLevel	1 - I
Username	
Password	
Node Name	F-se

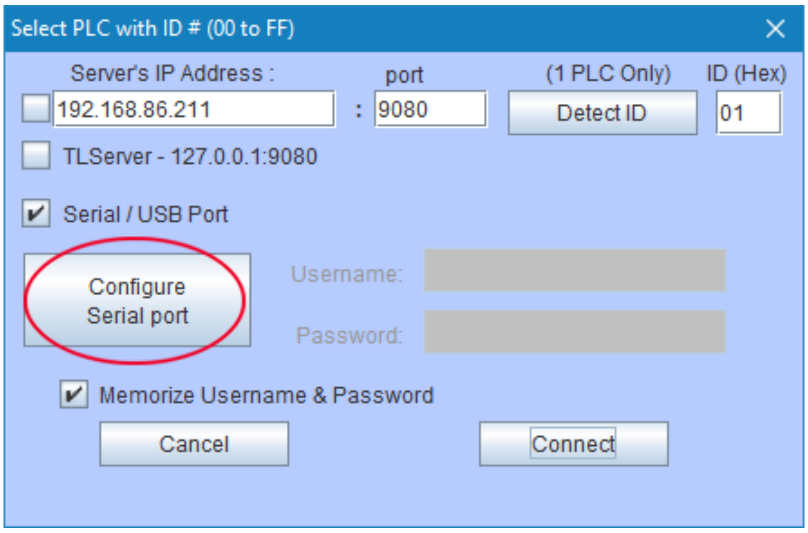
Firmware: Wx100 F94.0

Trusted IP #1 0.0.0.0 #2 0.0.0.0 Mobd

7. Now enter the SSID Name and the Wi-Fi security key (WPA/WPA2 Key) in the appropriate text fields and click the “Save Parameters to PLC” button to write the WiFi parameters to the PLC via the RS484 connection. The PLC will reboot immediately and you should then check on the OLED if it has successfully connected to the WiFi network.

Trouble Shootings RS485 Serial Port Connection

Usually if the USB-RS485 adapter is the only serial port device you connect to the PLC, the iTRiLOGI will recognize it and use it automatically so the connection to the serial port would be quite seamless. However, it is possible that your PC may have multiple serial port devices connected and thus iTRILOGI may not know which serial port is the one that is connected to the PLC. Hence if you receive an error message when you click the "Detect ID" button, then you can click the "Configure Serial Port" button to configure serial parameters as shown below:


Select PLC with ID # (00 to FF)

Server's IP Address : 192.168.86.211 : port 9080

(1 PLC Only) Detect ID ID (Hex) 01

TLServer - 127.0.0.1:9080

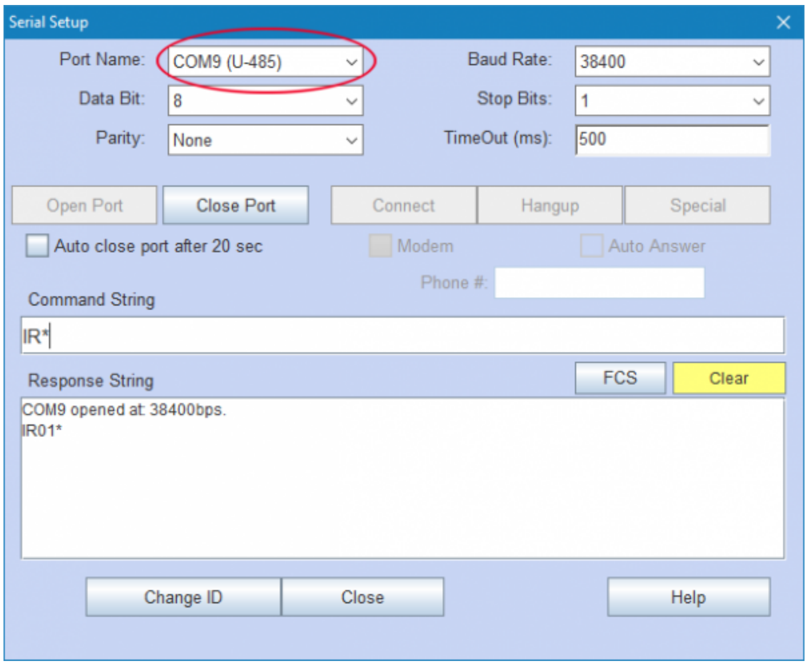
[✓] Serial / USB Port

Username:

Password:

[✓] Memorize Username & Password

1. Make sure that the COM port number select in the "Port Name" choice box is indeed the COM port number of the USB-RS485 adapter. If you have purchased the U-485 adapter supplied by TRi then the correct COM port number assigned to the U-485 will have the string "(U-485)" appended to it, making it easier to identify it. For other brands of USB-RS485 adapter, find out which COM port number Windows has assigned to it by checking the device manager in the Windows Control Panel)


Serial Setup

Port Name: COM9 (U-485)

Baud Rate: 38400

Data Bit: 8

Stop Bits: 1

Parity: None

TimeOut (ms): 500

Auto close port after 20 sec

Command String: IR*

Response String:

```
COM9 opened at: 38400bps.
IR01*
```

2. If you find multiple COM ports listed in the drop box then you have to find out which COM port number belongs to the USB-RS485 adapter and select the correct COM port. To test if

the correct COM port is select, type in the string “IR*” in the “Command String” text box and press Enter Key. If the PLC responds with “IRxx*” it means you have correctly connected to the PLC. However, if it replies with “Unable to Open COM port xx” it could mean either the COM port is wrong, or is currently used by another program. Try to close any other program that could be using the USB-RS485 adapter COM port and try again.

3. If the COM port is opened and you are sure that it is the correct COM port for the USB-RS485 adapter, yet you receive a “No Response From PLC” error, then it could be that the serial port settings on the PC does not match that in the PLC. The default PLC RS485 settings is 38400 bps, 8 data bit, 1 stop bit and no parity. If the PLC you are accessing is already pre-programmed by someone else, these default settings may be changed by the program and thus you will not be able to access the PLC using the default COM settings. If you suspect this is the case, turn on DIP Switch #4 in the WxPLC and power cycle the PLC. That will cause the PLC to PAUSE immediately upon power ON and thus will not run any user program that could modify the PLC’s serial port settings. Then try the above procedure again to make a connection to the PLC.
4. If all else fails, then you may either have a broken USB-RS485 adapter, or the PLC’s RS485 port has malfunctioned. Try a different USB-RS485 adapter or a new Wx100 PLC to see if you are able to resolve the problem. If the problem lies with the Wx100 PLC’s RS485 port then you may need to request an RMA to send in the PLC for repair.

d) Setup With Known SSID

Every Wx100 PLC is shipped with default WiFi parameters that were used during factory testing.

Hence, if you are a regular user of the Wx100 PLC then you could setup a WiFi router in your facility configured with the factory-default SSID and passkey.

Once you have received a new Wx100 PLC from the warehouse, it should immediately connect to the WiFi router upon power up and ready to use. You only need to connect your PC to this WiFi router and you will then immediately be able to start programming the PLC using the i-TRiLOGI software.

Note:

1. Please contact support@triplc.com to request for the passkey of the default SSID. Please supply your credential and evidence of purchase (invoice, receipt etc) of our Wx100 PLC.

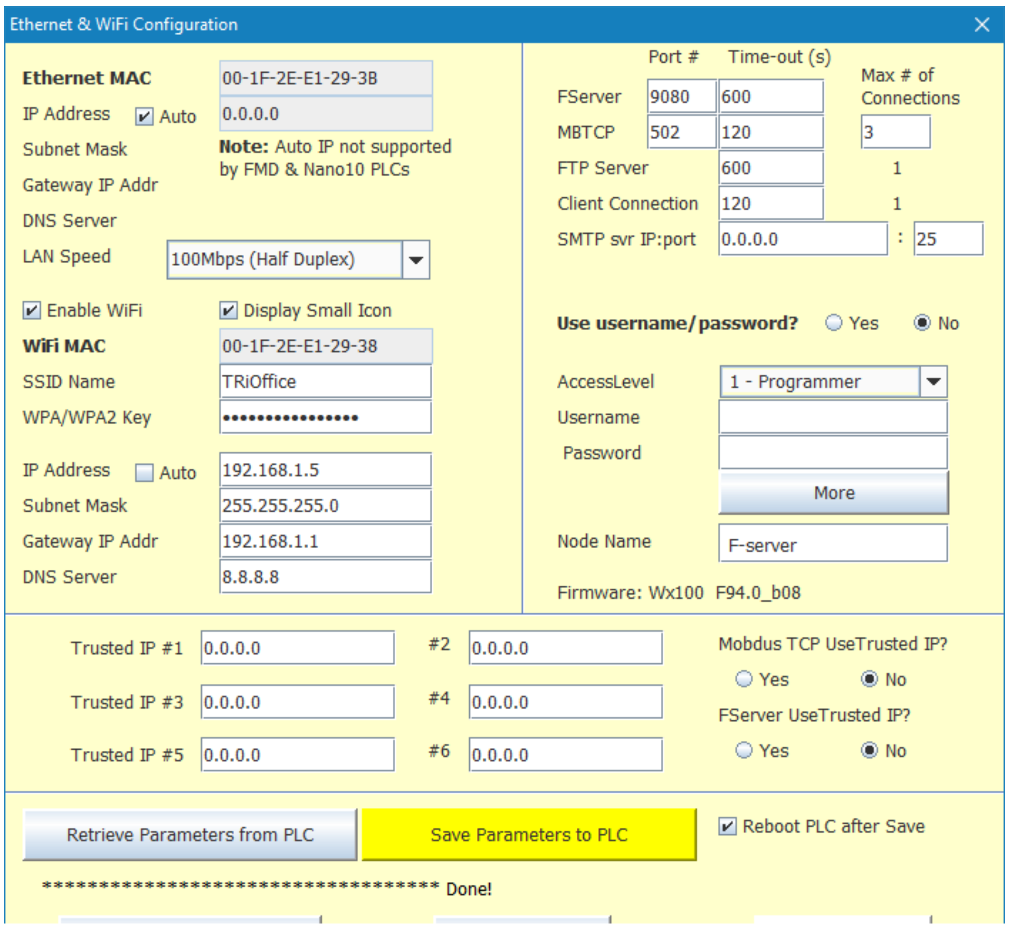
2. If a new Wx100 PLC is unable to connect to the network that you have setup with the default SSID and passkey, then please try one of the other 3 methods to gain connection to the PLC and retrieve the WiFI settings to confirm if it has been changed.

2.2 Configure Network Settings

If you are the end user of a machine or equipment you do not need to proceed further beyond connecting your PLC to your WiFi network. The PLC is fully operational once it is connected to the WiFi network and will run the last operational program transferred to the PLC. The next few sections describe all other PLC network settings that would be set mainly by the PLC programmers or the OEM equipment makers if the default factory settings is not suitable for their applications.

You will need the iTRiLOGI Version 7.4 or later in order to configure the Wx100 PLC's network settings. We have briefly touched on this topic in Section 2.1(c) when we describe how you can configure the PLC's WiFi SSID and security key via the RS485 port. In this section we will describe all the other network related settings such as IP address, gateway, web server and Modbus TCP servers, etc.

First, run the iTRiLOGI version 7.4 or later software. Click the "Controller" menu and select "Ethernet & WiFi Configuration". When the configuration screen appears, click the "Retrieve Parameters From PLC" button, then login to the PLC using the IP address shown on its OLED screen to retrieve a copy of all the previous settings saved into the PLC, as follow:


Ethernet & WiFi Configuration

Ethernet MAC 00-1F-2E-E1-29-3B
IP Address ✓ Auto 0.0.0.0
Subnet Mask
Gateway IP Addr
DNS Server
Note: Auto IP not supported by FMD & Nano10 PLCs
LAN Speed 100Mbps (Half Duplex)

✓ Enable WiFi
✓ Display Small Icon
WiFi MAC 00-1F-2E-E1-29-38
SSID Name TRiOffice
WPA/WPA2 Key ••••••••••••••••
IP Address ☐ Auto 192.168.1.5
Subnet Mask 255.255.255.0
Gateway IP Addr 192.168.1.1
DNS Server 8.8.8.8

	Port #	Time-out (s)	Max # of Connections
FServer	9080	600	
MBTCP	502	120	3
FTP Server		600	1
Client Connection		120	1
SMTP svr IP:port	0.0.0.0	: 25	

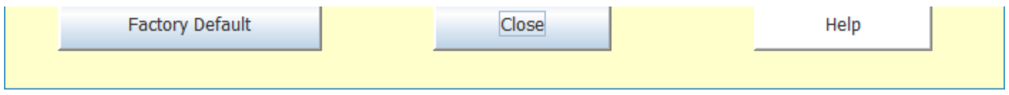
Use username/password? Yes [✓ No]
AccessLevel 1 - Programmer
Username
Password
More
Node Name F-server
Firmware: Wx100 F94.0_b08

Trusted IP #1	0.0.0.0	#2	0.0.0.0
Trusted IP #3	0.0.0.0	#4	0.0.0.0
Trusted IP #5	0.0.0.0	#6	0.0.0.0

Mobdus TCP UseTrusted IP? Yes [✓ No]
FServer UseTrusted IP? Yes [✓ No]

Retrieve Parameters from PLC
Save Parameters to PLC
✓ Reboot PLC after Save

************************************* Done!



After you have retrieved the existing parameters from the PLC, you will see that various fields in the configuration software screen are filled up. Note that the Wx PLC has two built-in “Server” programs that listen on a few different ports on the PLC for incoming TCP/IP request packets:

1) The FServer supports the TRi proprietary programs such as the i-TRiLOGI software and TRi-ExcelLink program, and also provide a web server for web page request from a browser software and implement simple web page applications. The FServer listens on the default port 9080.

2) A MODBUS/TCP server that listens on port #502 and supports the industry standard MODBUS/TCP protocols.

Note:

- These two servers share the same WiFi connection and therefore the same IP address and gateway addresses described in the following sections.
- The Wx100 PLC host both the FServer and Modbus/TCP Server, each can serve multiple simultaneous connections from external clients. This means that it is possible to connect multiple TRiLOGI, ExcelLink and Modbus/TCP clients to the PLC, all at the same time! The maximum number of available simultaneous connections will depends on the memory available in Wx100 and usually is less than 10 in total.

2.2.1 Disable / Enable WiFi

Wx100 WiFi is enabled by default so that it can make use of all the wonderful power of network connection. However, for whatever reason if your application demands that the PLC WiFi must be disabled, you can disable it by checking OFF the “Enable WiFi” check box in the “Ethernet & WiFi Configuration” screen before you deploy the equipment. In this case the PLC will disable its WiFi radio and all its network capability after reboot. Obviously once you have disabled the WiFi you can no longer access the PLC over the air anymore, and hence to re-enable the WiFi you need to connect the PC to the PLC via the RS485 port and run this iTRiLOGI command again. Please refer to Section 2.1(c) (https://docs.triplc.com/wx100-um/#2774) for more details).

> **Note**: Regardless of “Enable WiFi” setting, the WiFi will not be disabled if the PLC is rebooted with DIP Switch #3 turned ON. This is to provide a life-line for you to recover the settings by using the PLC’s Access Point setup mode. Please refer to Section 2.1(a) (https://docs.triplc.com/wx100-um/#2705) for more details.

If you are accessing the PLC remotely via the Internet, please make sure that you understand the implication of disabling the WiFi before doing so since you will lose connection to the PLC after you click the “Save Parameters To PLC” button. The only way to recover to to have physical access to the PLC and connect to it via RS485, or via Access Point mode.

2.2.2 Static or Dynamic IP Address

The factory default IP address in the PLC set to "Auto".

With the "Auto" checkbox selected, the PLC will be assigned a dynamic IP address by the network it is connected to. What it means is that if the PLC is turned OFF for some time (say for more than a day), the next time the PLC is powered ON it may get a different IP address from previous session. The assigned IP address will be displayed on the OLED screen when it is powered ON so the programmer can note down the IP address and use it in iTRiLOGI software to program the PLC over the air.

However, if the PLC needs to be accessed by another PLC or by other equipment (such as in M2M applications) then the other equipment must know or somehow able to find out the PLC's IP address without being able to look at the OLED screen. In this case, it is preferable that the PLC is assigned a known (static) IP address. There are two ways to achieve this as described below.

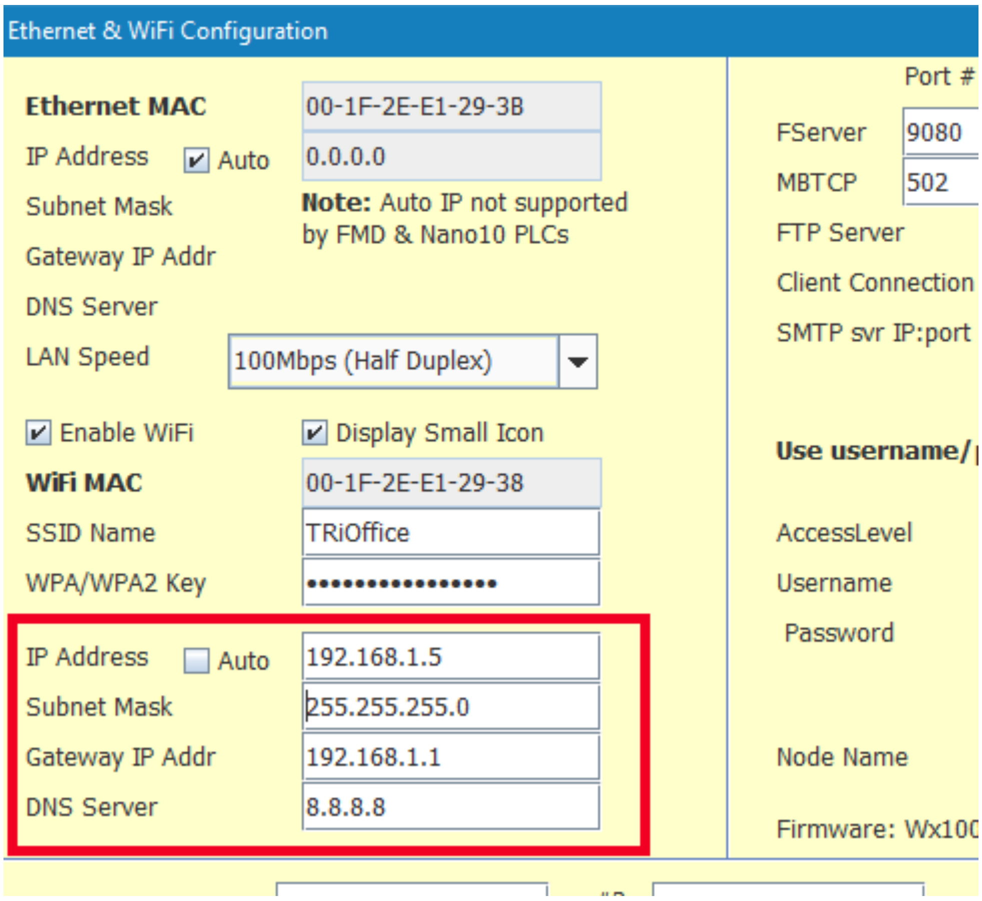
a) Setup DHCP Server To Reserve IP Address For Wx100 PLC

One way of assigning a static IP address to the PLC is to do it at the DHCP server (for small home WiFi network, the DHCP server is embedded within the router and accessible via the WiFi router's setup interface). Most DHCP servers would let you perform "IP Address Reservation" which means that you can reserve a particular IP address to a particular device connected to the network. The device is identified by its MAC ID and each Wx100 PLC will be assigned a unique MAC ID that you can read from the Ethernet & WiFi Configuration screen. For more information regarding IP address reservation, please consult the IT specialist in your organization or the user manual of your WiFi router. If you elect to use reserved IP address by the DHCP server, you should continue to check the "Auto" checkbox so that the reserved IP address will be assigned by the router and you do not need to go any further.

b) Assign Static IP address to Wx100 PLC via i-TRiLOGI Software

On the other hand, if you have some basic understanding of your network settings, such as the subnet mask, gateway IP address and DNS server, you can assign the PLC with a "static IP address". Most importantly you must make sure that the static IP address you want to assign to your Wx100 PLC is not in the range of dynamic IP addresses that the network DHCP server could assign to other devices attached to the network. This is to ensure that there is no duplicate IP addresses (i.e. two separate devices having the same IP address, which cause collisions) that can lead to a lot of communication trouble for your network.

If you check OFF the "Auto" check box next to the "IP Address" text field, you will be presented with a list of text fields allowing you to assign the static IP address and other necessary parameters to support the static IP address as shown below:



Ethernet & WiFi Configuration

Ethernet MAC	00-1F-2E-E1-29-3B		Port #
IP Address ☑ Auto	0.0.0.0	FServer	9080
Subnet Mask	**Note:** Auto IP not supported by FMD & Nano10 PLCs	MBTCP	502
Gateway IP Addr		FTP Server	
DNS Server		Client Connection	
LAN Speed	100Mbps (Half Duplex)	SMTP svr IP:port	
☑ Enable WiFi	☑ Display Small Icon	**Use username/**	
WiFi MAC	00-1F-2E-E1-29-38		
SSID Name	TRiOffice	AccessLevel	
WPA/WPA2 Key	••••••••••••••••	Username	
IP Address ☐ Auto	192.168.1.5	Password	
Subnet Mask	255.255.255.0		
Gateway IP Addr	192.168.1.1	Node Name	
DNS Server	8.8.8.8	Firmware: Wx10	

1. Subnet mask: For small network such as a home WiFi router, the subnet mask is usually 255.255.255.0

2. The Gateway IP address is usually the private IP address of the router itself (the same IP address that you use to setup the router via the web browser). The gateway is the network service that lets the Wx100 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 let Wx100 connect to the internet (to synchronize its real-time-clock, upload data to cloud server or send an email) then you must set the correct Gateway IP Address. For bigger network please consult your IT to obtain the gateway IP address and also to get authorization to allow your PLC to open a port and connect to the Internet.

3. The DNS (Domain Name Server) allows the Wx100 PLC to contact a remote server by means of domain name instead of IP Address. The DNS takes in the given domain name (such as triplc.com) and returns the IP address of the target server. You will need to fill in the DNS IP Address if you intend to program the PLC to connect to external web server on the Internet by using its domain name instead of IP address. The DNS server IP address could be the same as

the Gateway IP Addr described above. But it will be more efficient to enter the actual primary DNS server IP address that your ISP provides. You can also use the generic 8.8.8.8 and 4.4.4.4 (which is the universal DNS server IP addresses courtesy of Google).

Please enter all the parameters and click “Save Parameters To PLC” to complete the setup. After the PLC is rebooted, please check the PLC’s OLED screen display to ensure that the PLC can connect to the network and obtain the desired static IP address.

2.2.3 Server Port Number & Time-out Settings

a) FServer Port Number

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 or Modbus TCP server from across the network.

The default value for FServer is 9080, which is the same default value used by the TRiLOGI client software. You can change it to any port number between 1024 and 65535 except the Modbus TCP port number that is specified for the PLC (default is 502).

Please see the 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.

b) Modbus TCP Port Number

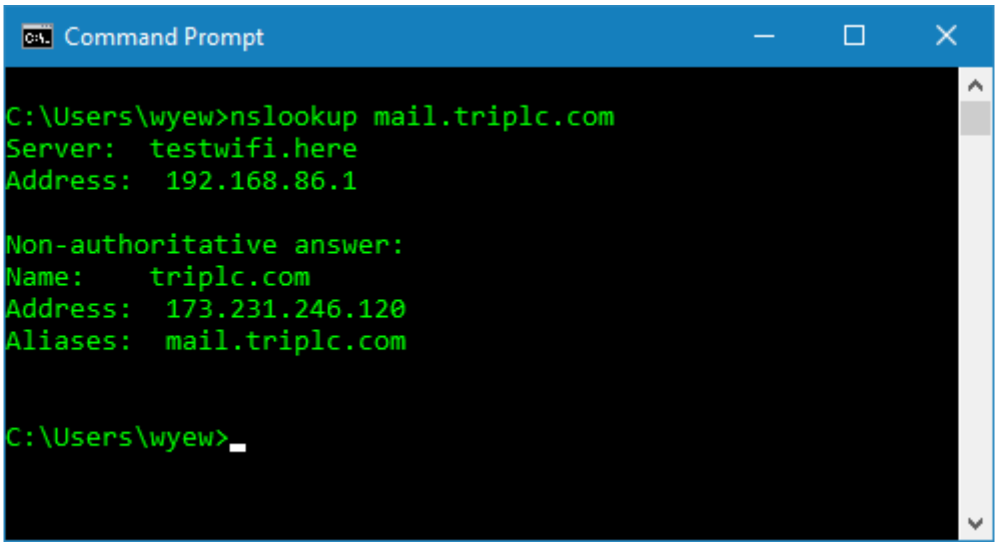
According to MODBUS.ORG specifications, all Modbus/TCP servers must listen on port #502. However, Wx100 PLC allows you to specify a different port number in case you need to use port forwarding from the router to more than one PLC.

You may specify any port number between 1024 and 65535 (except for the port number already used by the FServer).

c) SMTP Server IP Address

The SMTP (Simple Mail Transport Protocol) Server field lets you define the IP address of the email server that the PLC can use to send out emails from user's program (please see section 2.3.3 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 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":



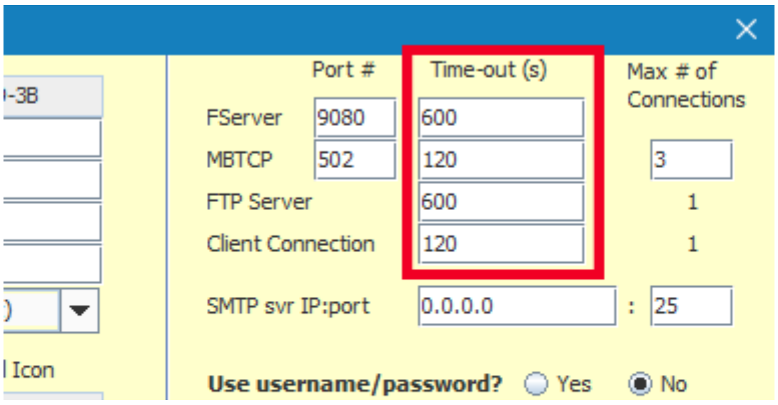
```
C:\Users\wyew>nslookup mail.triplc.com
Server:  testwifi.here
Address:  192.168.86.1

Non-authoritative answer:
Name:    triplc.com
Address:  173.231.246.120
Aliases:  mail.triplc.com


C:\Users\wyew>_
```

d) Time-out Settings

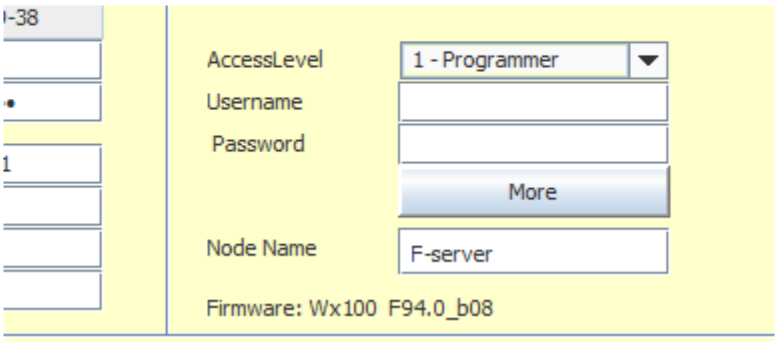
FServer and Modbus TCP Server in Wx100 PLC support only a limited number of connections (due to memory limit). In fact the FTP server in the PLC supports only as single connection. Once the maximum number of connections have been exhausted, no more clients can connect to the PLC anymore. By setting a time-out for these connection, Wx100 can unilaterally terminate the client connection if the client has been idle for a period exceeding the specified time-out. You can enter the time-out values (number of seconds) in the specified fields:



	Port #	Time-out (s)	Max # of Connections
FServer	9080	600	
MBTCP	502	120	3
FTP Server		600	1
Client Connection		120	1

SMTP svr IP:port 0.0.0.0 : 25

Use username/password? Yes [✓ No]



2.2.4 Other Network Settings

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

This field is not applicable to Wx100 PLC. It is used by the Nano-10, FMD and Fx PLCs to split the total number of client connects between Modbus and FServer.

b) Ethernet Settings and LAN Speed

This section is not applicable to Wx100 PLC. These fields are used by other PLC families that have built-in Ethernet port.

c) Username and Password (FServer only)

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 TRiLOGI software to encrypt the password transmission. FServer only permits a single set of username/password and this is limited to a length of 16 characters. (Although you may define the passwords for a USER and a VISITOR privilege).

d) 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 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. You can also access any web page hosted on the FServer without any login credentials.

e) Access Level

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 i-TRiLOGI Programmer's Reference manual for the definition of the access levels.

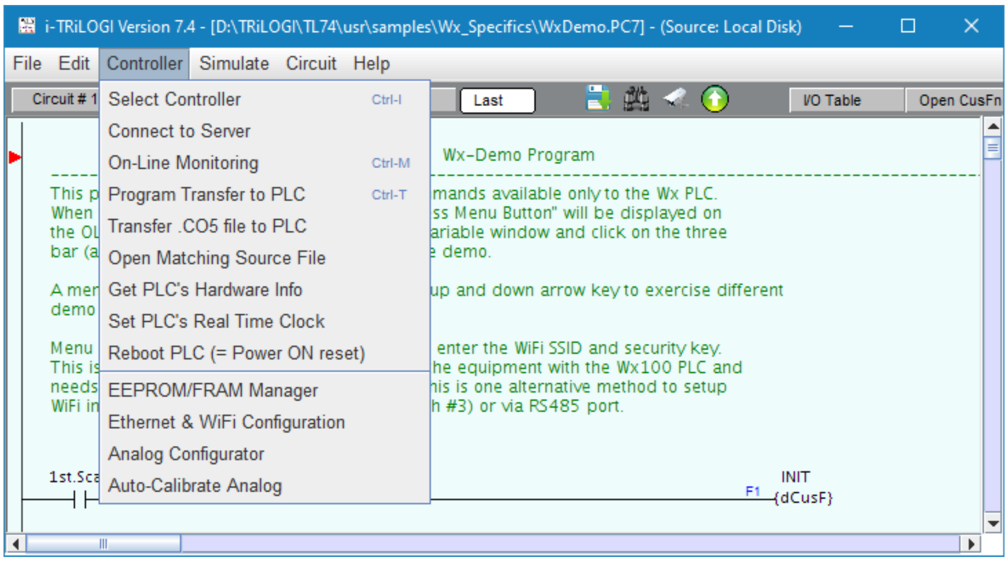
f) Trusted IP Addresses

This feature will be explained in Section 2.5.2 (https://docs.triplc.com/wx100-um/#3003) when we discuss the Modbus/TCP security issue and how to use this capability to protect the Modbus/TCP server from being accessed by unauthorized party.

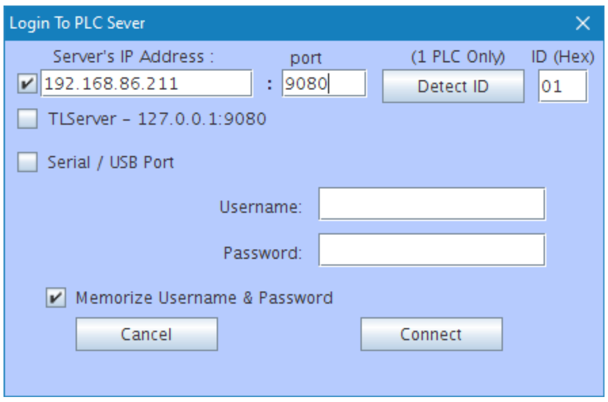
2.3 On-line Monitoring/Programming

Once the Wx100 PLC is connected to the WiFi and has an IP address, you can use i-TRiLOGI software version 7.4 or later to transfer your PLC program to the PLC and perform online monitoring.

All the commands to interact with the PLC are grouped under the "Controller" menu as shown below:


i-TRiLOGI Version 7.4 - [D:\TRiLOGI\TL74\usr\samples\Wx_Specifics\WxDemo.PC7] - (Source: Local Disk)
File Edit Controller Simulate Circuit Help
Circuit # 1
Last
I/O Table
Open CusFn
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
Reboot PLC (= Power ON reset)
EEPROM/FRAM Manager
Ethernet & WiFi Configuration
Analog Configurator
Auto-Calibrate Analog
Wx-Demo Program
mands available only to the Wx PLC.
ss Menu Button" will be displayed on
ariable window and click on the three
demo.
up and down arrow key to exercise different
enter the WiFi SSID and security key.
he equipment with the Wx100 PLC and
his is one alternative method to setup
h #3) or via RS485 port.
1st.Sca
INIT
F1
{dCusF}

When you run any commands (e.g. On-Line Monitoring) under this menu before the iTRiLOGI software is connected to the PLC, you will be presented with the login screen as follow:


Login To PLC Sever

[✓] Server's IP Address : 192.168.86.211 : port 9080 (1 PLC Only) Detect ID ID (Hex) 01

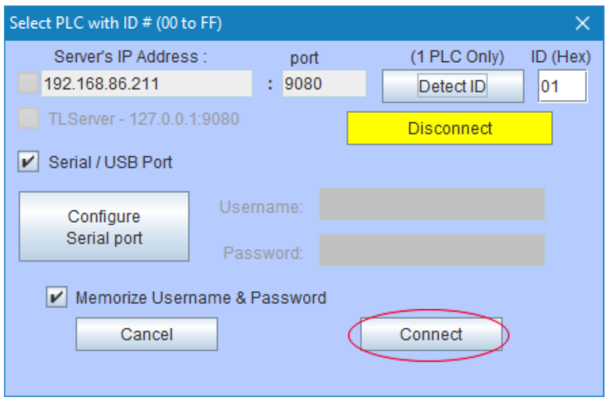
[ ] TLServer - 127.0.0.1:9080

[ ] Serial / USB Port

Username:

Password:

[✓] Memorize Username & Password


Select PLC with ID # (00 to FF)

[ ] Server's IP Address : 192.168.86.211 : port 9080 (1 PLC Only) Detect ID ID (Hex) 01

[ ] TLServer - 127.0.0.1:9080

[✓] Serial / USB Port

Username:

Password:

[✓] Memorize Username & Password

Click the checkbox next to the Server's IP Address and enter the IP address shown on the PLC's OLED screen. Enter the correct port number (default is 9080) assigned to the FServer. If you know the PLC's ID number (01 to FF hex. The default is 01) you can enter it in the ID textfield and then click the "Connect" button. However, if you don't know the PLC ID you can click the "Detect ID" button and i-TRiLOGI will make a connection and then send out the special "IR*" command to the PLC to retrieve its ID. If the PLC returns its ID and a yellow "Disconnect" button appears, that indicates a successful connection has been established. Just click the "Connect" button to complete the connection to proceed to perform all the controller related functions.

Note:

1. If you have defined and enabled the "Use Username/Password (https://docs.triplc.com/wx100-um/#2881)" option in the Ethernet & WiFi Configuration screen, then you must enter the defined username and password in the respective field on the login panel before you click the "Connect" or "Detect ID" button.

2. If you are unable to connect to the PLC, then check that both the PLC and the PC running your i-TRiLOGI software are connected to the same network and are on the same subnet. Generally for a subnet mask of 255.255.255.0, if the PC's IP address is 192.168.1.xxx then the PLC should have an IP address of 192.168.1.yyy and it may not work if the PLC has IP address such as 192.168.0.yyy or 192.168.2.yyy, since this means that the two devices are on different subnets. Likewise, if your PC's IP address is "192.168.0.xxx", and if you are defining static IP address for the PLC then please change your PLC's IP address to "192.168.0.yyy" Also ensure that the PLC's IP address is not already assigned to another device on the same network, otherwise a conflict would occur and communication is not possible.

2.4 Using "Network Services" Command

The Wx100 PLC implements a list of “Network Services” commands (NS), which can be used to instruct the Wx100’s operating system to perform a number of network related functions client connection via the WiFi network. These commands allow the PLC to connect remotely to another PLC in another building or another part of the world via the Internet! This allows peer-to-peer networking, or so-called “M2M” (machine to machine communication) to take place between the PLCs.

The PLC’s TBASIC program uses the PRINT #4 command to execute the NS commands . The PLC can also receive the return response string or from the O/S using the INPUT$(4) command. You may be aware that channel #n in the PRINT #n and INPUT$(n) refers to serial comm port number. For Fx and Wx100 PLC, when n=4 it is to send and receive strings via the Ethernet or the WiFi network and not a physical serial port. We refer serial port #4 as “virtual COMM port”.

Only PRINT #4 and INPUT$(4) are implemented on virtual COMM port #4. The Wx100 currently DOES NOT support the INCOMM (4) and OUTCOMM 4 commands. Note that any non -zero ASCII data can be sent using the PRINT #4 command

The following subsections describe the various Network Service commands available.

2.4.1 <IP> Get IP Address

Note: This IP address is returned instantly, so there is no need to wait for INPUT$(4).

Format:	<IP>
Response:	xxx.xxx.xxx.xxx (IP address)
Example:	PRINT #4 “<IP>” SETLCD 1,1,”Our IP=”+INPUT$(4)

2.4.2 <DNS> Resolve Domain Name

This command resolves the Domain Name string “domain-name” and return the IP Address of the domain.

Format:	<DNS [domain name]>

Success Response:	xxx.xxx.xxx.xxx (IP address string returned by DNS server)
Failure Response:	ERR:07-DNS Unresolved (Either DNS server not properly defined or the domain name does not exist.)
Example:	
STATUS(3):	This function returns 1 on success and 0 on failure.

```
PRINT #4 "<DNS triplc.com>"
PRINT #4 ""
FOR I = 1 to 10000
    A$ = INPUT$(4)
    IF LEN(A$) <> 0
        SETLCD 1,1," IP="+A$
        RETURN
    ENDIF
NEXT
```

Notes:
a) There is no need for the closing tag </> to end this command.

b) If your DNS server has been correctly defined, the above program should return the IP address as a string such as "173.231.246.120". You can then use this IP address string in all the other NS commands to be described in the following sub-sections.

c) The DNS server may take a while to resolve the domain name. If it is unable to resolve the domain name then it will return an error string, so your program should test to see if it receives the ERR:07 error message to determine whether the returned string is useable.

2.4.3 <SNTP> Sync RTC To NTP

This command makes it extremely easy to synchronize the PLC's built-in Real Time Clock (RTC) to the internet time server.

Format:	<SNTP [NTP server domain name]>
Success Response:	RTC.Err flag is cleared

Failure Response: RTC.Err flag is never cleared.

Description: The NTP server domain name is optional. If you don’t specify the domain name then it automatically uses the default NTP server domain name: “pool.ntp.org”, which requires the PLC to have Internet connection. If your PLC has only LAN but no Internet connection, you can synchronize its RTC with a PC on the same LAN that provides the NTP service.**Note:**

1. Only run this function just ONCE after power on RESET when the RTC.Err flag is ON.

2. If you run the PRINT #4 “<SNTP>” command before WiFi is connected to the router, the function will block the PLC program for up to 15 seconds waiting for connection. IF no WiFiconnection is made it will report an error message on the OLED screen and return.

3. The RTC is synchronized to the UTC time. In order for the PLC’s to act and display using local time it is necessary to define the time zone using the <TZ xxxxx> tag. The string assigned to TZ tag is in the same format defined by Unix TZ environment variables. Please refer to the next section (https://docs.triplc.com/#5778) for details

4. Once the PLC has registered to sync with the NTP server it will periodically (about once an hour) updates it RTC to provide very accurate time so there is no need to repeatedly run the “<SNTP>” command.

Example:

```
' Define the timezone where PLC is located
PRINT #4 "<TZ PST+8PDT,M3.2.0/2,M11.1.0/2>"

' Sync the PLC's RTC to default NTP server "pool.ntp.org"
PRINT #4 "<SNTP>"
```

STATUS(2): This function returns 1 on success.

2.4.4 <TZ> Time-zone Settings

Timezone Settings

The TZ string used to define the timezone is the same as the Unix TZ Environment variable. This is a condensed string that contains quite a lot of information, including whether daylight saving time is used and if so, when it begins and ends. For more details on the format please refer to the following URLs:

https://users.pja.edu.pl/~jms/qnx/help/watcom/clibref/global_data.html#TheTZEnvironmentVariable (https://users.pja.edu.pl/~jms/qnx/help/watcom/clibref/global_data.html#TheTZEnvironmentVariable)
https://www.gnu.org/software/libc/manual/html_node/TZ-Variable.html (https://www.gnu.org/software/libc/manual/html_node/TZ-Variable.html)

If your location does not adjust the clock during spring and winter time (daylight savings) then the string is quite easy. You can simply use “UTC+hh:mm” or “UTC-hh:mm”, where hh:mm is the number of hours and minutes to be added or subtracted from your location to make it the same as the UTC time. For example, Singapore is 8 hours ahead of UTC, so the TZ String to use is “UTC-8” (“:mm” is optional and can be omitted as in this case).

If your location is one of the following you can simply copy the string and use in your program:

```
Time-zones in North America
PRINT #4 "<TZ UTC+5DST,M3.2.0/2,M11.1.0/2>"   ' US/Canada EST with daylight savings
PRINT #4 "<TZ UTC+6DST,M3.2.0/2,M11.1.0/2>"   ' US/Canada CST with daylight savings
PRINT #4 "<TZ UTC+7>"                         ' US/Canada MST with no daylight savings
PRINT #4 "<TZ UTC+8DST,M3.2.0/2,M11.1.0/2>"   ' US/Canada PST with daylight savings (default)

PRINT #4 "<TZ UTC+6DST,M4.1.0/2,M10.5.0/3>"   ' Mexico CT with daylight savings

Timezones in Asia

PRINT #4 "<TZ UTC-8>"                         ' China, Singapore. No daylight savings
PRINT #4 "<TZ UTC-9>"                         ' Japan, South Korea. No daylight savings

Timezones in Europe - Daylight Savings Time (starts Last Sunday of March @2am, ends last Sunday of Oct @3am)

PRINT #4 "<TZ UTC-1>"                         ' central Europe = UTC+ 1hour. No daylight savings
PRINT #4 "<TZ UTC-1DST,M3.5.0/2,M10.5.0/3>"   ' central Europe with summer daylight savings time

Timezones in Australia - Daylight Savings (starts first Sunday of Oct @2am, ends first Sunday of Apr @3am)

PRINT #4 "<TZ UTC-10DST,M10.1.0/2,M4.1.0/3>"   ' Australia Eastern Time = UTC+10hour with Daylight Savings
PRINT #4 "<TZ UTC-9:30DST,M10.1.0/2,M4.1.0/3>" ' Australia Central Time = UTC+9:30 with Daylight Savings.
```

2.4.5 <EMAIL> Send Email

Format:	<USERNAME [smtp_username]> <PASSWORD [smtp_password]> <TCPCONNECT [SMTP_URL:port]> <EMAIL [recipient email address]> SENDER: [sender email address] SUBJECT: [whatever text string] [body of the email line 1] [body of the email line 2] … </>
Response:	<OK>-Email successfully sent ERR:04-Not Connected: Failed to connect to SMTP server. ERR:06-Email Failure: Failed to complete email transmission.
STATUS(3):	This function returns 1 on success and 0 on failure.
Description:	By default, this command sends an email via the SMTP server IP address and port number defined in the "Ethernet & WiFi Configuration" tool. However, with Wx100 PLC the program can also use a different SMTP server by specifying its URL using the "<TCPCONNECT xxxxxxx>" tag before running the "<EMAIL recipientEmail>" tag. More importantly, unlike Fx and FMD PLCs, the new Wx100 PLC now can also send email via authenticated (but non- encrypted) SMTP servers. Such SMTP servers typically accepts email via port 587. To support authentication, the program must first run the "<USERNAME xxxxx>" tag, followed by the "<PASSWORD xxxxxx>" tag. When the program finally runs the "<EMAIL xxxxxxx>" tag the Wx100 o/s will automatically negotiate with the SMTP server and supply the username and password mentioned above when prompted. Once the email request is accepted by the SMTP server the program will send the email subject, sender and email body. After the email has been sent the connection must be closed by the "</>" tag. If the program encounters any errors during authentication with the SMTP server the error messages will be reported on the "System & User Log" windows on the i-TRiLOGI 7.4 -> Online Monitoring -> View Variable" screen). Please check the error messages on the log screen, correct all the causes that led to the errors, and try again.

Example:

```
#DEFINELOCAL SMTP_username = "my_smtp_username"
#DEFINELOCAL SMTP_password = "my_smtp_password"
#DEFINELOCAL SMTP_URL = "my_smtp_server_url:587"
#DEFINELOCAL RecipientEmail = "jill5678@gmail.com" ' replace with you
r destination email
#DEFINELOCAL SenderEmail = "Jack1234@yahoo.com" ' must be a valid sen
der email address
SCREEN_CLEAR
SETLCD 1,1,"Sending Email to:"
SETLCD 2,1, RecipientEmail

PRINT #4 "<USERNAME "; SMTP_username;">"
PRINT #4 "<PASSWORD "; SMTP_password;">"

' Directly Connect to SMTP server, skipping settings configured in "E
thernet & WiFi Config"
PRINT #4 "<TCPCONNECT ";SMTP_URL;">"

PRINT #4 "<EMAIL "; RecipientEmail; ">"
PRINT #4 "SENDER: "; SenderEmail
PRINT #4 "SUBJECT: Email sent by your PLC"
PRINT #4 "I am a TRiLOGI PLC"
PRINT #4 "This is a Hello Message"
PRINT #4 "</>" ' end the email.
...
PRINT #4 "</>"
```

2.4.6 <Connect> To Other PLC

Format:	<CONNECT [IP address:port of the other PLC]> [username string] [password string]
Response:	• <CONNECTED>: Successfully connected to remote PLC's FServer IP address:port. • ERR:05-Prev Conn.ON: Another NS command has been executed and left the client socket opened but did not execute the PRINT #4 "</>" to close the client socket. • Failed to connect to remote FServer or TLServer.

STATUS(3):	This TBASIC function returns 1 if the connection is active and returns 0 if the connection has ended. You can test the connection status to determine if the connection is still alive.

Description: This service allows your PLC to log in to another TRi Super PLC such as another Wx100, Fx, FMD or Nano-10 PLC connected to the same network or over the Internet.

You execute this command by first sending the string “<CONNECT xxx.xxx.xxx.xxx:9080>” using the PRINT #4 command, where xxx.xxx.xxx.xxx is the IP address of the remote FServer or TLServer, followed by sending the username and password needed to log in to the remote server. Each line should be terminated with a CR (carriage return) character. (The PRINT #4 command automatically appends the CR character).Once a connection with the remote server is established, the CPU will return the response string <CONNECTED> to the user program, which can read it using the INPUT$(4) function. The STATUS(3) function can also be used to test if the connection is successful and alive. When the program gets the confirmation of connection, it can then use the TBASIC “NETCMD$(4, x$)” command to read or write data to the remote PLCs as if the remote PLC is locally connected to COMM4 port of this PLC, as shown in the following example:

```
A$ = NETCMD$(4, “@01RI00”)
```

Multiple NETCMD$ commands can be executed as long as the connection is alive. You can test the connection status by checking the result of the STATUS(3) function.

Once all the command exchanges have been completed, you should send a </> tag to close the client connection to the remote server so that other NS commands can be executed in other parts of the program.

Example: Please refer to the “fnConnect” and “fnNetCmd” custom functions in the demo program: “TestEthernet.PC7” file.

2.4.7 <REMOTEFS> to TLServer

Format:	`<REMOTEFS [IP Address of remote TLServer 2.1 & above]>`
	`[File Service tag for TLServer]`
	`    ...`
	`</REMOTEFS>`

Response:	[…]: The response strings sent by the remote TLServer in response to the [File Service tag] sent by this PLC. Or, ERR:04-Not Connected: Failed to connect to remote TLServer.
STATUS(3):	This TBASIC function returns 1 if the connection is active and returns 0 if the connection has ended. You can test the connection status to determine if the connection is still alive.
Description:	This commands allows the Wx100 PLC to connect to a remote TLServer to perform any of the “Files & Email Services” that a TLServer normally provides to PLCs that are connected to it. This includes creating text files on a remote TLServer and writing or appending data to it anytime. This makes it very convenient for the PLC to collect large amounts of data and save them to the easily accessible, virtually limitless hard disk storage space that is available in today’s PCs.
Example:	Please refer to the “fnRFS1” and “fnRFS2” custom functions in the demo program “TestEthernet.PC7” file.

Notes:

1. This network service enables the Wx100 PLC to connect to a PC on the same LAN that is running the “TLServer” software. TLServer is part of the i-TRiLOGI Client/Server software Suite. TLServer is a legacy program that allows earlier PLCs manufactured by TRi (1993-2010) that did not have networking and file storage capability to perform files operation on the PC. For more detailed descriptions of the available [File Service Tags] please refer to TRiLOGI programmer’s reference manual under the chapter “File & Email Services”.

2. Most users of Wx100 PLC probably would not need to use TLServer for these services anymore, since the Wx100 has both the networking and file storage capability and is able to access a remote webserver to upload data in real time.

All TLServer’s “File & Email Services” tags, such as <Email>, <WRITE>,<APPEND>, <READ> and <READ RTC> are available to the Wx100 PLC through the use of the <REMOTEFS> tag. You simply have to wrap the abovementioned command tags between the <REMOTEFS IPAddr:port> and </REMOTEFS> tag, where “IPAddr:port” is the IP address and listening port of the remote TLServer. E.g. through the <READ RTC[]> tag, the PLC can synchronize its Real Time clock with a remote TLServer. (As you will later see, this feature is probably not very useful for the Wx100 anymore since Wx100 has the ability to connect to the NIST Time Server to update its real time clock to Atomic clock accurately).

2.4.8 Other Network Services Tags

There are few other powerful networking services that will be explained in other sections, as follow:

1) <TCPCONNECT> – See Section 2.6 (https://docs.triplc.com/#3022)
– Enables the Wx100 PLC to make a TCP connection to any TCP/IP server

2) <MBTCPCONNECT> – See Section 2.7 (https://docs.triplc.com/#3028)
– Enables the Wx100 PLC to connect to any Modbus TCP server and read or write data using READMODBUS, WRITEMODBUS, READMB2, WRITEMB2 commands.

3) <DOWNLOAD_CO5> and <DOWNLOAD_CO5_PAUSE> See Chapter 19 (https://docs.triplc.com/#4800)
– Allows the PLC to automatically update its own program from a username/password protected web server.

2.5 MODBUS/TCP Connection

The Wx100 supports both the FServer and the industry standard MODBUS/TCP server simultaneously. This means that all F-series PLCs are ready to interface directly with many third party industrial control devices that support the MODBUS/TCP protocol. These include the SCADA software, HMI hardware, OPC Server, HVAC controllers and many other industrial control devices.

In addition, the Wx100 can be used both as a MODBUS/TCP SERVER as well as a MODBUS/TCP CLIENT simultaneously. This means that the Wx100 can readily read data from any device that has a MODBUS/TCP server, such as: flow meters, AC/DC drives, HVAC elements, RTUs, network sensors etc. It is also possible to perform peer-to-peer networking with other MODBUS/TCP controllers (e.g. another F-series PLC) over a LAN or over the Internet!

2.5.1 Wx100 MODBUS TCP Server

The Wx100 PLC will listen on the default, well-known MODBUS/TCP port #502 for one or all of the connections. However, it is also possible to change the MODBUS TCP port number as described in Section 2.2.3 (https://docs.triplc.com/wx100-um/#2858)

If you have a MODBUS/TCP client program (e.g. you can download a trial version of "Modbus Poll" from http://www.modbustools.com for testing), you simply specify the Wx100 PLC's IP address and connect to it. Once connected, you will then be able to read from or write to most of the Wx100's internal data from the MODBUS/TCP client. The PLC's I/O and internal variables are mapped to the MODBUS device space according to table 2.5.1.

Modbus addressing can be a bit confusing due to historical reasons. There is a 0-offset or 1-offset addressing scheme. Furthermore the original inventor of Modbus specify Modbus address using 0-xxxx for input bits, 1-xxxx for output bits, 3-xxxxx for input registers, 4-xxxxx for holding registers etc. And the Modbus protocol itself generally uses a 16-bit, 0-offset addressing scheme.

To learn more about Modbus addressing we recommend the following web page:

https://www.elkor.net/support/knowledgebase.php?article=14

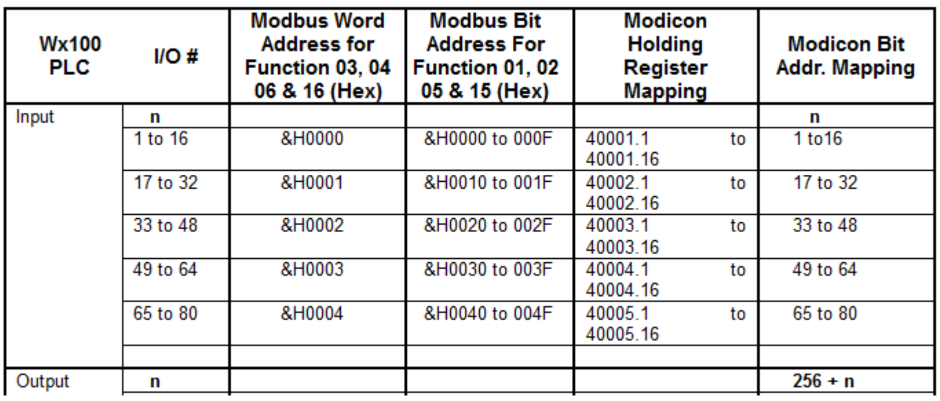
In this document we will refer to both the 0-offset binary address and Modicon addressing.

a) Bit Address Mapping

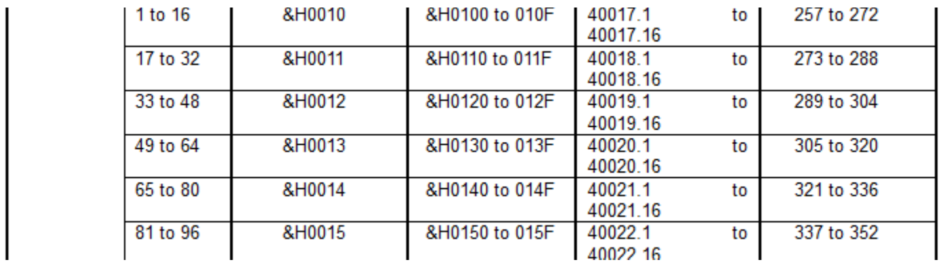
All the F-series I/O bits are mapped identically to both the Modicon "0x" and 1x space. The Modicon bit register offset is shown in the last column of Table 2.5.1. Although Modicon names the "0x" address space as "Coil" (which means output bits) and the "1x" address space as "Input Status" (which means input bits only), the Wx100 treats both spaces the same. Some Modbus drivers only allow a "read" from 0x space and a "write" to 1x space but you still use the same offset shown below on Table 2.5.1 (https://docs.triplc.com/html/table251.htm)

(https://docs.triplc.com/html/table251.htm)

Table 2.5.1 –Memory Mapping of Wx100 Internal Data to MODBUS Register



Wx100 PLC	I/O #	Modbus Word Address for Function 03, 04 06 & 16 (Hex)	Modbus Bit Address For Function 01, 02 05 & 15 (Hex)	Modicon Holding Register Mapping	Modicon Bit Addr. Mapping
Input	n				n
	1 to 16	&H0000	&H0000 to 000F	40001.1 to 40001.16	1 to16
	17 to 32	&H0001	&H0010 to 001F	40002.1 to 40002.16	17 to 32
	33 to 48	&H0002	&H0020 to 002F	40003.1 to 40003.16	33 to 48
	49 to 64	&H0003	&H0030 to 003F	40004.1 to 40004.16	49 to 64
	65 to 80	&H0004	&H0040 to 004F	40005.1 to 40005.16	65 to 80
Output	n				256 + n



	1 to 16	&H0010	&H0100 to 010F	40017.1 to 40017.16	257 to 272
	17 to 32	&H0011	&H0110 to 011F	40018.1 to 40018.16	273 to 288
	33 to 48	&H0012	&H0120 to 012F	40019.1 to 40019.16	289 to 304
	49 to 64	&H0013	&H0130 to 013F	40020.1 to 40020.16	305 to 320
	65 to 80	&H0014	&H0140 to 014F	40021.1 to 40021.16	321 to 336
	81 to 96	&H0015	&H0150 to 015F	40022.1 to 40022.16	337 to 352

(https://docs.triplc.com/html/table251.htm)

Example:

1. To read from the PLC Input 5 via Modbus, you select the Modicon bit address 0-0005.
2. To read from the PLC's output #2, you may have to specify Modicon register address 0-0258 and to write to PLC output #2, you may have to specify Modicon register address 1-0258. However, if your Modbus TCP client allows reading and writing to either 0-xxxx and 1-xxxx space, then you can use either 0-258 or 1-258 and the action will be identical.

b) Word Address Mapping

As shown in Table 2.5.1, to access the PLC's DM[1], you use MODBUS address space 4-1001 (The address naming scheme by Modicon. For binary addressing use 1000 (dec, or &H03E8) and so on. To access the Real Time Clock Hour data (TIME[1]), use 4-0513. The I/O channels can also be read or written as 16-bit words by using the addresses from 4-0001 to 4-0320.

Some MODBUS drivers (such as National Instruments "Lookout" software) even allow you to manipulate individual bits within a 16-bit word. So it is also possible to map individual I/O bits to the "4x" address space. E.g. Input bit #1 can be mapped to 4-0001.1 and output bit #2 is mapped to 4-0257.2, etc. This is how it is shown in Table 2.5.1. However, if you do not need to manipulate the individual bit, then you simply use the address 4-0001 to access the system variable INPUT[1] and address 4-0257 to access the system variable OUTPUT[1]. Note that INPUT[1] and OUTPUT[1] are TBASIC system variables and they each contain 16 bits that reflect the on/off status of the actual physical input and output bits #1 to #16.

2.5.2 MODBUS/TCP Access Security

If a Wx100 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.
Modbus/TCP protocol
is itself inherently insecure and not meant to be exposed directly to the Internet where malicious attacker can take control of the equipment.

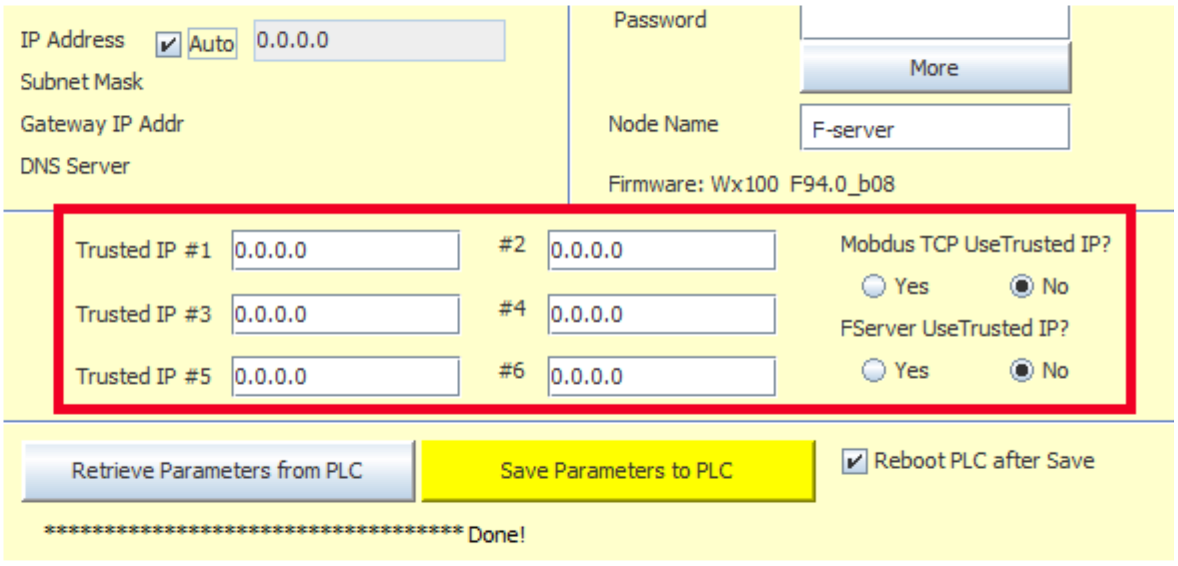
Typically Modbus TCP devices are deployed in a local area network and protected by the network firewall. A SCADA or OPC server program is used to connect to the Modbus/TCP devices on LAN and the PC running the SCADA/OPC server software can then be the only device exposed to the

Internet where secured access can be implemented by the server software. External HMI client software can then make connection to the SCADA/OPC server which will in turn access the Modbus/TCP devices such as the Wx100 PLCs.

However, if you only have a single Modbus/TCP device and need to expose it to the public Internet for external access, 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.

To define a list of “Trusted IP” addresses, please click on TL7.4’s “Controller” menu and select “Ethernet & WiFi Configuration”.

The first thing you should do is to click on the “Retrieve Parameters from PLC” so that you can capture a copy of the current configuration in the PLC and you can then modify selectively.



IP Address	✓ Auto	0.0.0.0	
Subnet Mask			
Gateway IP Addr			
DNS Server			
Password			
Node Name	F-server		

Firmware: Wx100 F94.0_b08

Trusted IP #1	0.0.0.0	#2	0.0.0.0
Trusted IP #3	0.0.0.0	#4	0.0.0.0
Trusted IP #5	0.0.0.0	#6	0.0.0.0

Mobdus TCP UseTrusted IP? Yes [✓ No]

FServer UseTrusted IP? Yes [✓ No]

✓ Reboot PLC after Save

************************************** Done!

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 non-volatile 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.

To see how the response data changes in response to the actual PLC’s input, please turn on some of PLC’s digital inputs 1 to 8, then right-click on the cell where the web query was defined and select te “Refresh Data” command. You should see a new “RIXX” string appear at the selected cell where “XX” is the hexadecimal representation of the 8 input bits 1-8. E.g. if only inputs 2 and 8 are turned ON, then the binary pattern is 1000 0010 which in hexadecimal form is 82 and the response string would therefore be “RI82”. You can then write an excel formula to extract the data “82” and use it for your other computation purpose. By using a different Host Link command, the Excel spreadsheet can read and write to the PLC’s internal data very easily.

Notes:
1. If you have enabled “Use Username/Password” for the FServer, you will be prompted by your Excel program to enter the Username and password before you can receive the response data.

2. We have provided a more complete Excel spreadsheet example “ExcelQuery.xls” which can be downloaded from: http://www.tri-plc.com/appnotes/F-series/ExcelQuery.xls (http://www.tri-plc.com/appnotes/F-series/ExcelQuery.xls)

The macro in this file converts the RIXX data it receives into ON/OFF indicators on the Excel Spreadsheet cells. Note that this spreadsheet file uses the “HEX2DEC” function that is not normally available when you first install the Excel program. But you can add it in by installing the “Analysis Toolpak”. Please search your Excel Help file for the specific method of adding in this toolpak as it may change from one version of Excel to another. On Excel 2000, you can click on the “Tools->Add Ins”, check the “Analysis Tookpak” check box and then click OK. MS Excel will automatically install the toolpak for you.

2.5.3 Wx100 Modbus TCP Client

By using the “Network Services” commands described in Section 2.4, it is unbelievably easy for the Wx100 to be used as a MODBUS TCP client to access any industrial control or HVAC device and sensors that support a MODBUS TCP server. Best of all, you can do it without learning any specifics of TCP/IP programming!

To open a client socket and connect to a Modbus TCP Server that is listening on port 502 (default Modbus TCP port), you only need to send the command tags <MBTCPCONNECT xxx.xxx.xxx.xxx:502> to the CPU via virtual COMM port #4.

E.g.   PRINT #4 “<MBTCPCONNECT 192.168.1.105:502>”

If connection is successful, the system will return the string “<CONNECTED>” on virtual comm. port #4, which you can check with the INPUT$(4) command.

Once the connection is successfully established, you can begin to use the built-in TBASIC commands: READMODBUS, WRITEMODBUS, READMB2 and WRITEMB2 operating on virtual comm. port #4 to send MODBUS commands and receive processed responses from a remote MODBUS TCP Server!! This greatly simplifies your programming task, since it is very similar to communicating with a Modbus RTU slave that is connected to the serial port #1, 2, or 3. Although in this case, the Modbus TCP device could be located in the other hemisphere and connected via the Internet!

The full syntax for the <MBTCPCONNECT> tag is described below:

Format:	PRINT #4 “<MBTCPCONNECT [IP address:502]>”
Response:	• <CONNECTED> Successfully connected to the Modbus TCP server of the specified IP address. • ERR:05-Prev Conn.ON: Another NS command has been executed and left the client socket opened but did not execute the PRINT #4 “</>” to close the client socket. • ERR:04-Not Connected: Failed to connect to the targeted Modbus TCP Server
STATUS(3):	This TBASIC function returns 1 if the connection is active and returns 0 if the connection has ended. You can test the connection status to determine if the connection is still alive.

Description:	This service allows your PLC to log in to any device that supports a Modbus TCP server and is connected to the same LAN or to the Internet. Of course, you may also use it to connect to another TRi Super PLC on the Internet since every TRi Super PLC has a MODBUS TCP server too. Once the connection with the Modbus TCP server is established, the CPU will return the response string <CONNECTED> to the users program, which can read it using the INPUT$(4) function.The STATUS(3) function can also be used to determine if the connection is successful and alive.When the program gets the confirmed connection, it can then use any one of the four TBASIC commands: READMODBUS, WRITEMODBUS, READMB2, WRITEMB2 to read or write data to the remote devices via the virtual comm. port #4, as if a Modbus slave device has been locally connected to a COMM4 port of this PLC. (You do not need to distinguish between Modbus ASCII and RTU in this case, simply use comm. port #4 in your all your commands).Multiple Modbus master commands can be sent as long as the connection is live. You can test the connection status by checking the result of the STATUS(3) function at any time. Once all the command exchanges have been completed, you should send a </> tag to close the client connection to the remote server so that other NS commands can be executed in other parts of the program.
Example:	Please refer to the “fnMBTCP”, “fnRdMBTCP” and “fnWrtMBTCP” custom functions in the demo program: “TestEthernet.PC7” file.

2.6 Getting data from Internet

The Wx100 PLC features a special NS command tag <TCPCONNECT xxx.xxx.xxx.xxx: portno> that allows you to connect to any server to download data. However, since the PLC does not have a lot of memory for storing incoming text data, it is not suitable for downloading information from a commercial website that sends many kilobytes of data in a single download. It can however, be very useful to connect to some servers that send small amounts of information.

Connecting to The Internet Time Server

For example, there are many Internet Time Servers on the Internet that allow users to synchronize their computer clocks via the Internet. The service responds to time requests from any Internet client in several formats, including the DAYTIME, TIME, and NTP protocols. The simplest are those that send responses in ASCII data and you can extract the date and time information from the response ASCII string once you know the format.

You can search on the Internet for a suitable timeserver and use the TELNET program on your PC to access them to examine their display format. Most timeservers listen either on port 13 or port 123 so you need to specify the port number together with their IP address when sending the <TCPCONNECT> command.

Format: <TCPCONNECT [IP address:portno] of time server>

Format:	PRINT #4 “<TCPCONNECT [IP address:portno] of time server>”
Response:	• <CONNECTED>: Successfully connected to the time server of the specified IP address. • ERR:05-Prev Conn.ON: Another NS command has been executed and left the client socket opened but did not execute the PRINT #4 “</>” to close the client socket. • ERR:04-Not Connected: Failed to connect to the targeted time Server
STATUS(3):	This TBASIC function returns 1 if the connection is active and returns 0 if the connection has ended. You can test the connection status to determine if the connection is still alive.
Description:	Once a connection is made, you can then interact with the remote server using the PRINT #4 and INPUT$(4) command. You use the INPUT$(4) command to read CR-terminated text strings sent by the server. You can also send data to the remote server using the PRINT #4 command.
Example:	Please refer to the “fnTCPconn1” custom function in the demo program: “TestEthernet.PC6” to see an example of how the PLC can connect to an NIST timer server and use the returned data to update the PLC’s real-time clock. **Note:** Some NIST time servers have strict policy against abuse so you should avoid sending repeated request within a short period of time, otherwise further connections may be denied once you are considered to have violated their connection policy.

2.7 Web Service: Accessing PLC's From MS Excel

The FServer provides an extremely useful feature called “Web Service”. You can actually use your web browser to access the Wx100 internal data by specifying the following URL:
<IP Address: portno of FServer>/HOSTLINK/<Point-to-point hostlink command without “*”>
E.g. Please enter the following URL into your web browser URL address space:

```
192.168.1.5:9080/HOSTLINK/IR
```

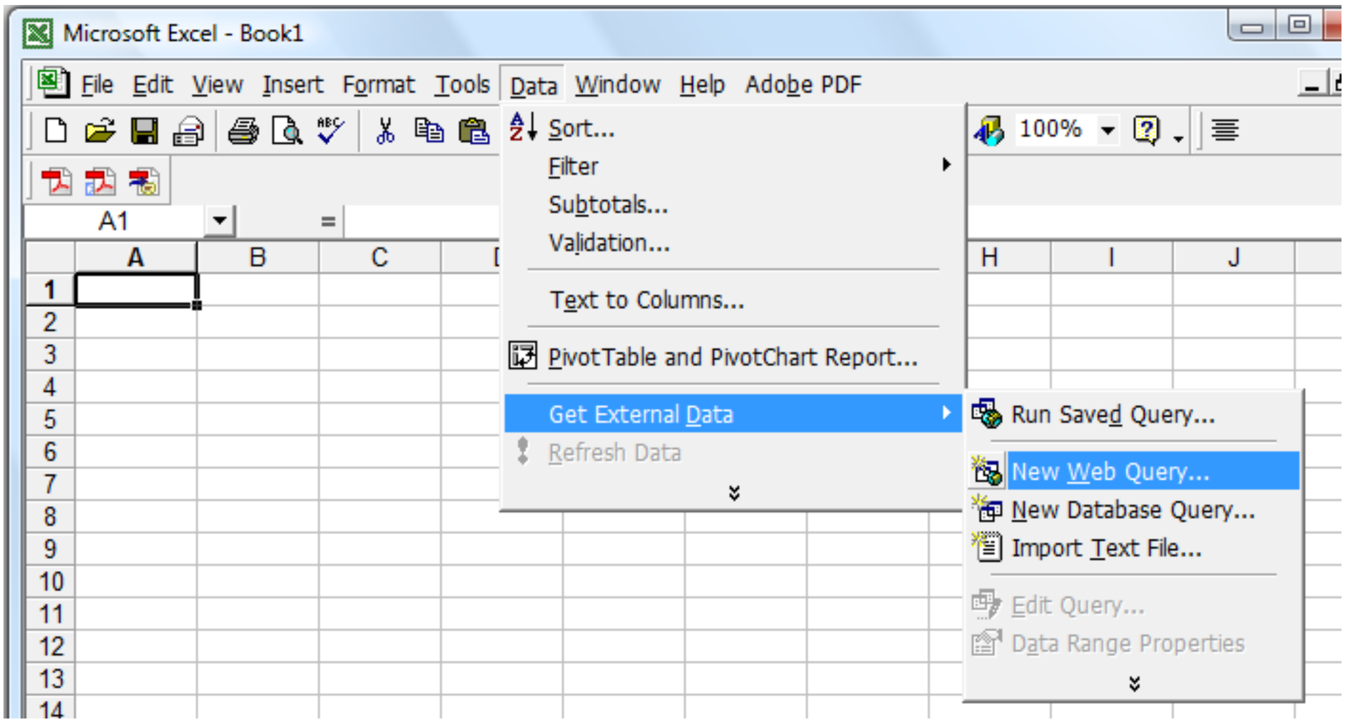
You will see the following data appear on your browser screen:

```
IR01
```

“IR” is one of the many “host link commands” that allows a host computer to read or write to the PLC’s internal data space using ASCII strings. This particular command “IR” is for reading the PLC’s ID and in this case the PLC returns “01” by default. For more details on the list of host link commands, please refer to Chapter 15 of this manual.

Normally the host link commands are sent to the PLC via the serial port (as per all other PLC models produced by TRi). The Fserver, however, permits these host link commands to be sent using the HTTP protocol, which enables the Wx100 to be easily accessible by enterprise software using what is known as “Web Query” methods. The enterprise software only needs to know the format of the host link command required to read the target data and then they can use their web query capability to query the PLC and extract the required data from the response string.

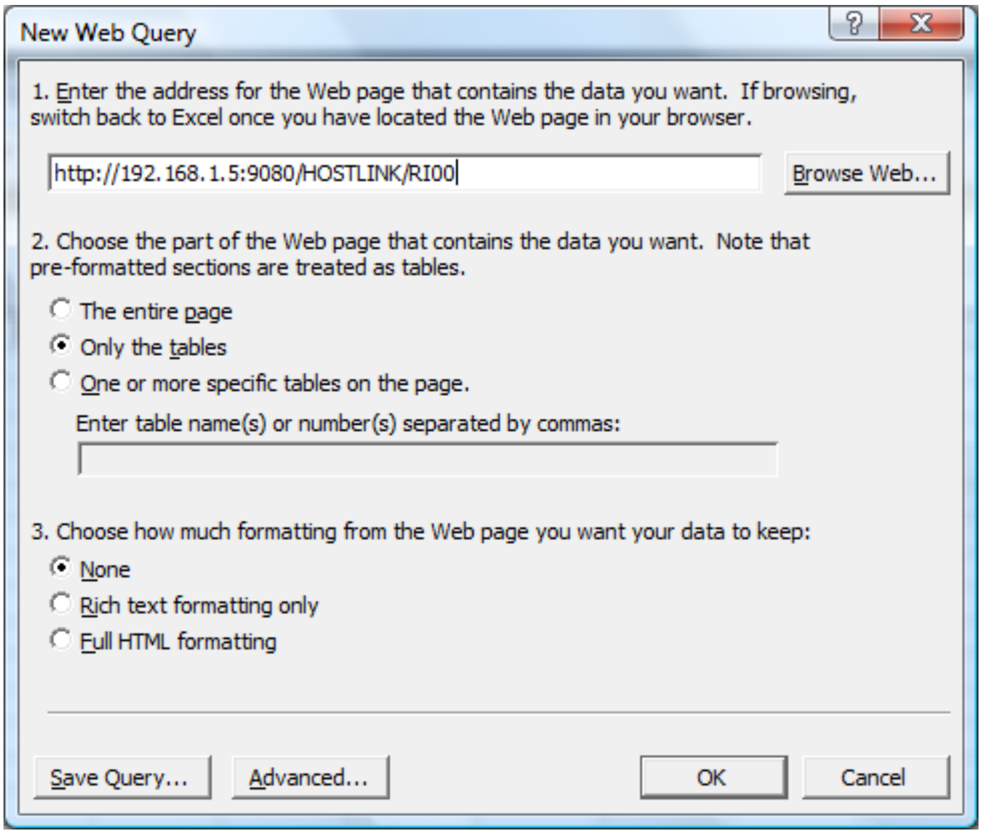
One example, which you can try immediately, is to use the Microsoft Excel 2000 (or later version) spreadsheet program. First, open a blank spreadsheet, then click on the “Data” menu and select “Get External Data” -> New Web Query, as shown below:



Next, please enter the text as shown in the following diagram and then click OK. This will command the Excel spreadsheet to send the web query string “RI00” to the Wx100 PLC that is connected to the network with IP address = 192.168.1.5 and port 9080. The query string “RI00” is for reading the status of 8-bit input channel #0 (which covers the logic states of input bit 1 to 8).

If the FServer is accessible by the PC from the network router, it will send the response data, which will be displayed on the selected spreadsheet cell where the New Web Query was defined earlier.

The response data shown on the cell could be RI00. The response data includes the command header “RI” as defined in the HostLink Command protocol described in Chapter 15. The data 00 indicates that none of the inputs 1 to 8 are currently turned ON.


New Web Query

1. Enter the address for the Web page that contains the data you want. If browsing, switch back to Excel once you have located the Web page in your browser.

http://192.168.1.5:9080/HOSTLINK/RI00

2. Choose the part of the Web page that contains the data you want. Note that pre-formatted sections are treated as tables.

- The entire page
- [✓ Only the tables]
- One or more specific tables on the page.

Enter table name(s) or number(s) separated by commas:

3. Choose how much formatting from the Web page you want your data to keep:

- [✓ None]
- Rich text formatting only
- Full HTML formatting

2.8 Accessing The PLC from Internet

2.8.1 Small Local Area Network

When you connect a Wx100 PLC to your home Ethernet router, the PLC would have joined a “private” local area network (LAN). It is accessible, through its private static IP address, by other devices on the same LAN as long as each device is on the same “subnet” (See section 2.1 for an explanation of subnet settings). The PLC is also able to access the Internet through the router because the router would translate a private TCP/IP packet sent from the PLC into a public TCP/IP packet out of the Internet and if there is any return data from the Internet meant for the PLC, the router would know that and automatically routes the return packet back to the PLC. The router performs what is known as “Network Address Translation (NAT)” and such routers are called NAT routers.

However, the same FServer and Modbus/TCP servers on the PLC are typically inaccessible from the public Internet. This is because the router has a built-in firewall that does not permit external TCP/IP packets from the public Internet to reach the devices on the private LAN. In other words, the NAT router allows the PLC outgoing access to the Internet but by default does not allow incoming access.

Most small NAT routers for home use such as those produced by Linksys, Netgear, D-Link or

Belkin do allow you to configure the router to “open” and “forward” a specific port number to a specific device on the private network. For example, if your PLC static IP address is 192.168.1.5 and you wish to open its FServer port (9080) but not its Modbus/TCP port (502) to the public internet, you would configure your router such that it will forward the incoming TCP/IP packet destined for port number 9080 to the device at IP address 192.168.1.5. Once you have done that, you will then be able to access the FServer from the Internet using the router’s public IP address (this is typically assigned by the Internet Service Provider) and the port number 9080. However, the Modbus/TCP port is not accessible from the Internet since this port number is not opened and not mapped by the router.

You should read your router’s User’s Manual to find out how to configure the router to perform the “port forwarding” described above since each router model has a different user interface.

2.8.2 Large Corporate Local Area Network

In the case of a medium to large corporate LAN, whether incoming and outgoing TCP/IP packets are allowed to go through the corporate firewall is entirely decided by the System Administrator according to the company’s security policy. Most corporate LANs would not allow incoming packets from reaching an internal server until the System Administrator has given the permission to do so. Some company’s network may not even allow devices such as the PLC to open a connection to the Internet to access external data. If your application requires the PLC to access the Internet or to be accessible from the Internet, then you would need to consult your system administrator on the required procedure.

2.9 Installing Control Web Page

The Wx100 PLC web server space can host up to 512K bytes of data files which can be HTML, JPG, JS (JavaScript) files etc. Thanks to its support of the “web services” commands, a programmer can create its own sophisticated control webpages using only standard HTML and JavaScript .

TRi has created an example control webpage which can be downloaded from:

```
    http://www.tri-plc.com/download/webapp/webapp02.zip (http://www.tri-plc.com/downloa
d/webapp/webapp02.zip)
```

When you have successfully unzipped all the files from the Webapp02.zip into your hard disk, you will need to transfer the files to the PLC’s web server using a FTP client program. The following sections describe how to use the free FileZilla FTP program to transfer these files and section 2.10 will describe how to customize the control web pages.

NOTE:

The “M.JS” JavaScript file used with the new preloaded 0.HTM file is no longer provided in the new web app download because it is loaded remotely. In the past, M.JS had been stored inside the PLC and was available directly for editing. However, now the user modifications in the HTML file are more expanded and in order to save space for additional files as well as protect against accidental modification of the M.JS file, it is now stored outside PLC memory.

If you are an experienced web programmer, or you plan to outsource web programming services to customize your PLCs web interface, then you may obtain a copy of this file for more advanced editing by writing to support@triplc.com and providing your purchase references (purchaser’s name, company name, invoice # etc).

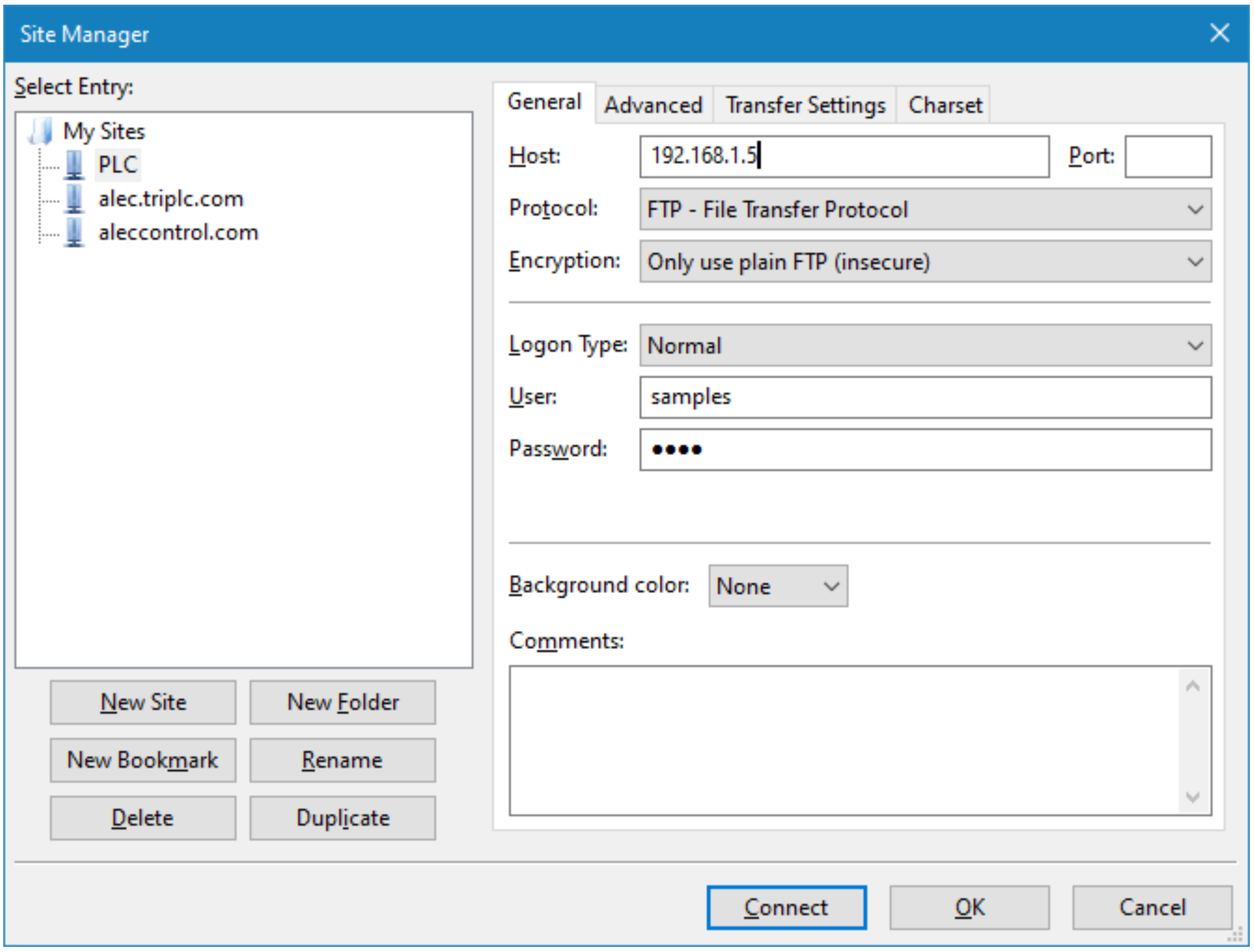
Of course, you may also like to build your own HTML and JavaScript/Jquery control structure, which is entirely feasible for any experience web programmer.

2.9.1 Install & Configure FileZilla

First please download the FileZilla client from: http://filezilla-project.org (http://filezilla-project.org)

Once you have gone to the web page from the above link, you will need to select the client version to download. Then you need to select the software version for your computers platform, typically the Windows version, and download it to your computer. This is the installation setup file, which you will need to execute after it has downloaded. Please perform a default installation by following the installation steps.

After installing the FileZilla client, please open it from the start menu or desktop icon and then go to the “File” menu and select “Site Manager”. The following menu will then pop up:


Site Manager

Select Entry:

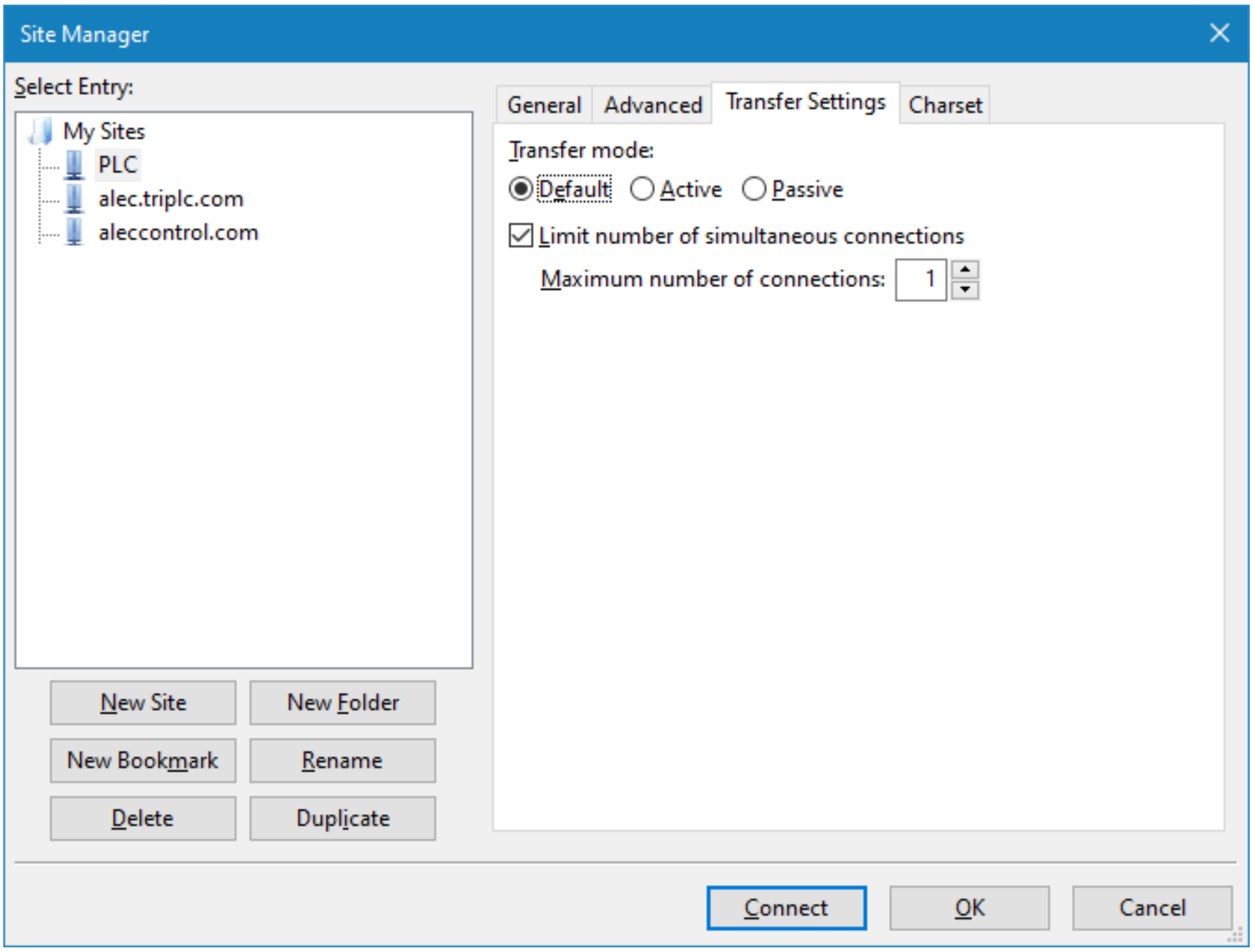
My Sites
- PLC
- alec.triplc.com
- aleccontrol.com

General | Advanced | Transfer Settings | Charset

Host:	192.168.1.5
Port:	
Protocol:	FTP - File Transfer Protocol
Encryption:	Only use plain FTP (insecure)
Logon Type:	Normal
User:	samples
Password:	●●●●
Background color:	None
Comments:	

Under the default “General” tab, you will need to enter the Host IP address and the Port can be left blank . For the Protocol select “FTP – File Transfer Protocol” and as for Encryption, select “Only use plain FTP”.

The Logon type should be set to Normal and the User and Password should be set to the same as for the login to the Fserver (if username/password is enabled). Nothing else needs to be configured in this area. Next go to the “Transfer” settings” tab.



Transfer mode: [✓ Default] Active Passive

[✓] Limit number of simultaneous connections

Maximum number of connections: 1

Under the "Transfer settings" tab you need to set the Transfer mode to "Active" and check the box to Limit number of simultaneous connections. The maximum number of connections should be 1. This is everything that needs to be configured, so you can now click on connect to connect to the PLC's FTP server.

Troubleshooting Connection Problems

Please refer to Section 2.9.3 (https://docs.triplc.com/wx100-um/#3091) for details.

2.9.2 File Transfer To PLC Web Server

Now that you have a connection to the PLC's web server and can view the directory listing, you should be able to see any files that have been preloaded into the PLC.

Downloading Files

The first thing to do is to download the preloaded files from the web server and store them somewhere on your computer. To do this you will need to open up the folder on your computer that you want to store the files in. Then you can drag the files from the web server via FileZilla into your destination folder. It is best to make a copy of these files as a backup on your computer so that you have the original copies available in case you need them.

Uploading Files

When you are ready to transfer your own web pages or applet to the PLC, you just need to drag it from the folder it is saved to on your computer into the bottom right window of FileZilla where the current files are shown. If you didn’t change the filename, you can tell the new file to overwrite the old file. Otherwise, you will need to manually delete the old file from the PLC’s web server in FileZilla by right clicking on it and selecting Delete.

2.9.3 Troubleshooting FileZilla

1) FileZilla appears to have connected to the PLC’s FTP server, but it cannot list the directory or transfer any file to the PLC.

The reason almost always have something to do with the PC’s software firewall. You need to configure the Windows Firewall (and any software anti-virus firewall on your PC) to allow incoming connections to the FileZilla program. You can try to disable the firewall temporarily to test the connection and if does work, you can then be sure that it is the firewall configuration issue that need to be resolved.

The following paragraphs explain how the firewall can affect the FTP communications for those who are interested:

The File Transfer Protocol is unique in that it requires two socket connections between two devices that are communicating via FTP. When FileZilla is connected to the PLC’s FTP server on port 21, it establishes a “command” channel and it is through this command channel that FTP commands are being sent.

However, once the command channel is established (FileZilla is “connected” to the FTP Server), a second socket connection (known as the “data channel”) needs to be made between FileZilla and the PLC. All data, such as the directory information and content of any files to be transferred between FileZilla and the PLC will need to go through the data channel. They are two possible ways of establishing this data channel, one is called the “Active Transfer” and the other is known as “Passive Transfer”.

FileZilla is able to operate in either transfer mode, but the PLC FTP Server can only operate in “Active Transfer” mode. “Active Transfer” mode requires that the client (FileZilla) provides a listening socket for the server (PLC) and the server (PLC) will then try to connect to this socket to establish the data connection. So if the FileZilla program sits behind the software firewall and no exception has been configured, then the PLC FTP Server will not be able to make a connection to the FileZilla data socket because it is blocked by the firewall and the connection will therefore fail.

This explains why FileZilla seemingly able to connect to the PLC but yet is unable to send any data or list the directory – because there is no data channel connection for it to do so.

However, sometimes the problem could persist after disabling all firewalls and in that case the only option is to revert to an older version of FileZilla, and then transfer the files by doing a right-click download or right-click upload (if drag and drop doesn't work). You can do this as follows :
a. Download the FileZilla client version 3.1.5.1 from http://www.oldapps.com/filezilla.php
b. Remove the current version and then install version 3.1.5.1
c. Connect to the PLC
d. There should be two directories (same as always) : local site (your PC) and remote site (the PLC). You should see your files in the PLC directory.
e. Open the folder in the local site that you would like to save the files to.
f. Right click on a file in the PLC directory such as 0.HTM and select download. You should see the file in the folder on your local site.
g. To move files from the PC to the PLC, open the folder in the local site where the file is located and right click it to select upload. You should see it in the PLC or you will be asked to overwrite it if the same file name is there already.

2) FileZilla Can Connect The First Time But Could Not Re-connect After Time-Out

FileZilla is normally setup to time out after 1 minutes of no activity. However, Since the PLC's FTP server can only handle a single FTP connection at a time, it is good to avoid letting FileZilla time out due to no activity. This is because when FileZilla times out it cutoff the connection to the PLC. But if for whatever reason the PLC were to miss the disconnection information it will be left in a "half-open" state and will not be able to accept a new connection. When this happens and if you try to reconnect to the PLC again after the connection has been dropped by FileZilla, you will most likely be unable to connect to the PLC again until after the PLC time out its FTP connection due to no activity. The only quick fix is to power-on reset the PLC so that it starts up fresh and can accept new connection again.

You can set the FileZilla timeout settings to a larger number (e.g. 300 seconds) using the FileZilla's "Edit->Settings" menu item so that it will not time-out automatically too quickly. You should always select "disconnect" after a file transfer to properly close the connection and then the PLC will be properly disconnected and is ready to accept new connection.

3) Problem Using FileZilla To Transfer Files to Multiple PLCs Configured To The Same Default IP Address Even Though Only One PLC Is Connected To The Router At A Time.

When you connect FileZilla to a PLC with a particular IP address (e.g. 192.168.1.5) for the first time, Windows will memorize the MAC-ID address of that PLC and associate it with this IP address in its "ARP cache" (ARP = Address Resolution Protocol). So when you power off one PLC and immediately plug another PLC with the same IP address to the network, then Windows will pass to FileZilla the MACID of the PREVIOUS PLC instead of the new PLC using data in its ARP cache memory. When FileZilla tries to connect to the right IP address but a wrong MACID, it of course would not be able to connect properly.

In such a scenario, you must clear the Windows ARP cache first using a command prompt as follow:
The “arp –d” command tells windows to delete its ARP cache data. So when you use FileZilla to connect to a new PLC Windows will do the necessary to properly connect to the new PLC that has a different MACID from the previous PLC.
Note that the same issue applies to using I-TRiLOGI to transfer program to multiple PLCs that are all set to the same default IP address, even though only one PLC will be connected to the network at a time. You will need to clear the ARP cache after every transfer to avoid problem. Or else you have to wait for windows to expire its ARP cache data before connecting the next PLC.

2.10 Access & Customize PLC Control WebApp

The second generation of Web control application is now available free for any TRi PLC user to download and install into their PLC. In the previous section (2.9 Installing a Control Web Page Into the F-series PLC) you were shown how to transfer html files to the FServer.
Click on the following link to download this new web application.

```
    http://www.tri-plc.com/download/webapp/webapp02.zip (http://www.tri-plc.com/downlo
ad/webapp/webapp02.zip)
```

The “Webapp02.zip” file you downloaded above (or in Section 2.9) for Fx PLCs contains the following:

- Four sample control web pages (0-C001-02.HTM, 0-C002-02.HTM, 0-C002-02-NoLCD.HTM, and 0-C003-02.HTM)
- A readme file (Readme.txt)
- A download link (UserGuide.htm) for the quickstart modification guide
- PLC program files for TRiLOGI 6 and 7 (webAppTest.PC6 and webAppTest.PC7) that can be loaded in the PLC

The user guide describing how you can modify the control webpage can be downloaded directly from:

```
    http://www.triplc.com/documents/WebApp_UserGuide.pdf (http://www.triplc.com/docume
nts/WebApp_UserGuide.pdf)
```

2.10.1 Next Generation Super PLC Web Control

These web files can be accessed from any standard web browser that supports AJAX technology (Internet Explorer, Firefox, Chrome, Safari) locally as long as your Wx100 PLC is connected to the LAN (local area network) or from anywhere in the world as long as your Wx100 PLC is connected to the Internet. Even the iPhone, Android phones, and other smartphones that support AJAX can be used to access these web pages, which means your application can be controlled and monitored from anywhere while on the go.

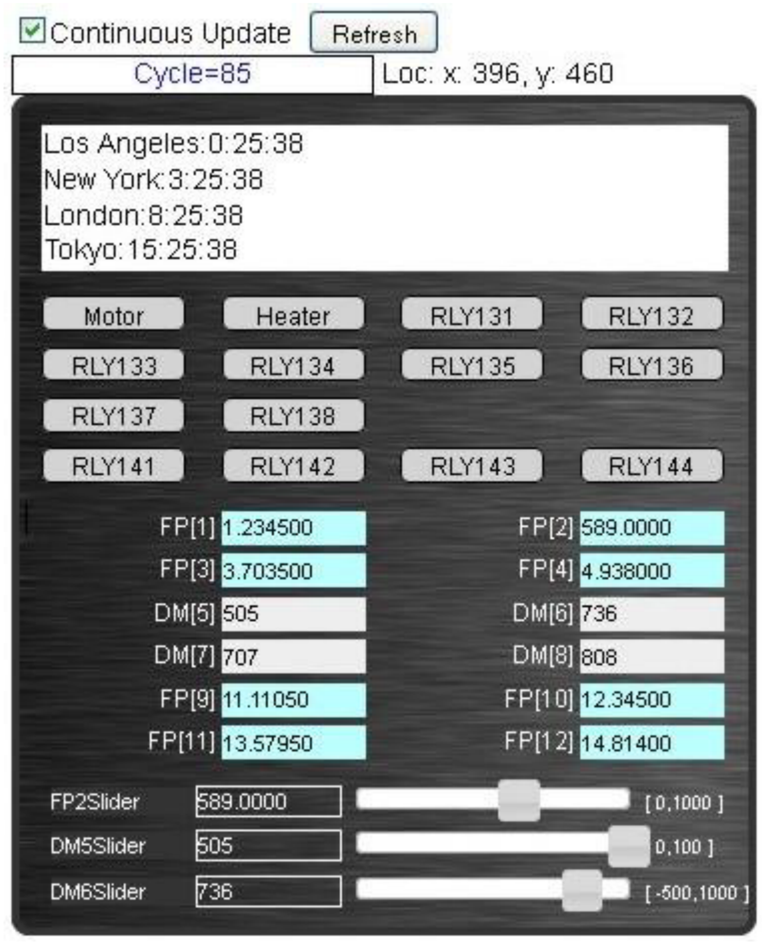
This new version makes use of the newer Jquery UI and no longer utilizes the M.JS JavaScript file. Instead it calls the Jquery file remotely (no longer stored inside the PLC).

Either way, the web files are only accessible from standard browsers that support AJAX technology. However, the HTML files are designed to be easily modified by anyone, even if you don't have any programming experience. The purpose is to be able to easily customize the web page layout for your specific application by defining some label names and a background image without having to worry about the programming required to interface to the PLC, which is already taken care of.

The basic HTML file "0.HTM" is the file where simple modifications can be made. The other files are variation of the basic "0.HTM" file.

Note that unlike the file system in Fx PLC, in the Wx100 the file system now can be any name instead of limited to 0.xxx to Y.xxx and Z001.xxx onwards. So feel free to rename these files to other names to test them.

The first release of this new Web control application accepts up to: 16 I/O buttons linked to Relay #128 to #144; 16 floating point FP[] variables; 16 DM[] variables, and 8 sliders that can be attached to display and/or control any of the FP[1] to FP[16] or DM[1] to DM[16] variables. Any of these control elements can be selectively displayed and their position, size and color are all user customizable using a simple text editor, with no programming required at all. See below for a screenshot of 0-C001-02.HTM.



Refer to the user guide linked above for more information on the available user modifications.

2.10.2 Advanced Customization

The “M.JS” JavaScript file used with the new preloaded 0.HTM file is no longer provided in the new web app download because it is loaded remotely. In the past, M.JS had been stored inside the PLC and was available directly for editing. However, now the user modifications in the HTML file are more expanded and in order to save space for additional files as well as protect against accidental modification of the M.JS file, it is now stored outside PLC memory.

If additional modifications need to be made beyond what can be done in the HTML file, then the following two options are available:

a) Enlist Design Services

Contact our Solution Partner, Sparrow Design LLC (https://triplc.com/rsp_sd.htm), for browser/app control design services.

b) Request the JavaScript Files.

If you are an experienced web programmer, or you plan to outsource web programming services to customize your PLCs web interface, then you may obtain a copy of this file for more advanced editing by writing to support@triplc.com and providing your purchase references (purchaser's name, company name, invoice # etc).

Of course, you may also like to build your own HTML and JavaScript/Jquery control structure, which is entirely feasible for any experience web programmer.

Users only need to refer to chapter 16 in this manual for the list of Hostlink commands to be sent in web query form (refer to section 2.7) in order to implement web based data exchange with TRi PLCs.

Chapter 3 - Programming The I/Os

The Wx100 will have a certain no. of physical digital inputs and outputs depending on the configuration of its expansion boards, but all will have 512 internal relays available in both ladder logic and BASIC.

3.1 Ladder Logic DIO Programming

The physical I/O and internal relays can be programmed in ladder logic in a few simple steps.

3.2.1 For Physical I/O

1. Edit the label names
2. Place the input contact(s) into the ladder logic circuit
3. Place the output coil at the end of the ladder logic circuit

3.2.2 For Internal Relays (Non-Latching)

1. Edit the label names
2. Place the relay contact(s) into the ladder logic circuit
3. Place the relay coil at the end of the ladder logic circuit

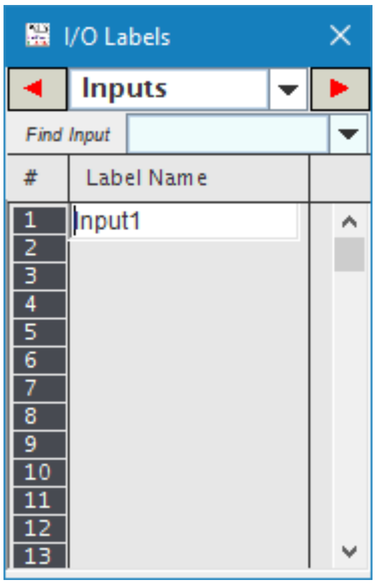
3.2.3 For Internal Relays (Latching)

1. Edit the label names
2. Place the activating input/relay contact into the ladder logic circuit
3. Place the latching relay in parallel with the activating contact
4. Place the relay coil at the end of the ladder logic circuit

3.2.4 Programming Examples:

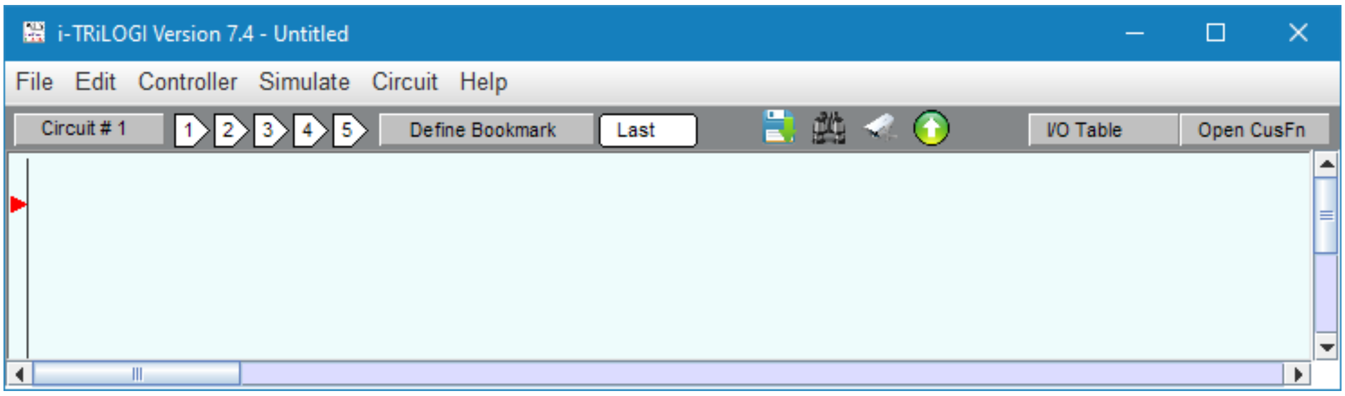
a) Example 1 – Editing Label Names

The Digital I/O can be named by selecting "I/O Table" from the "Edit" menu and choosing the particular digital I/O that you want to name. In Figure 3.1, physical input #1 is being named "Input1".

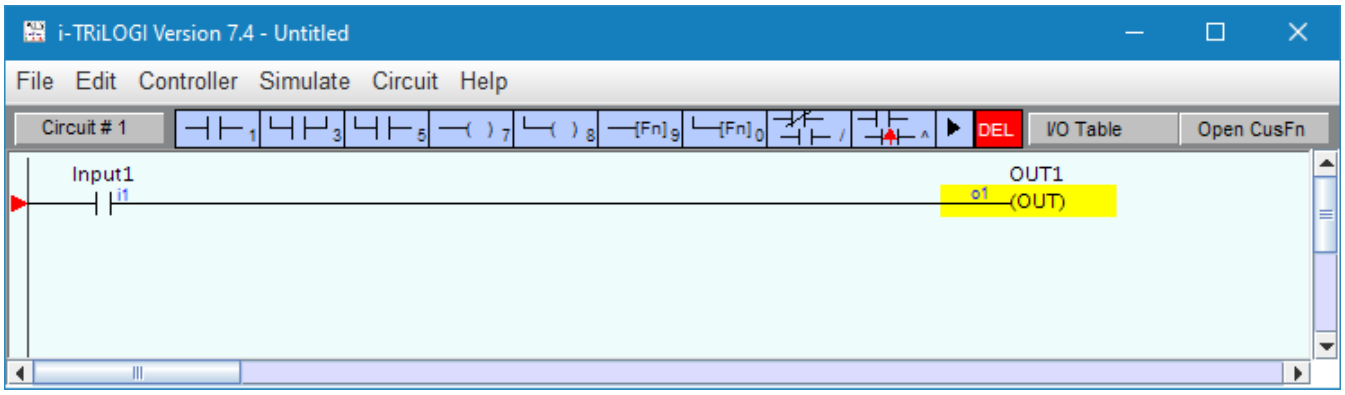


b) Example 2 – Creating a Simple Ladder Logic Circuit

You can place components in the circuit by clicking in the green area to the right of the red arrow, as shown below. This will bring up the component tool bar in the gray area above the green circuit area.



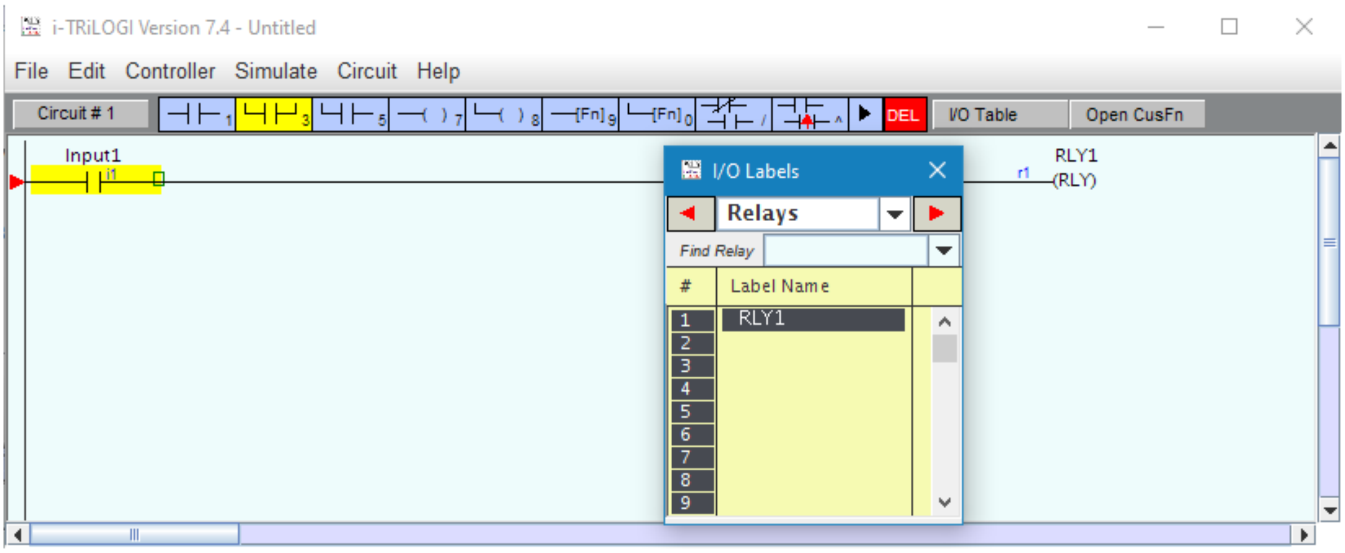
Once the component toolbar is shown, you can place your input/relay contact by selecting the #1 component from the toolbar and then selecting the digital input from the I/O Table. The contact will then be automatically placed in the ladder logic circuit. The same can be done for the output coil by selecting the #7 component from the toolbar and then selecting an output that has been entered into the I/O Table. After selecting one input and one output, the ladder logic circuit should like something like the figure below:

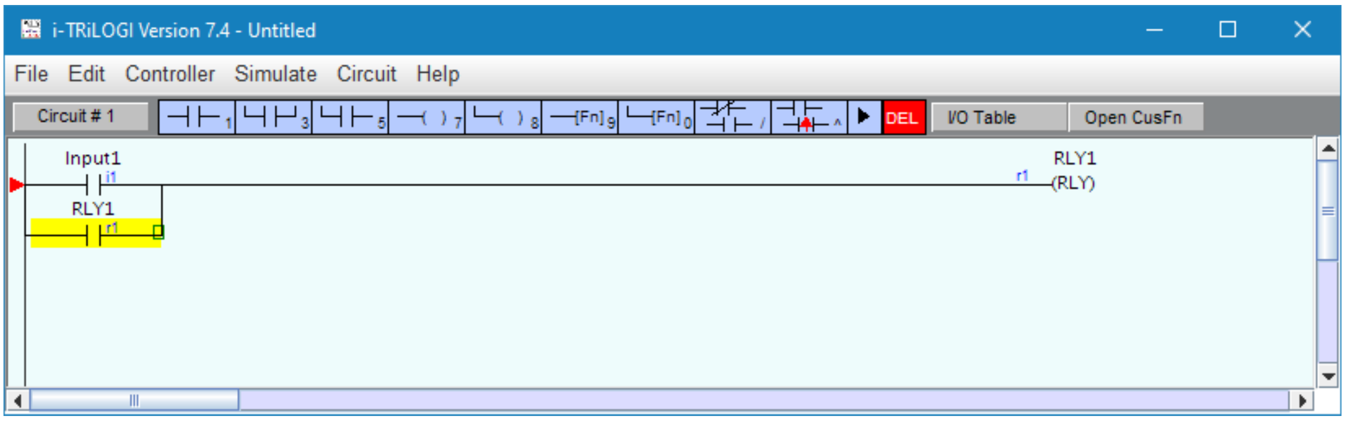


i-TRiLOGI Version 7.4 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 1
DEL
I/O Table
Open CusFn
Input1
i1
OUT1
o1
(OUT)

c) Example 3 – Creating a Latching Relay Circuit

The first part of the circuit follows the same procedure as the previous example, except that the #7 coil should be a Relay coil. So it should look similar to the circuit in Figure 3.3. The next part requires a parallel contact to be added to the Input1 contact. This is done by selecting the Input1 contact (or whichever contact was used) and then adding the "#3" parallel contact as shown in the following figure:




i-TRiLOGI Version 7.4 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 1
DEL
I/O Table
Open CusFn
Input1
i1
RLY1
r1
RLY1
r1
(RLY)

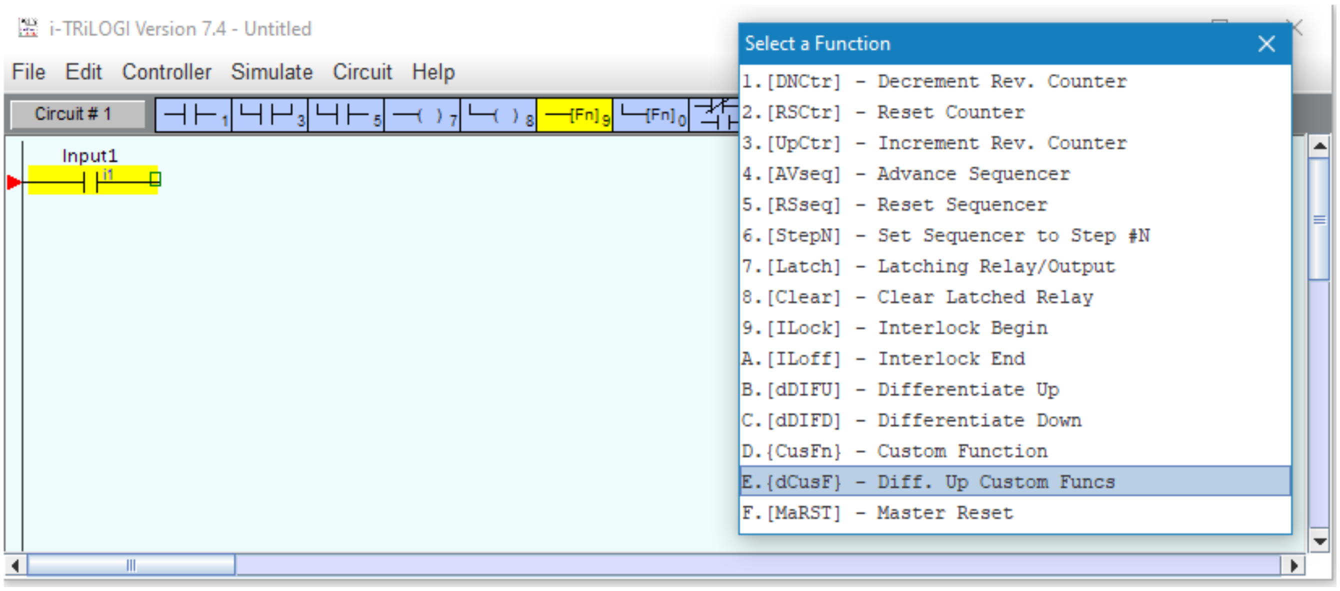
3.2 Programming DIO in TBASIC

In order to program digital I/O or anything in a custom function, a custom function must be created in the I/O Table and added in a ladder logic circuit. Custom functions act the same way as coils in ladder logic, in that that they need a contact to activate them. Once they are activated, the code inside them will execute.

To create a custom function circuit, follow these 3 steps:

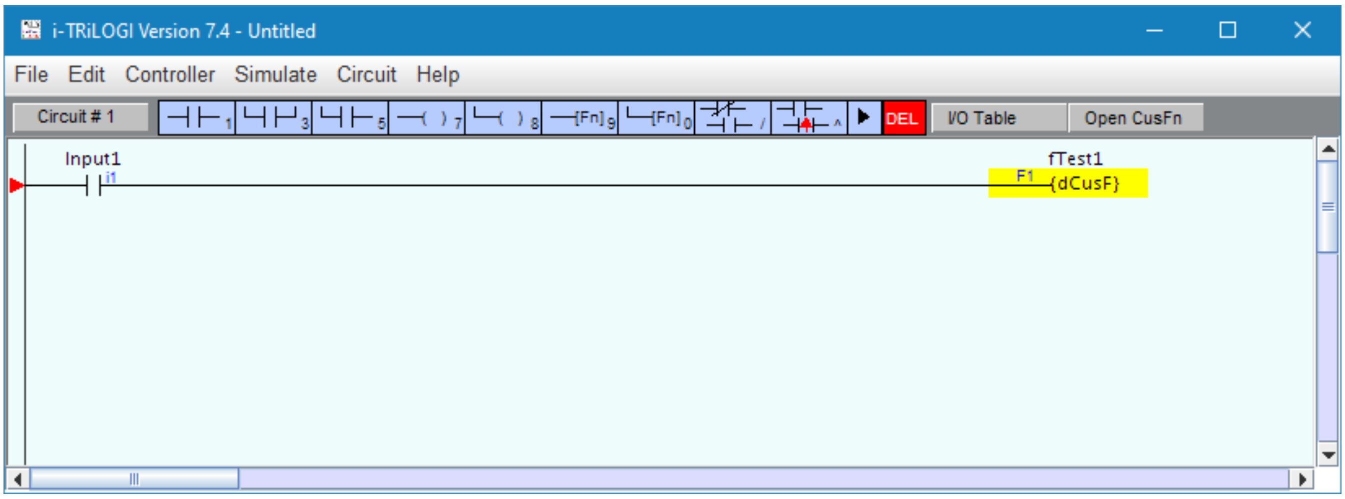
1. Edit the name of the custom function in the I/O Table
2. Place the activating contact in the ladder logic circuit
3. Place the custom function at the end of the circuit

Placing the custom function in the circuit is done the same way as other ladder logic contacts and coils, by selecting the —[Fn] symbol and then choose the Differential custom function {dCusF} from the pop-up window. Gives the custom function a name when prompted (up to 16 characters max) – e.g. fTest1

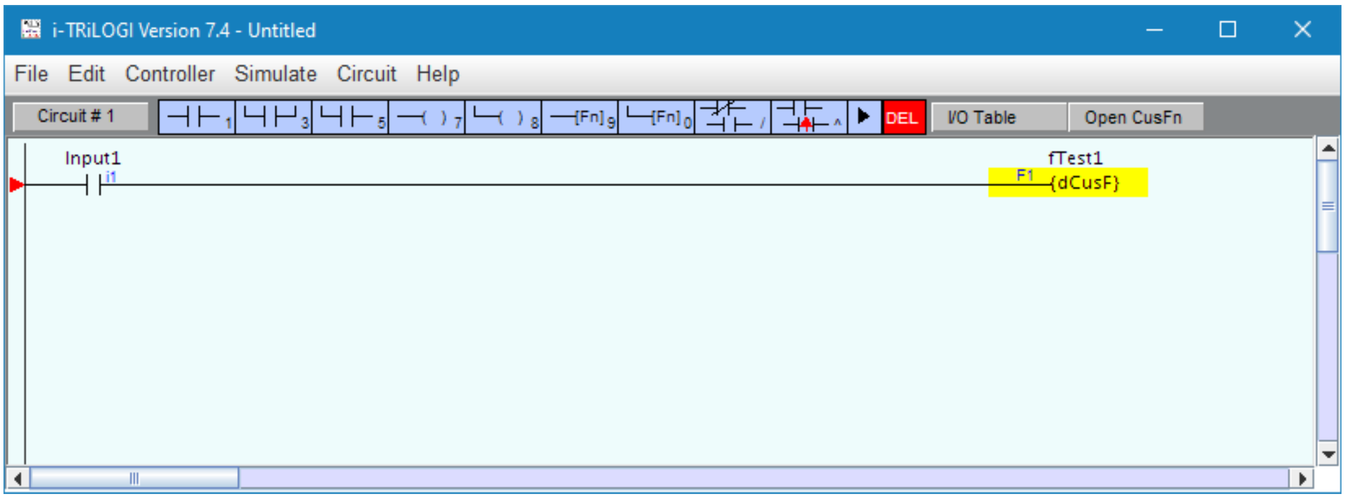

i-TRiLOGI Version 7.4 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 1
Input1
i1
Select a Function

```
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
```

The completed circuit should look something like below:


i-TRiLOGI Version 7.4 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 1
DEL
I/O Table
Open CusFn
Input1
i1
fTest1
F1
{dCusF}

An empty custom function looks like this:


i-TRiLOGI Version 7.4 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 1
DEL
I/O Table
Open CusFn
Input1
i1
fTest1
F1
{dCusF}

TBASIC code is entered into the custom function, which allows the possibility of total control of all of the PLCs functions and hardware.

There are 7 TBASIC functions available to control all of the digital I/O, which are:

```
1. SETIO labelname
2. CLRIO labelname
3. TOGGLEIO labelname
4. TESTIO (labelname)
5. SETBIT v,n
6. CLRBIT v, n
7. TESTBIT (v, n)
```

Each function has its own advantage depending on what needs to be done to a digital I/O. Each of these functions is explained in the programmer's reference manual, which should be referred to for further information. Here are some examples of how to control digital I/O using these functions.

Example 1 – Turn on/off an Output

This can be done using both the SETBIT v,n / CLRBIT v,n command and the SETIO labelname / CLRIO labelname command.

1. Using SETBIT v,n / CLRBIT v,n

```
SETBIT OUTPUT[1], 0 'This will turn on the first output using the output[] register
CLRBIT OUTPUT[1], 7 'This will turn off the 8th output using the output[] register
```

2. Using SETIO labelname / CLRIO labelname

```
SETIO out1  'This will turn on the output out1
CLRIO out5  'This will turn off the output out5
```

In this case, out1 and out5 would need to be entered in the I/O Table as an output. Otherwise, there will be a compilation error.

Example 2 – Toggle an Output

```
TOGGLEIO light 'This will change the output, light, from off to on or on to off
```

The output, light, would need to be entered into the I/O Table as an output as well and could represent any desired output.

Example 3 – Test the Status of an Output

This can be done using both the TESTBIT (v, n) and TESTIO (labelname) command.
Using TESTBIT (v, n)

```
X = TESTBIT (INPUT[2], 1) 'status of input #10 (on = 1, off = 0) is stored in variable
</me)
X = TESTIO (button) 'status of input label 'button' (defined in the I/O table) is store
d in variable X
```

Chapter 4 - Timers, Counters & Sequencers

4.1 Introduction

a) Timer Coils

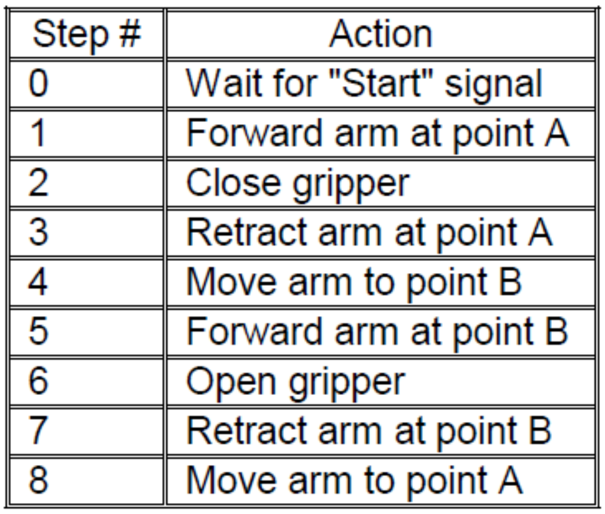
A timer is a special kind of relay that, when its coil is energized, must wait for a fixed length of time before closing its contact. The waiting time is dependent on the “Set Value” (SV) of the timer. Once the delay time is up, the timer’s N.O. contacts will be closed for as long as its coil remains 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 the last aborted timer is re-energized, the delay timing will restart and use the SV of the timer rather than continue from the last aborted timing operation.

b) 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 decrements its count register value by 1. Note that the coil must go through an OFF to ON cycle in order to decrement the counter. If the coil remains energized all the time, the counter will not decrement. Hence, a counter is suitable for counting the number of cycles an operation has gone through. When the count register hits zero, all of 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].

c) Sequencers

A sequencer is a highly convenient feature for programming machines or processes that operate in fixed sequences. These machines operate in a fixed, clearly distinguishable step-by-step order, starting from an initial step, progressing to the final step, and then restarting 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

4.2 Timers/Counters In Ladder Logic

The timers and counters can be programmed in ladder logic in a few simple steps.

For Timers

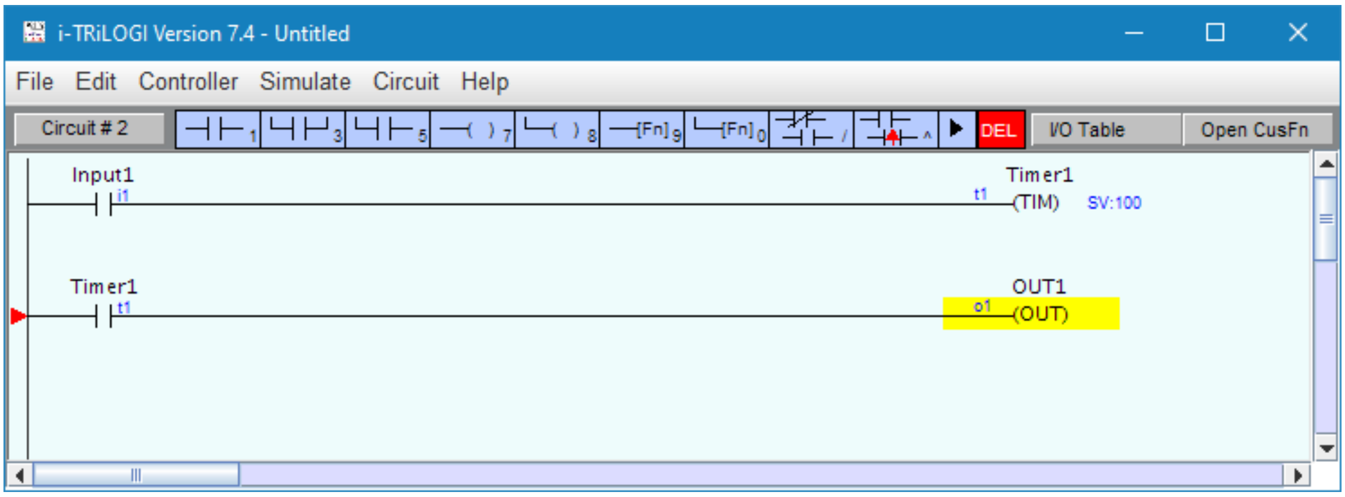
1. Edit the label names
2. Place the input contact(s) into the ladder logic circuit
3. Place the timer coil at the end of the ladder logic circuit
4. Place the timer contact in one or more ladder logic circuits

For Counters

1. Edit the label names
2. Place the input contact(s) into the ladder logic circuit
3. Place the counter coil at the end of the ladder logic circuit
4. Place the counter contact in one or more ladder logic circuits (optional)

Example 1 – Creating a Simple Timer Circuit in Ladder Logic

After creating a ladder circuit that contains one input and one timer output and another ladder circuit that contains one timer contact and one output, the ladder logic circuit should like something like Figure 4.2, below:



4.3 Timers/Counters in CustFn

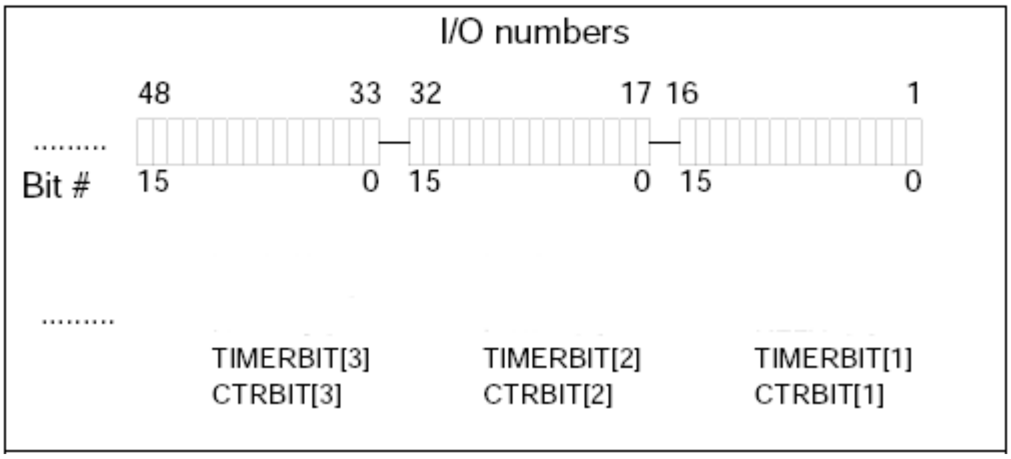
a) Timers and Counters Present Values

The present values (PV) of the 64 timers and 64 counters in the PLC can be accessed directly as system variables:

```
timerPV[1] to timerPV[64], for timers' present value
ctrPV[1] to ctrPV[64], for counters' present value
```

b) Inputs, Outputs, Relays, Timers and Counters Contacts

The bit addressable I/Os elements are organized into 16-bit integer variables 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:



c) Changing The Timer and Counter Set Values in a Custom Function

You can use the SetTimerSV and SetCtrSV functions to change the Set Value (SV) for a timer and counter respectively. An example of this is shown below:

```
SetTimerSV 1,500 'Define Timer #1 to have a Set Value of 500
SetCtrSV 10,1000 'Define Counter #10 to have a Set Value of 1000
```

d) Controlling a Timer or Counter in a Custom Function

You can activate a timer or a counter directly from within a custom function simply by assigning their present value counter to a desired value. E.g. To start a 50 seconds timer:

```
TIMERPV[2] = 500   ' Timer #2 will time out 50.0 seconds later.
```

E.g. To decrement a counter or a sequencer:

```
CTRPV[10] = CTRPV[10] - 1 ' Counter #10 is decremented by 1.
```

4.4 Sequencers on Ladder Logic

Introduction

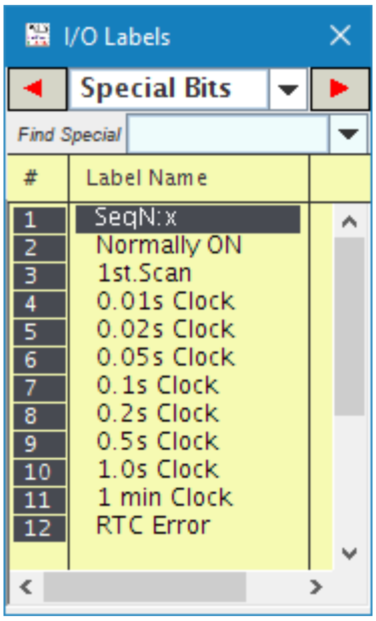
TRiLOGI Version 7 supports eight sequencers of 32 steps each. Each sequencer uses one of the first eight counters (Counter #1 to Counter #8) as its step counter. Any one or all of the first eight counters can be used as sequencers “Seq1” to “Seq8”.

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.

A circuit that uses the special function “Advance Sequencer” [AVSeq] will need to be constructed. 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 changes 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 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. 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 and 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.

Quite a few of the ladder logic special functions are related to the use of the sequencer. These are described below:

a) Advance Sequencer – [AVseq]

Increment the sequencer’s step counter by one until it overflows. This function is identical to (and hence interchangeable with) the [UpCtr] function.

b) 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.

c) 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.

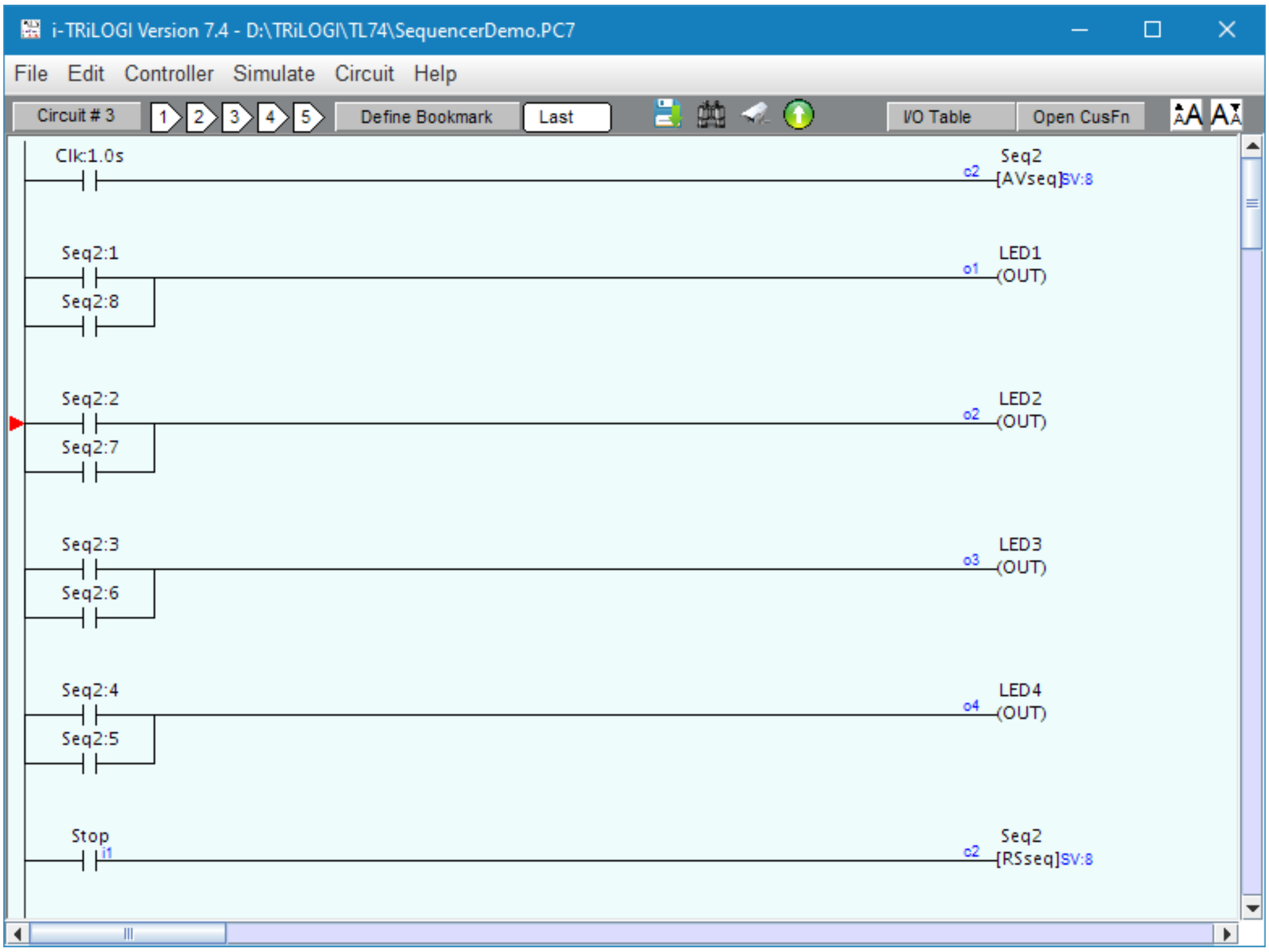
d) Reversing a Sequencer

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.

e) 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 in Figure 4.5.

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 LEDs are turned ON by Step 0, all LEDs will be OFF.



4.5 Sequencers in CustFn

You can change the current step of Sequencer easily from within a Custom Function by changing the present value of their equivalent counter. E.g. Sequencer #3 is the same as Counter #3, thus if you wish to assign Sequencer #3 to Step 10, you can achieve it as follows:

```
CTRPV[3] = 10
```

Chapter 5 - Analog I/Os Programming

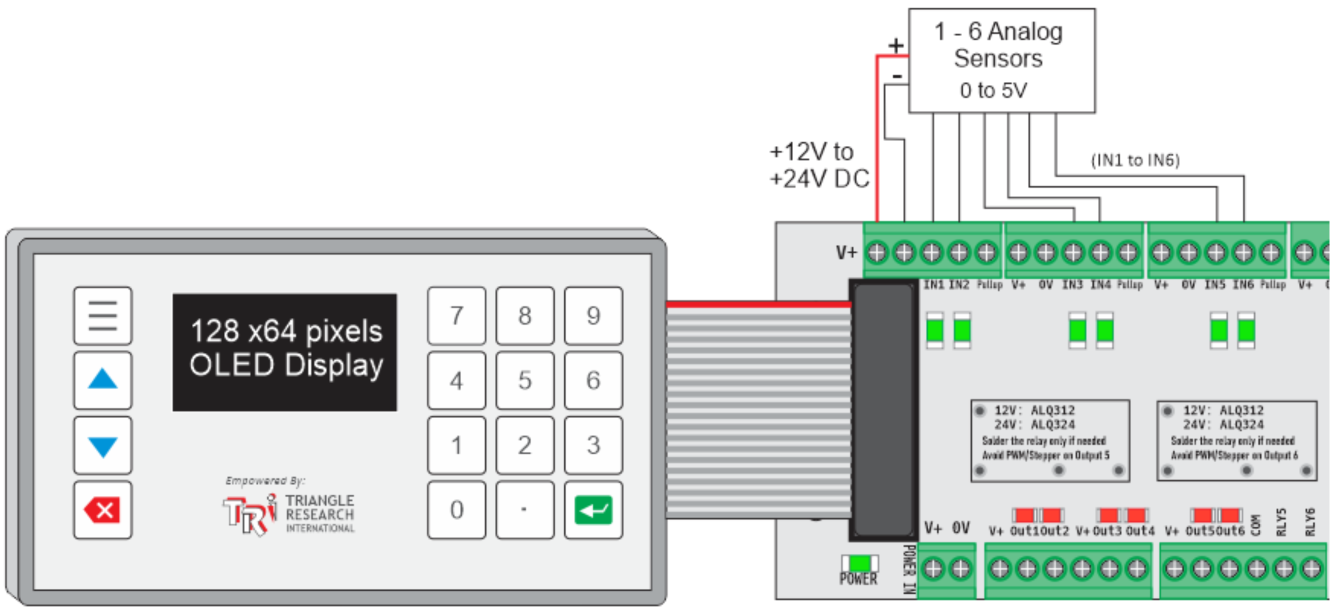
5.1 Analog Inputs: Electrical

Wx100 PLC has built-in 6 channels of 12-bit, 0.5V to 5V analog inputs that is pin-shared with digital inputs 1-6.

When the PLC is first powered on input 1-6 defaults to digital inputs. Once you run a ADC(n) command the input #n becomes an analog input. The ADC(n) function returns the 12-bit analog reading from channel n. The built-in analog input is unable to properly read analog input voltage < 0.5V. Hence the range is between about 412 and 4095.

a) Wiring ADC to Wx100 PLC via Wx-TERM8 or Wx-TERM16

With Wx-TERM8 and Wx-TERM16, you simply wire the 0-5V output from an analog sensor directly to any of the input 1-6 and 0V to the Wx-TERM8 0V input and run the ADC(n) command to obtain the analog data.

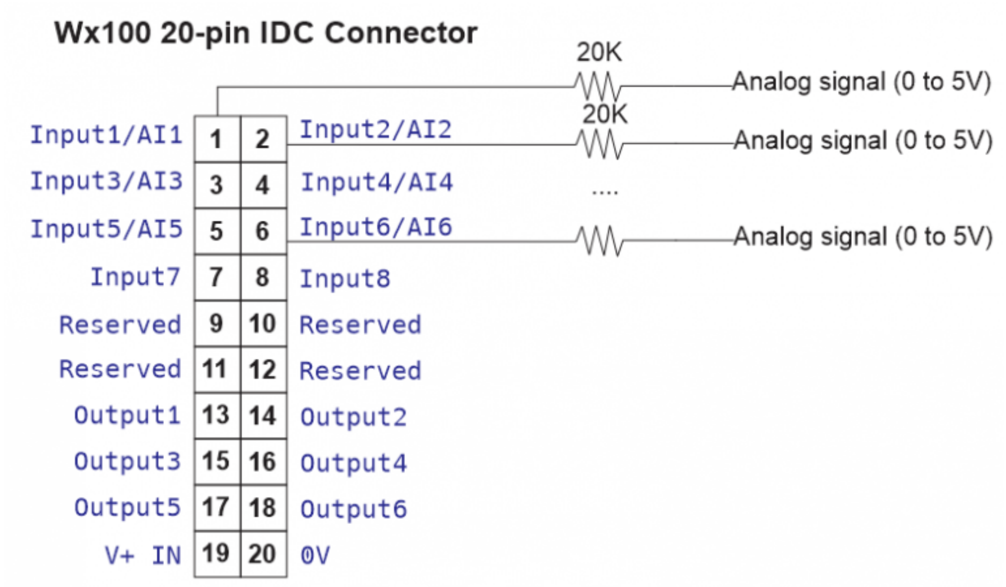


1 - 6 Analog Sensors
0 to 5V
+
-
+12V to +24V DC
(IN1 to IN6)
V+
IN1 IN2 Pullup V+ 0V IN3 IN4 Pullup V+ 0V IN5 IN6 Pullup V+
128 x64 pixels OLED Display
7 8 9 4 5 6 1 2 3 0 .
Empowered By: TRIANGLE RESEARCH INTERNATIONAL
12V: ALQ312 24V: ALQ324
V+ 0V
V+ Out1 Out2 V+ Out3 Out4 V+ Out5 Out6 COM RLY5 RLY6
POWER
POWER IN

b) Wiring ADC to Wx100 PLC via Customized Terminal Board

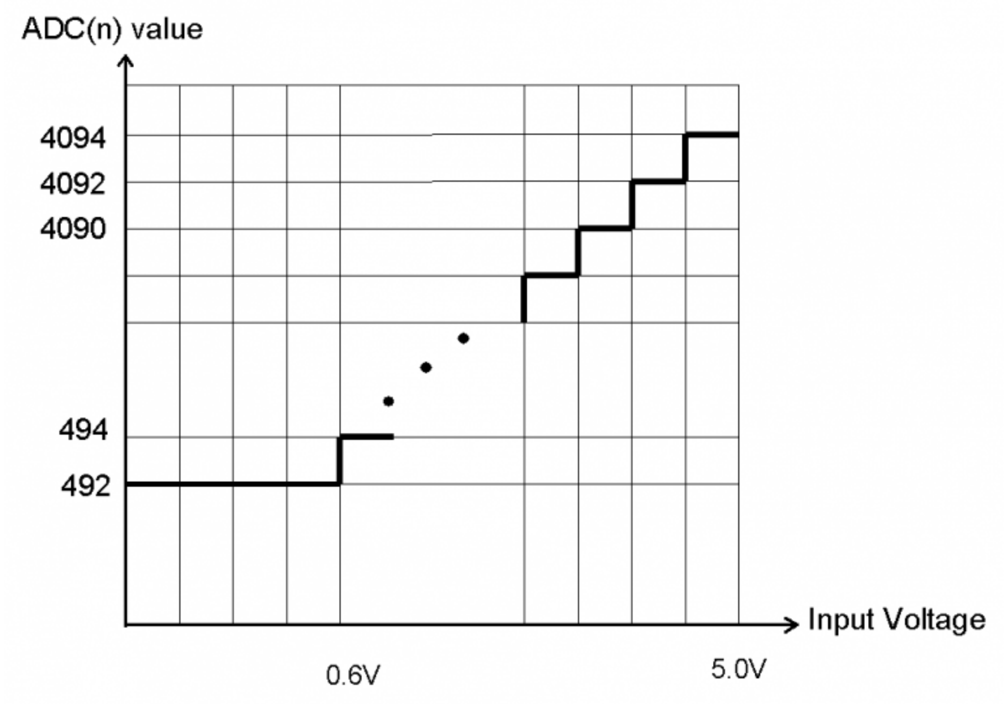
Note that each digital/analog input on a WxTERM-8 and WxTERM-16 incorporates a 20K ohm 1% current limiting resistor in between the input terminal and the signal to the Wx100 ribbon cable connector. The Wx100 has scaled the analog inputs that include the 20K ohm resistors in its calculation.

Therefore, if you design your own terminal board and have designated an input to be analog input, then you need to add a 20K ohm 1% resistor in series with the analog signal.



c) Electrical Characteristics

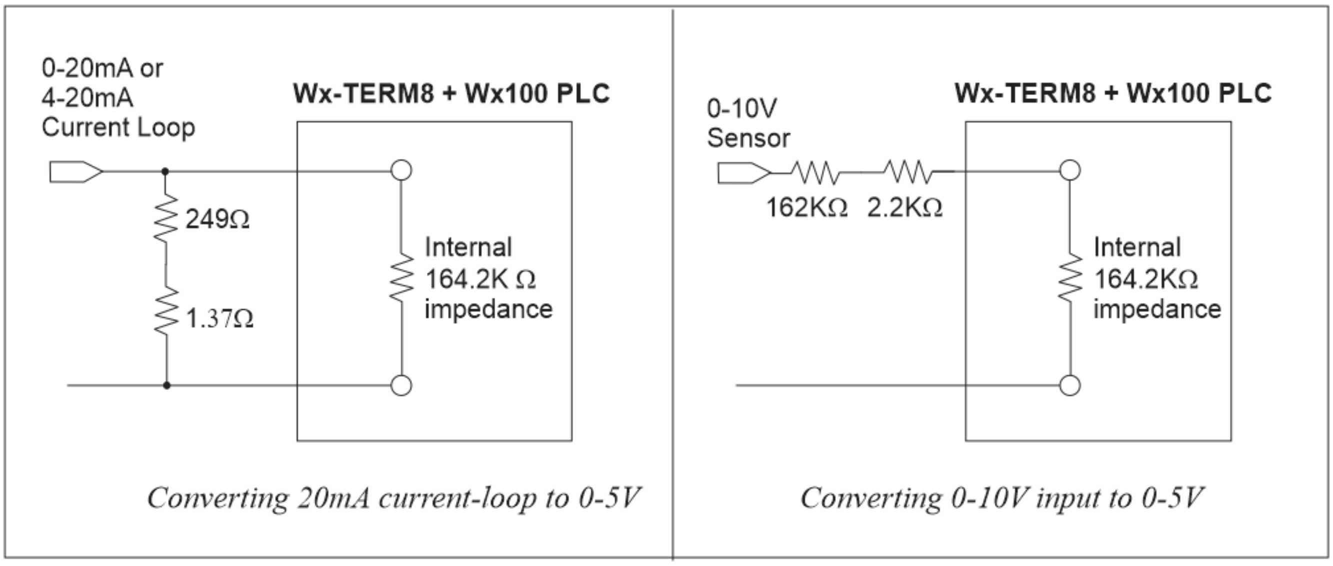
No. of A/D channel	6
Resolution	12-bit
Input Impedance	164.2K ohm



Note that each ADC input channel has an input impedance of 164.2K ohm. Hence to avoid loading effect your sensor need supply at least (5/164200 = 0.03mA) at 5V.

d) Interfacing to 0-20mA and 0-10V Analog Signals

With addition of simple resistors you can easily convert 0-20mA or 0-10V signal to 0 to 5V which can then be measured by the Wx100 analog inputs. Please refer to the following diagrams:

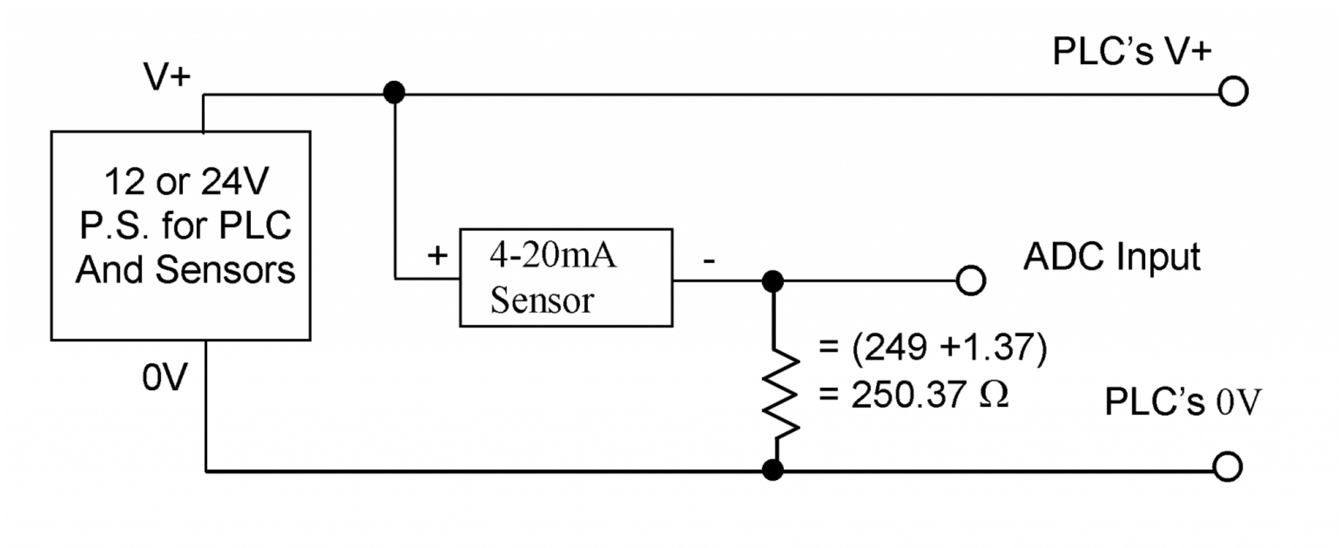


Note: The parallel resistance of the (249+1.37=250.37 ohm) and the internal 164.2K impedance of the Wx-TERM8+Wx100 PLC yields 250 ohm, which is then used to convert 20mA current to 5V.

e) Interfacing to Two-Wire 0-20mA Analog Signals

Many 4-20mA analog sensors only have two wire connections and are designed to be powered by the 4-20mA output current that flows through it. These types of sensors can be interfaced easily to the 0-5V analog inputs of the PLC as shown in the following diagram:

The sensor output will be converted to a 1-5V analog voltage and can be read by the PLC using the ADC(n) statement, which will return readings of between 820 and 4092.



5.2 Analog Inputs: Expansion

Up to 64 additional 12-bit, 0-5V analog inputs can be added to Wx100 PLC with adequate expansion I/O boards. Please refer to the Wx-EXP User Manuals for more details when the product is available.

5.3 Analog Inputs: Programming

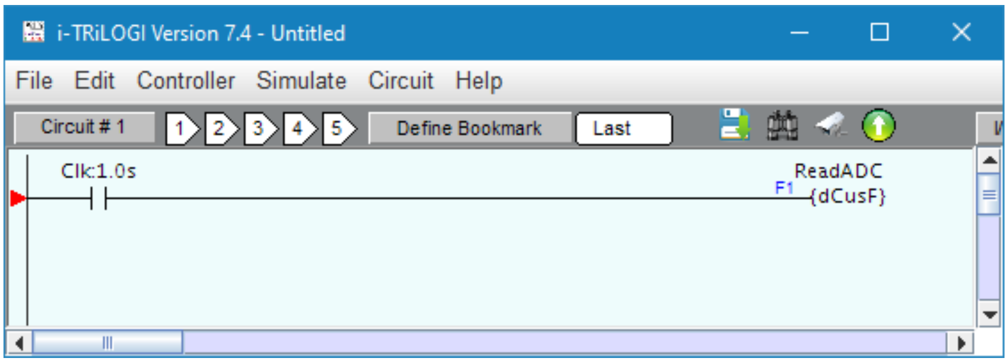
The 6 analog input signals are read by the TBASIC command ADC(1) to ADC(6). The ADC(n) function will return a number between 0 and 4095(12-bit resolution), which corresponds to the measured voltage at any of the analog inputs n. The resolution of a 12-bit ADC is 1/4096 = 1/4096, this means that for the 0-5V ADC range, the resolution is 5/4096V = 1.22mV.

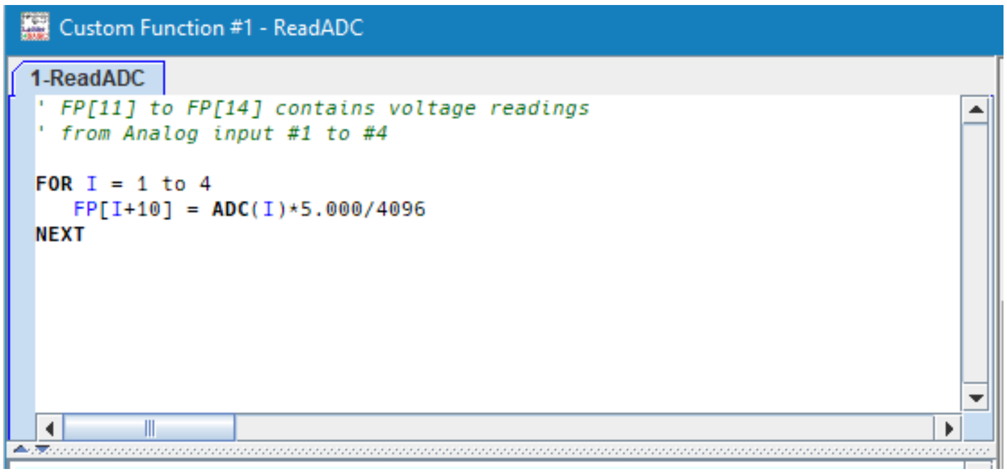
That means that if you apply a 2.500V to the PLC's analog input #3, ADC(3) should return a value of 2.500/5.000 x 4096 = 2048.

Note that the CPU only accesses the analog input #n when the TBASIC function ADC(n) is called. Since the analog input on Wx100 is pin-shared with the digital inputs, the pin will be configured as analog input the first time ADC(n) function is run on the channel n. The shared input terminal will be reset to digital inputs only when the Wx100 is power cycled.

Also, in order to monitor the analog input, you have to execute the ADC function periodically. The frequency that the ADC function is called is known as the "sampling rate" and it depends on how fast the analog data changes. If the analog data changes slowly (such as room temperature) then there may be no need to sample the analog at high frequency.

A very simple example of sampling the analog inputs #1 to #4 every second and converting the data into voltage readings of 0.50 to 5.00V) is shown as follow:


i-TRiLOGI Version 7.4 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 1
1 2 3 4 5
Define Bookmark
Last
Clk:1.0s
ReadADC
F1
{dCusF}



```
' FP[11] to FP[14] contains voltage readings
' from Analog input #1 to #4

FOR I = 1 to 4
   FP[I+10] = ADC(I)*5.000/4096
NEXT
```

Moving Average

The Wx100 PLC offers a built-in “Moving Average” computation routine for ADC channel 1-6. When moving average is enabled the PLC firmware would store the past analog readings for each channel in its own historical memory array, and each new instantaneous reading would overwrite the oldest reading. When you run the ADC(n) function, the PLC firmware would return the average of these past readings instead of the instantaneous new reading.

You can define a moving average of 1 to 9 points using the procedure described in Section 5.4.5

Defining a larger moving average can better help to even out fluctuations in ADC readings that can be caused by interference from digital noise. However, the larger the moving average, the slower the ADC(n) function can detect a sudden change in the amplitude of the analog signal due to the averaging effect.

For a system that needs to quickly detect a signal change, consider using either a smaller number of moving average points or call the ADC(n) function more frequently so that a sudden change can be detected earlier.

To further even out fluctuation in the analog readings, you can perform a number of reads and then take their average.

E.g.

```
     A = 0
     FOR I = 1 to 100
        A = A + ADC(1)
     NEXT
     B = A/100  ' B contains a average of 100 readings taken from the ADC channel #1
```

Scaling of Analog Data

The analog inputs on the Wx100 return data in the range of 0 to 4092, which corresponds to the full range of the voltage input presented at the analog pin. However, very often a user needs a formula to translate this numeric data into units meaningful to the process (e.g. degree C or F, psi etc). To do so, you need to know at least two reference points of how the native unit maps to the PLC’s ADC reading.

```
 Reference Point    ADC Reading
     x1                 a1
     x2                 a2
```

Hence, for any reading A = ADC(1), the corresponding X is derived from:

```
 (X - x1)/(x2 - x1) = (A - a1)/(a2- a1)
 --> X = (x2 - x1)*(A - a1)/(a2 - a2) + x1
```

Note that since x1, x2, a1, and a2 are all constants the actual formula is much simpler then it appears above. E.g. Temperature measurement

```
 Temp         ADC(n)
 ------------------------
 30.0          200
 100.0         3000
```

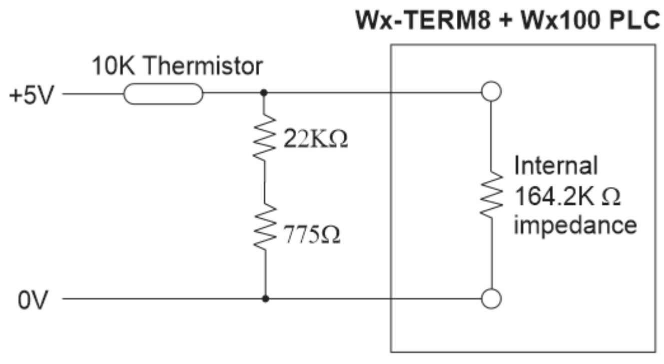
So for any ADC readings A, the temperature is: X# = 70.0*(A – 200)/2800 + 30.0

5.4 Temperature Measurement

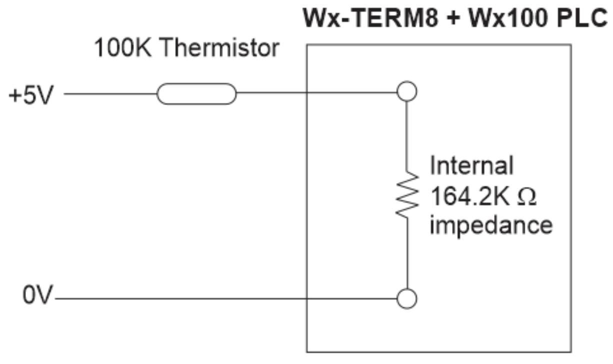
a) Thermistor Temperature Sensors

A thermistor is a kind of resistor whose resistance decreases when its surrounding temperature increases. It is a very low cost and stable device that can be used to measure a wide range of ambient temperature from freezers to hot water boilers, which are commonly used in HVAC applications.

In order to convert the resistance changes into voltage readings to be read by the PLC's analog input, you can use it to form an arm of a voltage divider circuit which would present a variable voltage to the analog input when the temperature changes. A type of thermistor that measures 10.0K ohm at 25 degree C (simply called 10K thermistor) is especially suitable for use with the Fx2424 PLC, as illustrated below:



10K Thermistor Interface



100K Thermistor Interface

For ADC #1 to #6, you can use an external 22.775 K Ohm resistor to form the voltage divider with the 10K thermistor. The 22.775K resistance, when parallel with the internal 164.2K resistor yields a 20K resistance, which can readily use the example program in the sample program without modification. However, you can also choose a slightly different resistor value and compute the parallel resistance, then enter the value into the Excel spreadsheet included with the example program.

Note that since the thermistor resistance value vs. temperature change is a non-linear function, you cannot simply use a formula to calculate the temperature from the voltage value. For better accuracy you need to use a look up table plus a linear interpolation technique to determine the temperature based on the ADC readings. The look up table and interpolation method can be implemented using TBASIC quite easily. For your convenience, we have provided a sample TBASIC program that you can download from the following web page:

http://www.tri-plc.com/appnotes/F-series/ThermistorSensorFPLC.zip (http://www.tri-plc.com/appnotes/F-series/ThermistorSensorFPLC.zip)

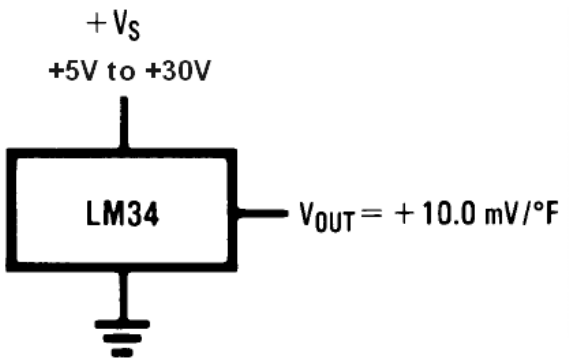
This example uses the R-T (Resistance-Temperature) graph of the Precon Type III thermistor to implement the temperature look up. We have provided an Excel file that computes the ADC reading vs ambient temperature for this thermistor type. The TBASIC program uses these ADC readings to determine the temperature.

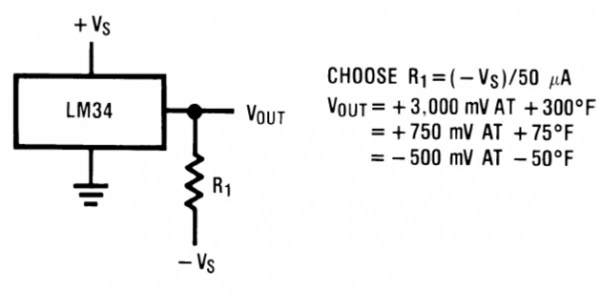
The sample program is structured such that the lookup table values are stored in the EEPROM and you can readily adapt it to other types of thermistors with a different R-T graph. The program only implements lookup for a temperature range of –10oF to 100oF, but you can also easily change the temperature range of interest.

Note:

You may also use a 100K thermistor to connect directly to the PLC's analog input and using its internal 164.2K impedance and the thermistor to form a voltage divider. We did not provide an example program for such a connection but you can write the program by following the example of 10K thermistor forming a voltage divider with a 20K resistor.

b) Using LM34 Semiconductor Sensor





Full-Range Fahrenheit Temperature Sensor

The LM34 is a wonderful, low cost semiconductor temperature sensor with a range of –50 oF to 300 oF. It is extremely easy to use for measuring temperature above 0 oF. You simply connect one pin to the positive voltage (+5 to +30V) and the other pin to 0V, and the signal pin will output a voltage that is directly proportional to the ambient temperature in oF. The output voltage is 10mV per degree F.

So at room temperature of 72 degrees F, the device will output a voltage of 72 x 0.01 = 0.72V. If you connect this to the PLC's analog input, you can obtain the temperature in degree F or degree C using the formula:

```
F = ADC(1)*500/4096  'in degree F
```

OR

```
      C = (F – 32)*5/9     'in degree C
```

Although another part number LM35 can output temperature in 10mV per degree C, the output falls into even lower ADC range for ambient temperature measurement since the same 72 degree F is only 22 degree C, which means LM35 only outputs 22 x 0.01 = 0.22V. Hence for better accuracy and resolution we recommend using LM34 instead of LM35 and if need be then convert it to degree C using TBASIC.

Another advantage for using LM34 instead of LM35 is that you can measure down to 0.oF (-17 degree C) without using a negative voltage source as shown in the circuit on the right in Figure 5.7.

c) Using Thermocouple

Thermocouples are very rugged devices that are widely used in the industry because of their stability, accuracy, and wide functional temperature range. They are commonly used in measuring temperature in ovens that may go up to several hundred degrees C.

However, thermocouple output signals are in the range of tens of microvolts to mille-volts, which is too small to be measured by the Wx100’s analog input directly. You will need a “signal conditioner” that can amplify the thermocouple output to 0-5V, which can then be connected to the PLC’s analog input.

d) Using PT100 Temperature Sensor

PT100 is a positive temperature coefficient thermistor that is made from platinum. It has the advantage of being very stable and highly accurate. It is usually connected to a signal conditioner in a balanced bridge configuration and the signal conditioner will convert temperature changes to 0-5V output for the PLC.

5.5 Analog Outputs

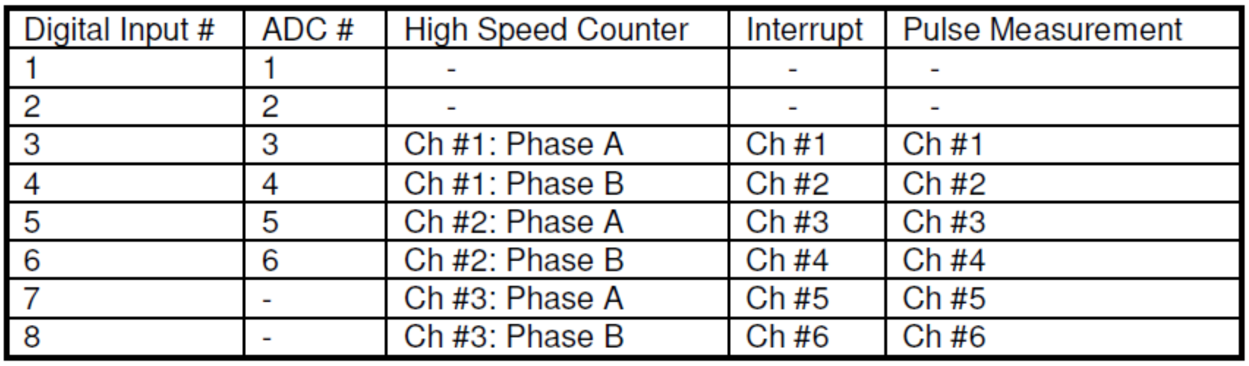
The basic Wx100 + Wx-TERM8 does not support any analog output. However, there are 6 channels of PWM outputs on the digital outputs which in many cases could be used to simulate analog output for control purposes.

The Wx-TERM16, a more powerful terminal boards for Wx100 PLC, supports 4 channel of 12-bit analog outputs. 2 of these analog outputs are 0-20mA and 2 are 0-5V. Please refer to Wx-TERM16 User’s Manual for more details.

Chapter 6 - Special Digital I/Os

6 of the first 8 ON/OFF inputs of the Wx100 can be configured as “special inputs” such as High Speed Counters, Interrupts and Pulse Measurement. All 6 of the digital outputs can also be configured as PWM, or stepper motor controller/driver pulse-outputs. If these special I/Os are not used, then they can be used as ordinary ON/OFF type I/O in the ladder diagram. The High Speed Counters and Pulse measurement inputs share physical inputs, but they can be used simultaneously as HSC and PMON. Note that if any other two special functions share the same I/O then only one of them can be active at any one time. The location of these special I/Os are tabulated as follows:

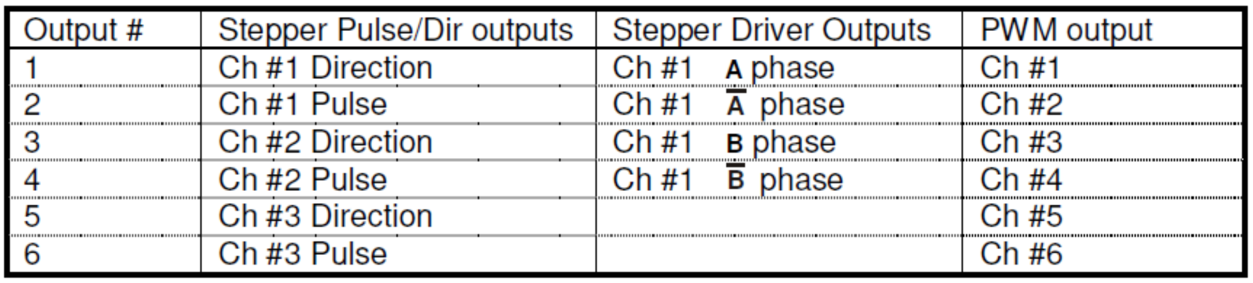
Special Inputs



Digital Input #	ADC #	High Speed Counter	Interrupt	Pulse Measurement
1	1	-	-	-
2	2	-	-	-
3	3	Ch #1: Phase A	Ch #1	Ch #1
4	4	Ch #1: Phase B	Ch #2	Ch #2
5	5	Ch #2: Phase A	Ch #3	Ch #3
6	6	Ch #2: Phase B	Ch #4	Ch #4
7	-	Ch #3: Phase A	Ch #5	Ch #5
8	-	Ch #3: Phase B	Ch #6	Ch #6

Note: inputs 3-8 can be used simultaneously as High Speed Counters, Pulse Measurement pins, and
interrupt inputs.

Special Outputs



Output #	Stepper Pulse/Dir outputs	Stepper Driver Outputs	PWM output
1	Ch #1 Direction	Ch #1 **A** phase	Ch #1
2	Ch #1 Pulse	Ch #1 $\overline{\mathbf{A}}$ phase	Ch #2
3	Ch #2 Direction	Ch #1 **B** phase	Ch #3
4	Ch #2 Pulse	Ch #1 $\overline{\mathbf{B}}$ phase	Ch #4
5	Ch #3 Direction		Ch #5
6	Ch #3 Pulse		Ch #6

These special I/Os therefore share the same electrical specifications as the ON/OFF type I/Os, which have already been described in the Chapter 1 – Installation Guide. We will describe each of these special I/Os in greater details in the following chapters.

Chapter 7 - High Speed Counters

Technical Specifications:

No. of Channels	3
Maximum pulse rate:	10KHz
Quadrature signal decoding	Automatic
Revelant TBASIC Commands	HSCDEF, HSCOFF, HSCPV[ ]

7.1 Introduction

Digital inputs #1 + 2, #3 + 4, and #5 + 6 form three channels of high speed counter inputs which can interfaced directly to a rotary encoder that produces "quadrature" outputs. A quadrature encoder produces two pulse trains at a 90o phase shift from each other as follows:

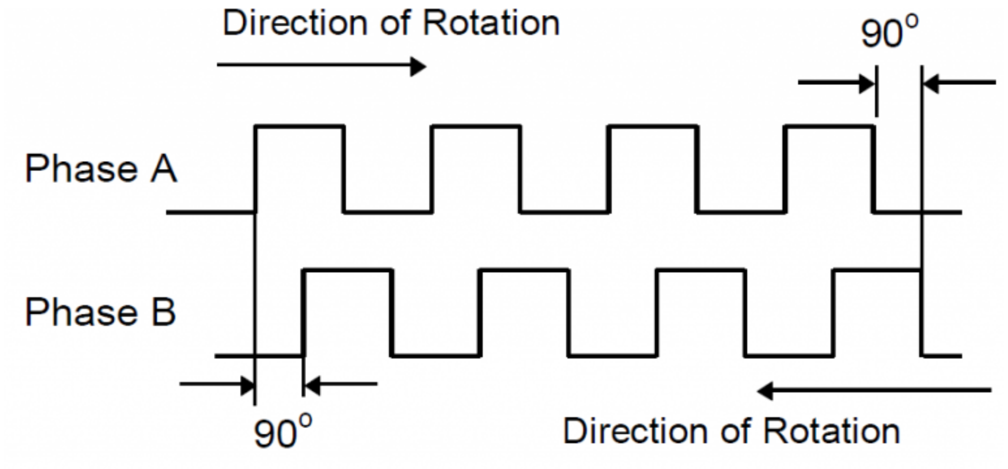


Figure 7.1

When the encoder shaft rotates in one direction, phase A leads phase B by 90 degrees. When the shaft rotates in the opposite direction, phase B will lead phase A by 90 degrees. The quadrature signals therefore provide an indication of the direction of rotation.

The Wx100 handles the quadrature signals as follows: if the pulse train arriving at input #3 leads the pulse train at input #4, the High Speed Counter (HSC) #1 increments on every pulse. If the pulse train arriving at input #3 lags the pulse trains at input #4, then the HSC #1 decrements. Note that if input #4 is OFF, then pulse trains arriving at input #3 are considered to lead the input #4 and HSC #1 will be incremented. Likewise if input #3 is OFF, then pulse trains arriving at input #4 will decrement HSC #1.

Inputs #5 and #6 form the inputs for High Speed Counter channel #2 and Inputs #7 and #8 forms the inputs for High Speed Counter channel #3. HSC#2 and HSC#3 operate in the same way as

HSC#1 described above.

The fact that the Wx100 automatically takes care of the direction of rotation of the quadrature encoder greatly simplifies the programmer’s task of handling high-speed encoder feedback. The HSCDEF statement can be used to define a CusFn to be executed when the HSC reaches a certain pre-defined value. Within this CusFn you can define the actions to be taken and define the next CusFn to be executed when the HSC reaches another value. Please note that the HSDDEF statement will also activate the Pulse Measurement hardware as described in the Pulse Measurement section.
A programming example of the HSC can be found in your iTRiLOGI program folder:

C:\TRiLOGI\TL6\usr\samples\HighSpeedCtr.PC6

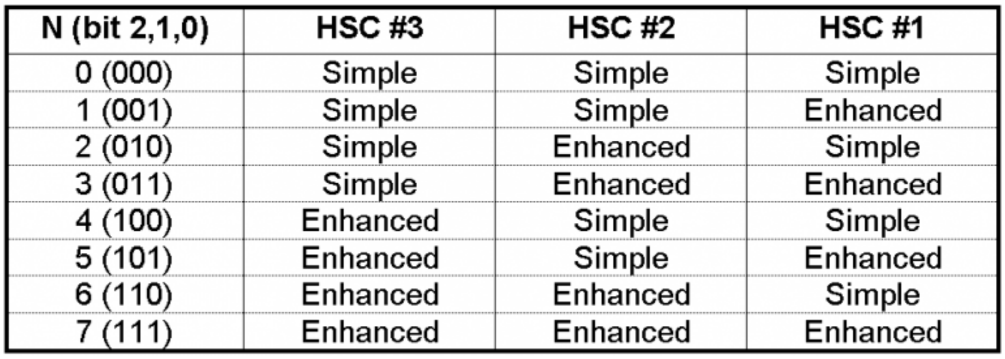
7.2 Enhanced Quadrature Decoding

The default method in which the PLC handles quadrature signals as described above is somewhat simplistic. It does not take into consideration the “jiggling” effect that occurs when the encoder is positioned at the transition edge of a phase. Mechanical vibration could cause multiple counts if the rotor shaft “jiggles” at the transition edge of the phase, resulting in multiple triggering of the counter. This simplistic implementation, however, does have the advantage that the HSC can also be used for single phase high-speed counting.

For the Wx100, an enhanced quadrature decoding routine is provided which will lock out multiple counting by examining the co-relationship between the two phases. You can configure the Wx100 to use the enhanced quadrature counting by using the SETSYSTEM command, as follows:

```
SETSYTEM 4, n
```

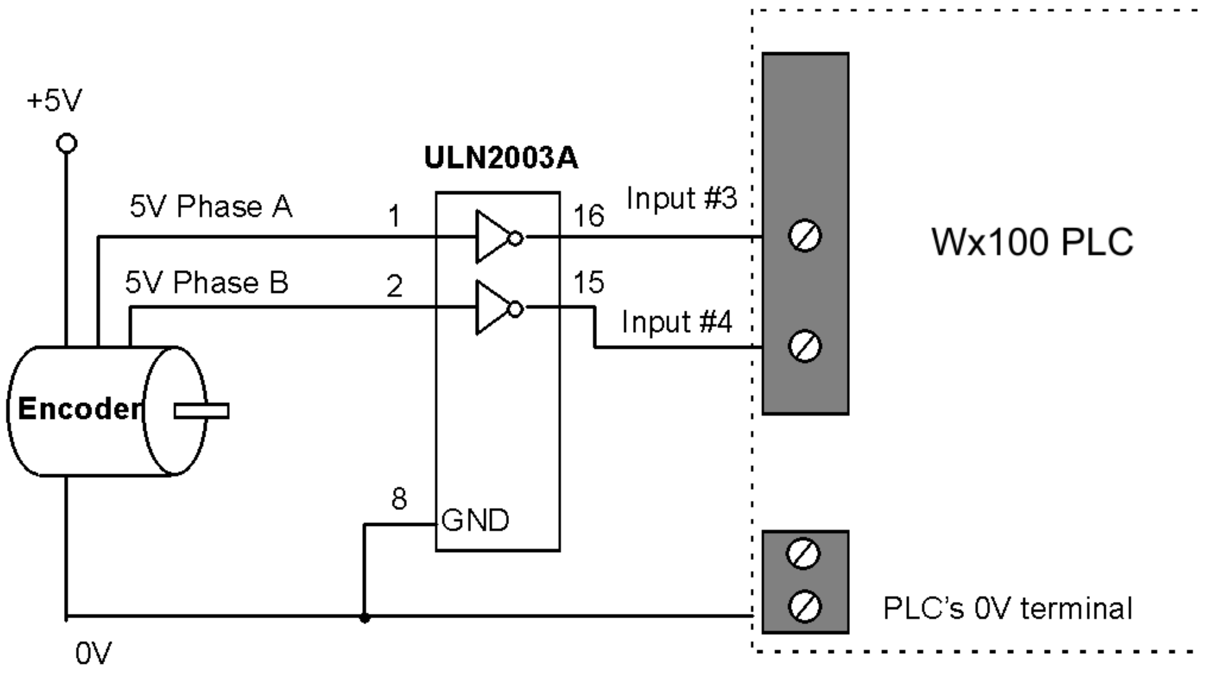
The value of n at bit 0, 1, and 2 respectively defines if the HSC channel 1, 2, and 3 is to run in “Simple” (when the bit is 0) or “Enhanced” (when the bit is 1) mode. As such:



N (bit 2,1,0)	HSC #3	HSC #2	HSC #1
0 (000)	Simple	Simple	Simple
1 (001)	Simple	Simple	Enhanced
2 (010)	Simple	Enhanced	Simple
3 (011)	Simple	Enhanced	Enhanced
4 (100)	Enhanced	Simple	Simple
5 (101)	Enhanced	Simple	Enhanced
6 (110)	Enhanced	Enhanced	Simple
7 (111)	Enhanced	Enhanced	Enhanced

7.3 Interfacing to 5V type Quadrature Encoder

If you have a choice, you should select an encoder that can produce 12V or 24V output pulses so that they can drive the inputs #3,4,5 ,6,7 or 8 directly. If you have a 5V type of encoder only, then you need to add a transistor driver to interface to the PLC's inputs. The simplest way is to use an IC driver ULN2003A or ULN2803A connected as shown in the figure. Since ULN2003/ULN2803A output is current-sinking type (NPN), you should connect V+ to the Pull-up input to turn the digital input into NPN type so that the driver chip output can sink current from the respective inputs.



Chapter 8 - Pulse Measurement

The Wx100 provides a very straightforward means to measure the **pulse width** (of the ON cycle), the **frequency**, or the **period** of a rectangular-wave pulse-train arriving at its Pulse Measurement (PM) inputs #3,4,5,6,7 & 8 (Which are mapped to digital inputs #1 to #6 – see below as well as Chapter 6).

Pulse Measurement Input Channel	Digital Input #	Frequency Range
1	3	1 to 50KHz
2	4	1 to 50KHz
3	5	1 to 50KHz
4	6	1 to 50KHz
5	7	1 to 50KHz
6	8	1 to 50KHz

8.1 Programming of PM Input

1. To use the PM input to measure pulse width or frequency, execute the PMON statement ONCE to configure the relevant input to become a pulse measurement input. You usually put the PMON statement in the init custom function and execute it with a “1st.Scan” pulse.

2. Thereafter the pulse width (in microseconds) or the pulse frequency (in Hz) can be easily obtained from the PULSEWIDTH(n) or PULSEFREQUENCY(n) functions. You can also obtain the pulse period (inverse of frequency) using the PULSEPERIOD function.

```
E.g.   A = PULSEWIDTH(1)
       B = PULSEPERIOD(1)
       C = PULSEFREQUNCY(1)
```

3. All PM inputs by default return the measured pulse width and pulse period in unit of microsecond. However, for those who desire better resolution, you can define PM #1 to #4 to return the measured pulse width and pulse period in 0.1 microsecond resolution by executing the following command once only during initialization:

```
SETSYSTEM 20, 1
```

4. Once the above statement is executed, if PUSLEWIDTH(1) to PULSEWIDTH(6) returns the value 1234 it means the measured pulse width is 123.4 microseconds. A sample program can also be found on your i-TRiLOGI installation folder at:

```
C:\TRiLOGI\TL6\usr\samples\PulseMeasurement.PC6
```

8.2 Applications

a) Measuring RPM Of A Motor

One useful application of the PM capability is to measure the speed of rotation (RPM) of a motor.

A simple optical sensor, coupled with a rotating disk with slots fitted to the shaft of a motor (See Figure 8.1) can be fabricated economically. When the motor turns, the sensor will generate a series of pulses. The frequency of this pulse train directly measures the rotational speed of the motor (RPM = Frequency x 60) and can be used to provide precise speed control.

Note that the above setup can also double as a low cost position-feedback encoder when used with the high speed counter, since the number of pulses counted can be used to determine the displacement. With the Wx100 PLC, the pulses can be both counted and measured **simultaneously** on the same input.

b) Measuring Transducer with VCO Outputs

Some transducers incorporate a Voltage-Controlled-Oscillator (VCO) type of output that represents the measured quantities in terms of varying frequency of the output waveform. Such transducers may be used conveniently by the Wx100 using the pulse measurement capability. However, the frequency of such signals should be below the maximum input pulse rate.

c) Measuring Transducer with PWM Outputs

Some transducers may output the readings of their measurands (the quantity that is being measured) in the form of “pulse-width modulated” outputs. This means that the transducer would send rectangular pulses with varying duty cycles that are proportional to the measured quantities. You can then easily use the PULSEWIDTH and PULSEPERIOD functions to compute the duty cycle of the incoming PWM pulses and readily convert it to the actual units of the measurands. The Wx100 PLC is able to measure with reasonable accuracy the pulse width of incoming pulses with frequency up to 50KHz .

8.3 HSC Frequency Measurement

For applications that require frequency measurement and pulse counting of the same signal, you only need to feed the pulse input into any pair of the inputs #1 & 2, or inputs #3 & 4, or inputs #5 & 6 and define it as a High Speed Counter (see Chapter 7). This is because an input pin that has been defined as an HSC will automatically be enabled for pulse measurement.

In other words, 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, you only need to execute the HSCDEF. The HSCDEF function will automatically start the Pulse Measurement hardware so it is not necessary to use the PMON function. However, if you use only the PMON function, it would not enable the HSC function. However, if you execute the HSCDEF function and then execute the

PMON function, the HSC will be disabled, even though it was previously enabled.

Chapter 9 - INTERRUPTS

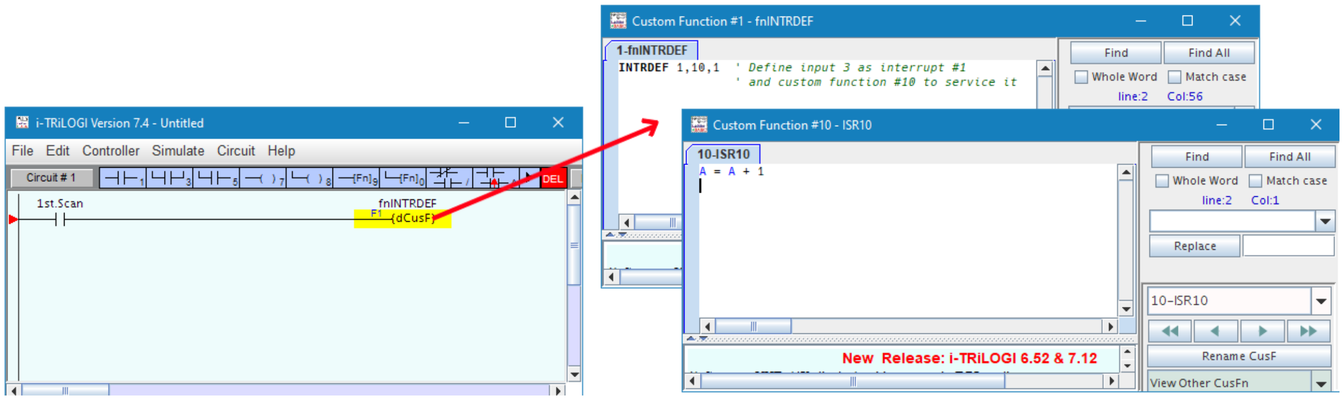
9.1 Input Interrupts

During normal PLC ladder program execution, the CPU scans the entire ladder program starting from the first element, progressively solving the logic equation at each circuit until it reaches the last element. After which it will update the physical Inputs and Outputs (I/O) at the end of the scan. Hence, the location of a logic element within the ladder diagram is important because of this sequential nature of the program execution.
When scanning the ladder program, the CPU uses some internal memory variables to represent the logic states of the inputs obtained during the last I/O refresh cycle. Likewise, any changes to the logic state of the outputs are temporarily stored in the output memory variable (not the actual output pin) and will only be updated to the physical output during the next I/O refresh.

You can see that the CPU will only notice any change to the input logic state when it has completed the current scan and starts to refresh its input variables. The input logic state must also persist for at least one scan time to be recognized by the CPU. In some situations this may not be desirable because any response to the event will take at least one scan time or more.

An interrupt input, on the other hand, may occur randomly and the CPU will have to suspend whatever it is doing as soon as it can and start “servicing” the interrupt. Hence, the CPU responds much faster to an interrupt input. In addition, interrupts are “edge-triggered”, meaning that the interrupt condition occurs when the input either changes from ON to OFF or from OFF to ON. Consequently, the input logic state need not persist for longer than the logic scan time for it to be recognized by the CPU.

The Wx100 PLC supports up to 10 interrupt inputs: Any one or all of digital inputs #1 to #6 and #9 to #12 can be defined as interrupt inputs using the INTRDEF statement. The Interrupt inputs may also be defined as either rising-edge triggered (input goes from OFF to ON) or falling-edge triggered (input goes from ON to OFF). When the defined edges occur, the defined CusFn will be immediately executed irrespective of the current state of execution of the ladder program. A simple interrupt test function is as follow:


i-TRiLOGI Version 7.4 - Untitled
File Edit Controller Simulate Circuit Help
Circuit # 1
1st.Scan
fnINTRDEF
dCusF
Custom Function #1 - fnINTRDEF
1-fnINTRDEF
Custom Function #10 - ISR10
10-ISR10
Find
Find All
Whole Word
Match case
Replace
Rename CusF
View Other CusFn
New Release: i-TRiLOGI 6.52 & 7.12

```
INTRDEF 1,10,1  ' Define input 3 as interrupt #1
                ' and custom function #10 to service it
```

```
A = A + 1
```

Variable A will be incremented on every rising edge sensed on input #3 (mapped to interrupt input #1)

Note:

Since inputs 1 to 6 can also be used as other special inputs such as High Speed Counter (HSC) inputs and/or Pulse Measurement (PM) as described in Chapter 6, 7 and 8, if these inputs are defined as interrupts using the INTRDEF statement, then they will lose their other special function. I.e., they can only be defined either as a HSC/PM or as an interrupt input and not both. Inputs 9 to 12, on the other hand, are not shared with other special functions. Hence, these can be used either as ordinary digital inputs or as interrupt inputs.

9.2 Periodic Timer Interrupt (PTI)

The Periodic Timer Interrupt (PTI – not available on T100M+ PLC) lets you define a custom function that will be executed by the CPU precisely every x number of milliseconds (ms). The syntax for setting up a PTI is as follow:

```
INTRDEF 18, cfnum, x    ' Interrupt 18 is reserved for PTI
      cfnum - custom function number to execute when PTI event takes place.
          x - The period in number of milliseconds between two PTI events.

E.g. INTRDEF 18, 101, 15 ‘ call function #101 every 15 ms.
```

The Periodic Timer Interrupt runs independently of the ladder logic and its execution is therefore not affected by the total PLC program scan time.

When the PTI timer times up, the CPU will suspend the execution of the ladder logic or a (non-interrupt) TBASIC function and immediately calls up the custom function defined by the INTRDEF 18 statement. However, if the CPU is currently executing a user-interrupt service routine (e.g. an input interrupt or HSC interrupt), then the CPU will have to complete the current interrupt service routine before it will run the PTI interrupt function.

Notes:

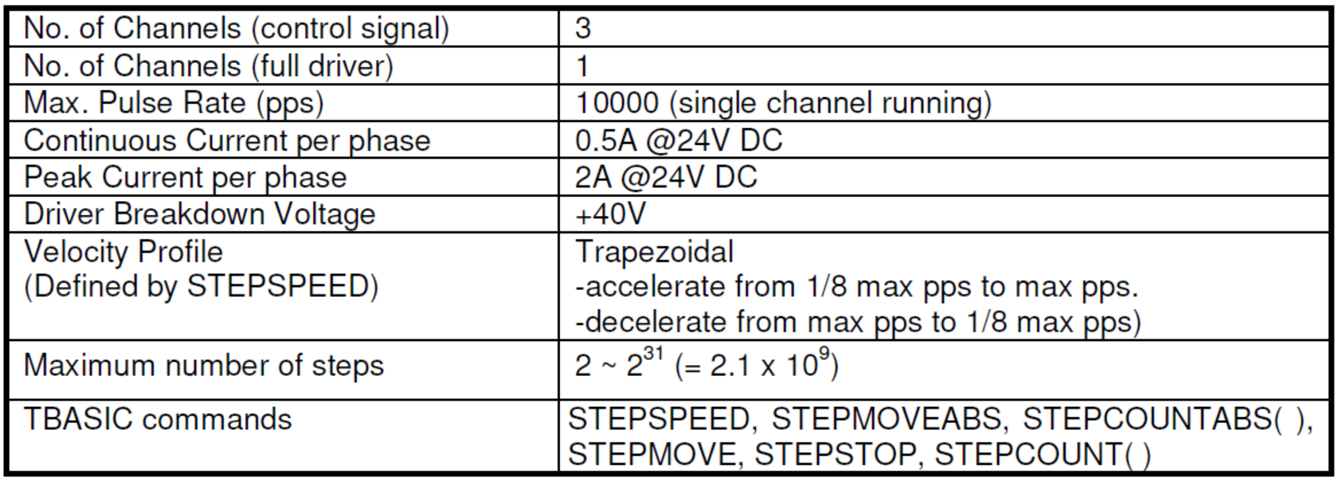
1) Limit the use of PTI only for critical code that requires precise timing between two events. Program bugs that occur due to problems in the PTI interrupt routine may be quite hard to debug.

2) For normal periodic routines, such as checking for temperature or checking serial port for incoming bar code data every few seconds, it is better to use the system clock pulses e.g. “Clk:1.0s” to trigger a {dCusF}.

3) Always try to keep your interrupt service routine short and ensure that it will not end up in an endless loop. The TBASIC custom function execution time should be much shorter than the period of the PTI events. Otherwise, you may find that the CPU will be spending most of its time servicing the PTI interrupt routine, leaving very little time for scanning the ladder program, and that will have an adverse impact on the CPU performance.

Chapter 10 - Stepper Motor Control

10.1 Technical Specifications



No. of Channels (control signal)	3
No. of Channels (full driver)	1
Max. Pulse Rate (pps)	10000 (single channel running)
Continuous Current per phase	0.5A @24V DC
Peak Current per phase	2A @24V DC
Driver Breakdown Voltage	+40V
Velocity Profile (Defined by STEPSPEED)	Trapezoidal -accelerate from 1/8 max pps to max pps. -decelerate from max pps to 1/8 max pps)
Maximum number of steps	2 ~ 2^{31} (= 2.1 x 10^9)
TBASIC commands	STEPSPEED, STEPMOVEABS, STEPCOUNTABS( ), STEPMOVE, STEPSTOP, STEPCOUNT( )

It is essential to understand the difference between a stepper motor “Controller” and a stepper motor “Driver”. A stepper motor “Driver” comprises the power electronics circuitry that provides the voltage, current, and phase rotation to the stepper motor coils. A stepper controller on the other hand, only supply the “direction” and “pulse” signals to an external stepper motor driver to actually rotate the stepper motors.

The Wx100 PLC supports 3 channels of stepper controller only, or 1 channel of stepper motor driver + 1 channel or stepper controller. i.e A small stepper motor can be driven **directly** by the PLC, or the PLC can supply up to 3 sets of control signals to an external stepper motor driver.

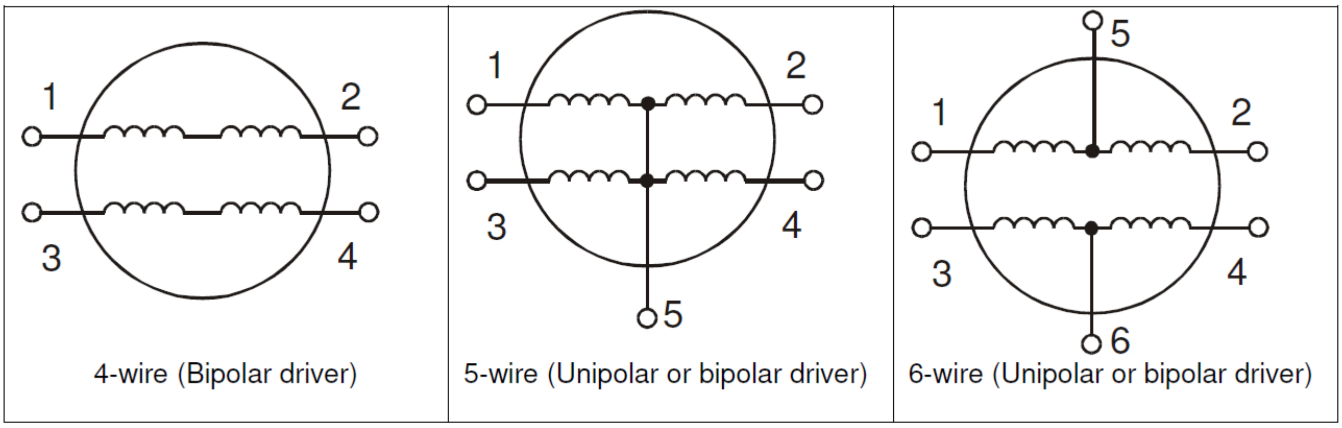
The PLC as a stepper driver takes up 4 out of its 6 high speed digital outputs and is limited by its

0.5A current drive limit. It is hence suitable mainly for driving small stepper motor where maximum coil current < 0.5A. To control higher power stepper motors, we recommend using external stepper motor drivers that also provide other important functions such as thermal management, over current protection etc.

10.2 Stepper Motor Driver

The Wx100 can drive 1 small stepper motors directly, using digital outputs #1, 2, 3 & 4, All these outputs are capable of driving stepper motors of up to 0.5A per phase. Note that any outputs that have been defined for use with a particular stepper channel would no longer be available as regular digital outputs in the ladder or TBASIC program. Please refer to Chapter 6 for a definition of the list of special digital I/Os.

a) Stepper Motor Wiring Types


1
2
3
4
4-wire (Bipolar driver)
1
2
3
4
5
5-wire (Unipolar or bipolar driver)
5
1
2
3
4
6
6-wire (Unipolar or bipolar driver)

The most commonly available stepper motors today are either 4-wire, 5-wire or 6 wires types and their windings are as shown in Figure 10.1

- 4-wire stepper motors require drivers that can reverse current flow from coil terminal 1 to 2 or from 2 to1, as well as from terminal 3 to 4 or from 4 to 3. These kinds of drivers are known as "bipolar drivers".
- 5-wire stepper motors typically connect the common terminal (5) to the power (e.g. +24V DC) and 4 current sink drivers are connected to terminals 1 to 4 to sink the current to 0V. The sequence in which the four coils are activated controls the rotation of the stepper motor. These kinds of drivers are known as "unipolar" drivers.
- 6-wire stepper motors are the most flexible and may be connected either to bipolar drivers (leaving terminals 5 and 6 unconnected) or to unipolar drivers (connecting terminals 5 & 6 together to "+" terminal).

Since all the Wx100 digital outputs are NPN (current sink) type, only 5- or 6-wire type stepper motors can be used.

If you have a stepper motor with 6 wires (most common) but you do not have the wiring diagram, you may be able to figure out which wire is which terminal by using a multi-meter to measure the resistance between coils. E.g. the resistance between terminals 1 and 2 will be twice the resistance between terminals 1 and 5 or between 2 and 5, likewise for terminals 3,4 and 6. There will be no connectivity between terminal 1 and 3 and between terminal 2 and 4. It will however be more tricky to figure out the actual wiring of a 5-wire stepper motor since it is difficult to figure out which is the coil 1-2 and which is the coil 3-4, so some trial and error may be required to get the motor working.

b) Damping Circuits (Inductive Bypass)

Since the stepper motor windings are all inductors, it is important to connect damping diodes across each winding to absorb the inductive kicks produced when the windings are turned OFF. There are several ways to connect the damping circuit; the simplest way is to connect a diode from the PLC's output to the stepper motor power supply V+, as shown below in Figure 10.2.

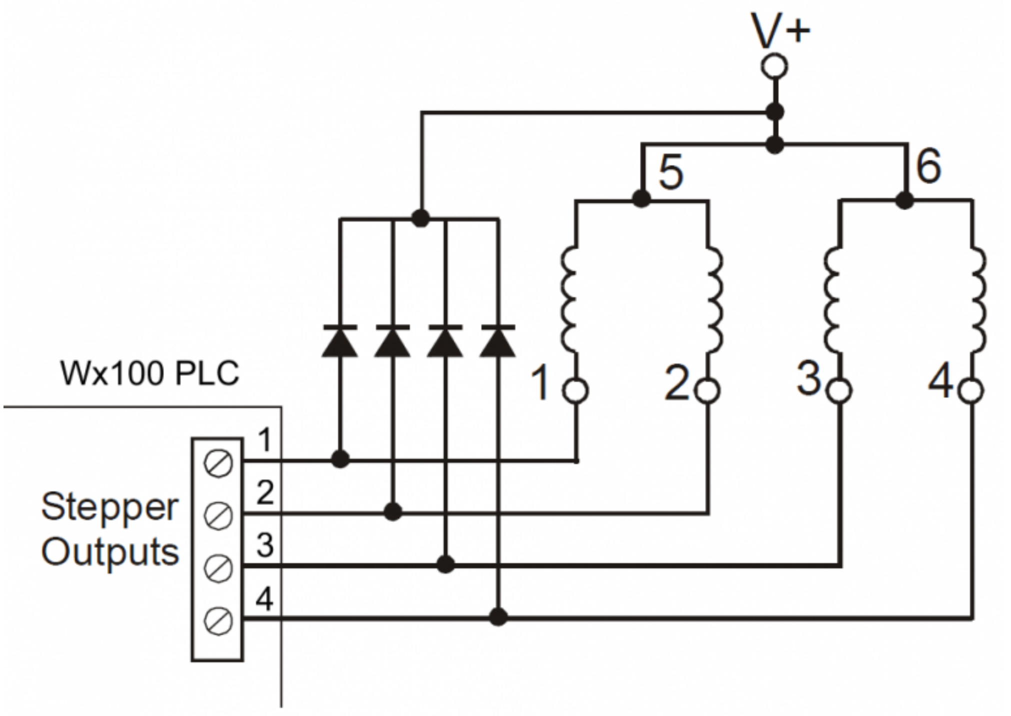


Figure 10.2

The above circuit works best when the stepper motor power is the same voltage as or is a slightly higher voltage than the PLC's power supply. (Note: The maximum stepper motor's power supply voltage must not exceed the PLC's stepper output driver breakdown voltage of 40V)

If the stepper motor's power supply were lower than the PLC's power supply, then connecting the output shown in Figure 10.2 would result in the PLC's output LED to light up, even when the output driver is off. This is because the PLC's output LED is tied to the PLC's power supply and a lower voltage at the load's power supply would cause current to flow out of the PLC's output terminal into the load's power supply and therefore turning on the output LEDs. Although this should not affect the driver's function it can be misleading to the user as to the PLC's output state. Hence, we recommend connecting a diode in series with each of the PLC's outputs to block such reverse current flow:

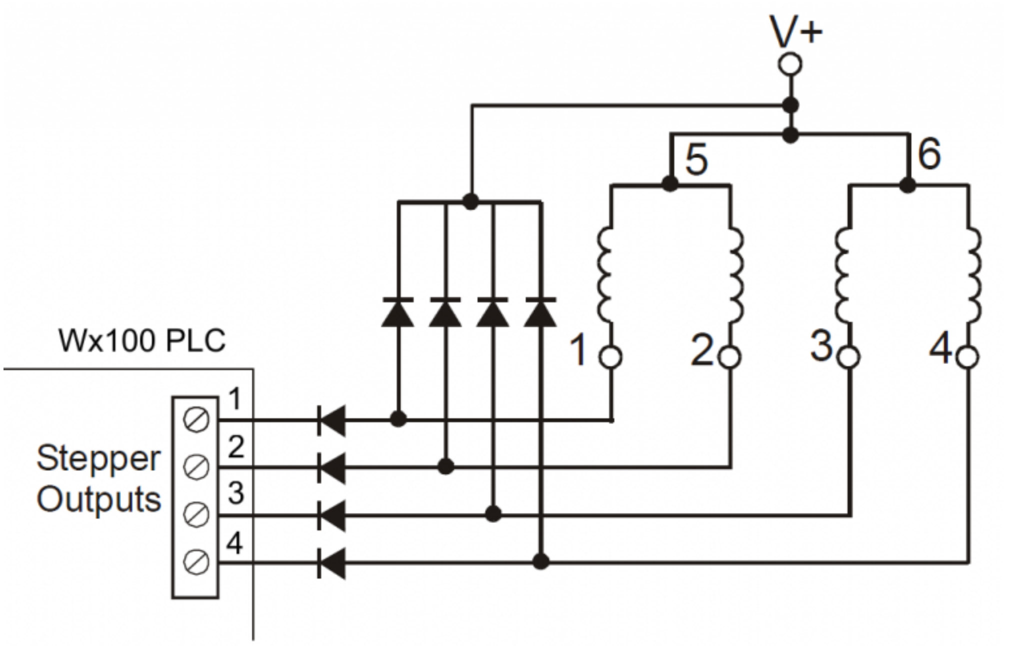


Figure 10.3 – If V+ is lower voltage than PLC's power supply voltage.

10.3 Stepper Motor Controller

Since the built-in driver use up 4 of the 6 outputs available on Wx100 with Wx-TERM8, you may need to preserve the outputs for other purposes, or or if the built-in stepper driver is not suitable for your application (e.g. if you need micro-stepping or you need to drive a higher power stepper motor), then you can use an external stepper motor driver and use the PLC to send direction and output pulses to the stepper motor driver.

When configured as a Stepper-Motor Controller, the PLC would generate the required number of "pulses" and sets the direction signal according to the defined acceleration and maximum pulsing rate specified by "STEPSPEED" and "STEPMOVE" commands. The "pulse" and "direction" outputs are not meant to be connected directly to the stepper motor. Instead, you will need a stepper motor "driver", which you can buy from the motor vendor. Depending on the power output, the number of phases of the stepper motor, and whether you need micro-stepping, the driver can vary in size and cost. Most stepper motor drivers have opto-isolated inputs which accept a direction signal and stepping-pulse signal from the "Stepper Motor Controller". In this case the Wx100 is the "Stepper Motor Controller" which will supply the required pulse and direction-select signals to the driver. A Wx100 PLC can control up to 3 stepper motor drivers using all its 6 digital outputs.

Note that the digital outputs #1, #3, and #5 automatically become the direction-select signals for the Stepper controller channels #1, #2, and #3, respectively, when the stepper controllers are being used. The direction pin is turned ON when the motor moves in the negative direction and turned OFF when the stepper motor moves in the positive direction. The STEPMOVEABS command makes it extremely simple to position the motor at an absolute location, while the STEPMOVE command lets you implement incremental moves in either direction for each channel.

Interfacing to 5V Stepper Motor Driver Inputs

Some stepper motor drivers accept only 5V signals from the stepper motor controller. In such a case, you need to determine whether the driver's inputs are opto-isolated. If they are, then you can simply connect a 2.2K current limiting resistor in series with the path from the PLC's output to the driver's inputs, as shown in Figure 10.4.

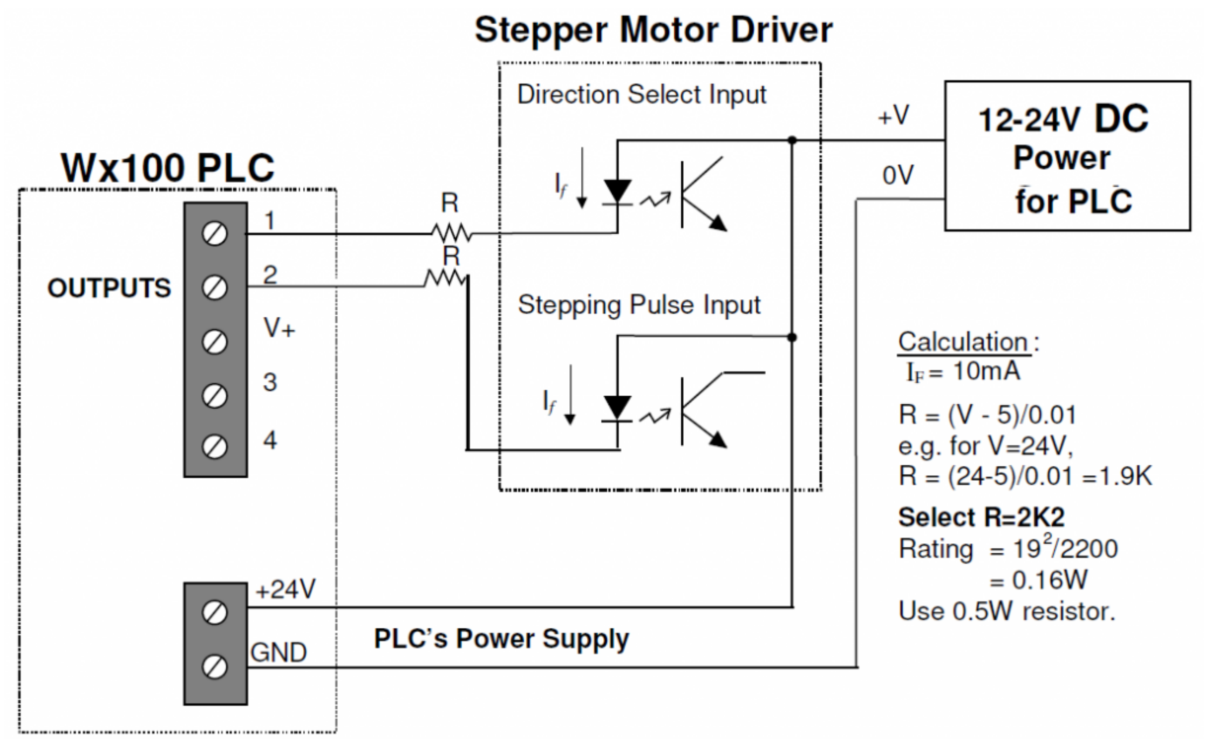


Figure 10.4

However, if the stepper motor driver input is only 5V CMOS level and non opto-isolated, then you need to convert the 12-24V outputs to 5V. This can be achieved using a low cost transistor such as a 2N4403. A better way is to use an opto-isolator with a logic level output, as shown in Figure 10.4. This provides a galvanic isolation between the PLC and the stepper motor driver.

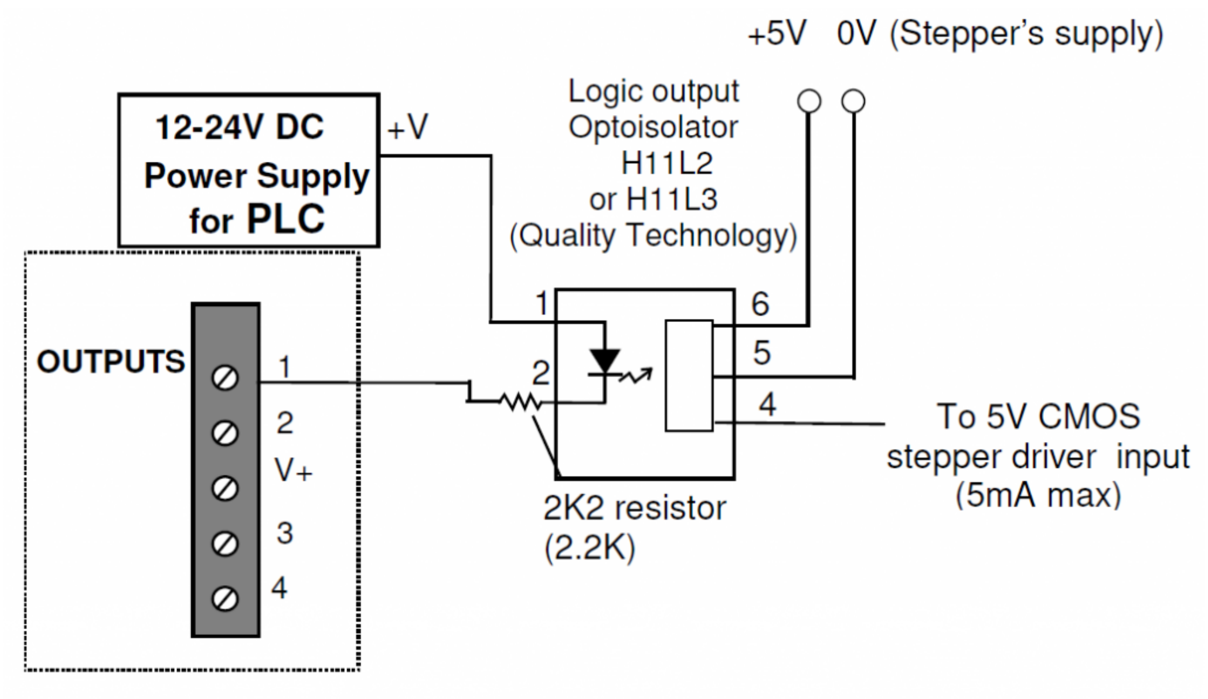

+5V
0V (Stepper's supply)
12-24V DC Power Supply for PLC
+V
Logic output Optoisolator H11L2 or H11L3 (Quality Technology)
OUTPUTS
1
2
V+
3
4
1
2
6
5
4
2K2 resistor (2.2K)
To 5V CMOS stepper driver input (5mA max)

Figure 10.5 Conversion of Wx100 outputs to 5V logic level

10.4 Programming Stepper

a) Introduction

Any of the 3 stepper-motor channels can be configured as either a Controller (direction/pulse) or a Driver; and as a Driver, can drive in FULL STEP mode or HALF STEP mode. The stepper channels are controlled by the PLC program using the "STEPMOVE" and "STEPMOVEABS" commands. These commands have the same parameters as they did when they were used on M-Series PLCs. For example,

```
STEPMOVE ch, count, r
STEPMOVEABS ch, position, r
```

The ch parameter is the channel. This is how the PLC knows which stepper motor channel to turn on. Now this parameter is also used to set the stepper motor channel to controller mode, full-step driver mode, or half-step driver mode.

To set the stepper motor channel to controller mode, the ch parameter should be a "1", "2", or "3".

To set the stepper motor channel to driver mode, the ch parameter needs to have a "1" prefixed to it for full-step mode and a "2" prefixed for half-step mode.

b) Full-Step Driver Mode:

```
STEPMOVE 11, 100, 1 'Stepper channel 1 will be driven 100 full steps
STEPMOVE 12, 100, 1 'Stepper channel 2 will be driven 100 full steps
STEPMOVE 13, 100, 1 'Stepper channel 3 will be driven 100 full steps
```

c) Half-Step Driver Mode:

```
STEPMOVE 21, 100, 1 'Stepper channel 1 will be driven 100 half steps
STEPMOVE 22, 100, 1 'Stepper channel 2 will be driven 100 half steps
STEPMOVE 23, 100, 1 'Stepper channel 3 will be driven 100 half steps
```

The same format should be used for STEPMOVEABS, except that the count parameter is changed to the position parameter (more details provided in the Programmers Reference manual and also in the program examples section of this chapter).

d) Setting the Acceleration Properties

Whether the stepper motor channels are being used to control a stepper motor driver or to drive the motor directly, the STEPSPEED command must be executed first in order to define the acceleration settings. Once this command has been executed, the acceleration settings will not change until another STEPSPEED command is executed with different settings or if the PLC is powered down. If the PLC is powered down, the STEPSPEED command will need to be executed again before the stepper motor channels can be used. However, if a software reset executed (from online monitoring or within the program), the STEPSPEED command does not need to be re-executed.

```
STEPSPEED ch, pps, acc
```

The *ch* parameter should be a value between 1 and 3. Speed, *pps*, is based on the no. of pulses per second (*pps*) output by the pulse generator. The *pps* parameter should be set to a value between 1 and 10000 (max rated *pps* for the Wx100). The acceleration, *acc*, determines the total number of steps taken to reach full speed from a standstill and the number of steps from full speed to a complete stop. The stepper motor controller calculates and performs the speed trajectory according to these parameters when the STEPMOVE or STEPMOVEABS commands are executed.
To set stepper motor channel #3 to a speed of 100 *pps* in 50 steps, the command would be as follows:

```
STEPSPEED 3, 100, 50
```

This would be equivalent to an acceleration of 100 pulses/s2, which can be calculated using the following expression:

```
A = V^2/2S = 100^2/2*50 = 100pps^2
```

10.4.1 STEPMOVE

Once the STEPSPEED command is executed, the STEPMOVE command can be used to move the stepper motor forwards or backwards. The STEPMOVE command is also used to specify whether the PLC is a motor controller, a full-step motor driver, or a half-step motor driver.

```
STEPMOVE ch, count, r
```

The *ch* parameter specifies which stepper motor channel is being used and whether it is being used as a controller, full-step driver, or half-step driver.

The *count* parameter specifies how many pulses the motor will move. If the motor were in half-step driver mode, then count would be in half steps.

The *r* parameter specifies which internal relay will be activated once the motor has moved count number of steps. This relay would be cleared when the STEPMOVE command is executed in case it was already activated.

a) Example: PLC as a Motor Controller

In this example, the PLC would be a controller and would send the control pulse and direction pulse to a driver board.

Moving Forwards

```
STEPMOVE 1, 500, 101 ‘ channel 1 would send 500 pulses to a driver and then turn on rel
ay 101
STEPMOVE 2, 750, 102 ‘ channel 2 would send 750 pulses to a driver and then turn on rel
ay 102
STEPMOVE 3, 200, 103 ‘ channel 3 would send 200 pulses to a driver and then turn on rel
ay 103
```

Moving Backwards

```
STEPMOVE 1, -500, 101 ‘ channel 1 would send 500 pulses to a driver and then turn on re
lay 101
STEPMOVE 2, -750, 102 ‘ channel 2 would send 750 pulses to a driver and then turn on re
lay 102
STEPMOVE 3, -200, 103 ‘ channel 3 would send 200 pulses to a driver and then turn on re
lay 103
```

b) Example: PLC as a Full-Step Motor Driver

Moving Forwards

```
STEPMOVE 11,8000,101 ‘ channel 1 would send 8000 pulses to a motor and then turn on rel
ay 101
STEPMOVE 12,50,102 ‘ channel 2 would send 50 pulses to a motor and then turn on relay 1
02
STEPMOVE 13,900,103 ‘ channel 3 would send 900 pulses to a driver and then turn on rela
y 103
```

Moving Backwards

```
STEPMOVE 11,-300,101 ‘ channel 1 would send 300 pulses to a motor and then turn on rela
y 101
STEPMOVE 12,-1000,102 ‘ channel 2 would send 1000 pulses to a motor and then turn on re
lay 102
STEPMOVE 13,-7500,103 ‘ channel 3 would send 7500 pulses to a motor and then turn on re
lay 103
```

c) Example: PLC as a Half-Step Motor Driver

Moving Forwards

```
STEPMOVE 21,8000,101 ‘ channel 1 to send 8000 forward pulses to motor and turn on relay
101
STEPMOVE 22,50,102 ‘ channel 2 to send 50 forward pulses to a motor and turn on relay 1
02
STEPMOVE 23,900,103 ‘ channel 3 to send 900 forward pulses to motor and turn on relay 1
03
```

Moving Backwards

```
STEPMOVE 21,-300,101 ‘ channel 1 to send 300 backward pulses to motor and turn on relay
101
STEPMOVE 22,-1000,102 ‘ channel 2 to send 1000 backward pulses to motor and turn on rel
ay 102
STEPMOVE 23,-7500,103 ‘ channel 3 to send 7500 backward pulses to motor and turn on rel
ay 103
```

10.4.2 STEPMOVEABS

This command allows you to move the stepper motor # ch to an absolute position indicated by the position parameter. 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 ± 2 x 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 the 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 the 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. The STEPMOVEABS command is also used to specify whether the PLC is a motor controller, a full-step motor driver, or a half-step motor driver.

```
STEPMOVEABS ch, position, r
```

The *ch* parameter would specify which stepper motor channel is being used and whether it is being used as a controller, a full-step driver, or a half-step driver.

The *position* parameter specifies how many pulses the motor will move relative to its home position. If the motor were in half-step driver mode, then position would be in half steps.

The *r* parameter specifies which internal relay will be activated once the motor has moved to its new position. This relay would be cleared when the STEPMOVEABS command is executed in case it was already activated.

a) Example: PLC as a Motor Controller

this example, the PLC would be a controller and would send the control pulse and direction pulse to a driver board.

```
STEPMOVEABS 1,500,101 ‘ Stepper #1 to move fwd 500 steps from home and turn on relay 10
1
STEPMOVEABS 2,-75,102 ‘ Stepper #2 to move back 75 steps from home and turn on relay 10
2
STEPMOVEABS 3,200,103 ‘ Stepper #3 to move fwd 200 steps from home and turn on relay 10
3
```

b) Example: PLC as a Full-Step Motor Driver

```
STEPMOVEABS 11,300,101 ‘ Stepper #1 to move fwd 300 pulses from home and turn on relay
101
STEPMOVEABS 12,100,102 ‘ Stepper #2 to move fwd 100 pulses from home and turn on relay
102
STEPMOVEABS 13,-90,103 ‘ Stepper #3 to move back 90 pulses from home and turn on relay
103
```

c) Example: PLC as a Half-Step Motor Driver

```
STEPMOVEABS 21,300,101 ‘ Stepper #1 to move fwd 300 pulses from home and turn on relay
101
STEPMOVEABS 22,-10,102 ‘ Stepper #2 to move back 10 pulses from home and turn on relay
102
STEPMOVEABS 23,50,103 ‘ Stepper #3 to move fwd 50 pulses from home and turn on relay 10
3
```

10.4.2 Stepper Demo Program

A demo program for programming the Stepper Motor Controller and Driver:

```
    "Stepper Motor - F2424.PC6"
```

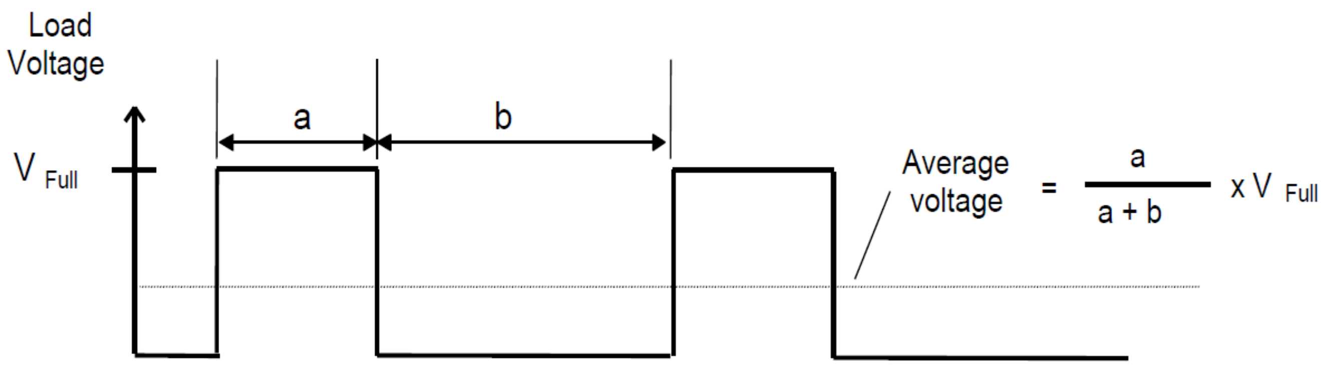
can be found in the FPLCsamples.zip file which can be downloaded from:

```
    http://www.tri-plc.com/trilogi/FPLCsamples.zip (http://www.tri-plc.com/trilogi/FPLC
samples.zip)
```

Chapter 11 to 20

Chapter 11 - PWM Outputs

Pulse-Width Modulation (PWM) is a highly efficient and convenient way of controlling output voltage to devices with large time constants, such as controlling the speed of a DC motor, the power to a heating element, or the position of a proportional valve. The PWM works by first turning on the output to full voltage for a short while and then shutting it off for another short while and then turning it on again, and so on, in consistent and accurate time intervals. This can be illustrated with the following diagram:

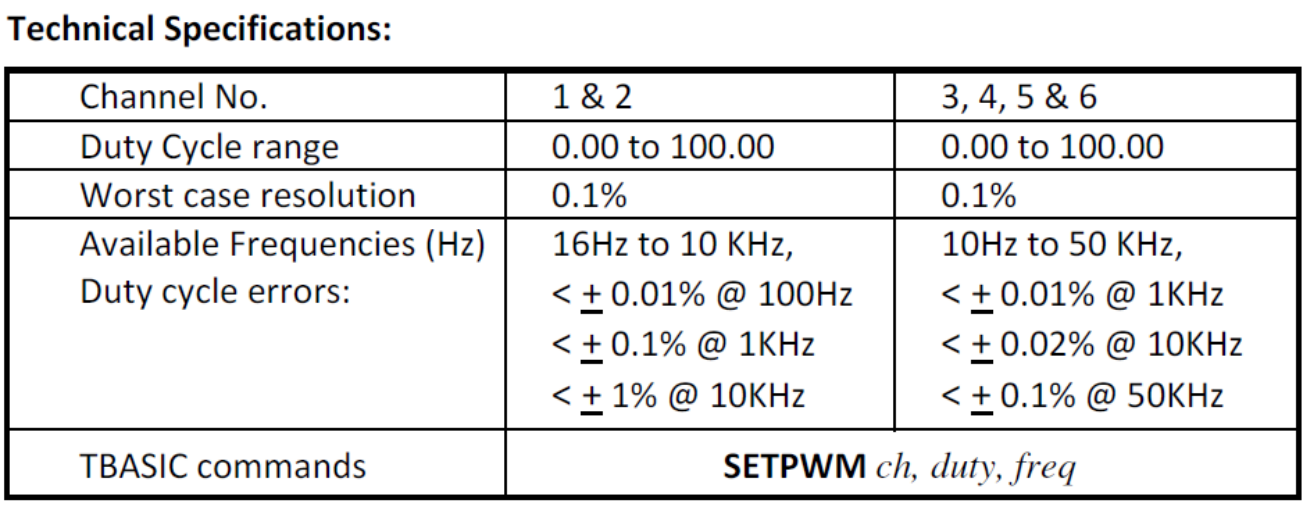


The average voltage seen by the load is determined by the “duty cycle” of the PWM waveform. The duty
cycle is defined as follow:

```
Duty Cycle =  a / (a + b) x 100%
Period = (a + b)
Frequency = 1/period Hz
```

Average voltage = % duty cycle multiplied by the full load voltage V_{Full}. Since the voltage applied to the load is either “Fully ON” or “Fully OFF”, it is highly efficient because the switching transistors are working in their saturated and cut-off region and dissipate very little power when it is fully turned ON.

11.1 PWM Specifications



Technical Specifications:

Channel No.	1 & 2	3, 4, 5 & 6
Duty Cycle range	0.00 to 100.00	0.00 to 100.00
Worst case resolution	0.1%	0.1%
Available Frequencies (Hz) Duty cycle errors:	16Hz to 10 KHz, < ± 0.01% @ 100Hz < ± 0.1% @ 1KHz < ± 1% @ 10KHz	10Hz to 50 KHz, < ± 0.01% @ 1KHz < ± 0.02% @ 10KHz < ± 0.1% @ 50KHz
TBASIC commands	**SETPWM** *ch, duty, freq*	

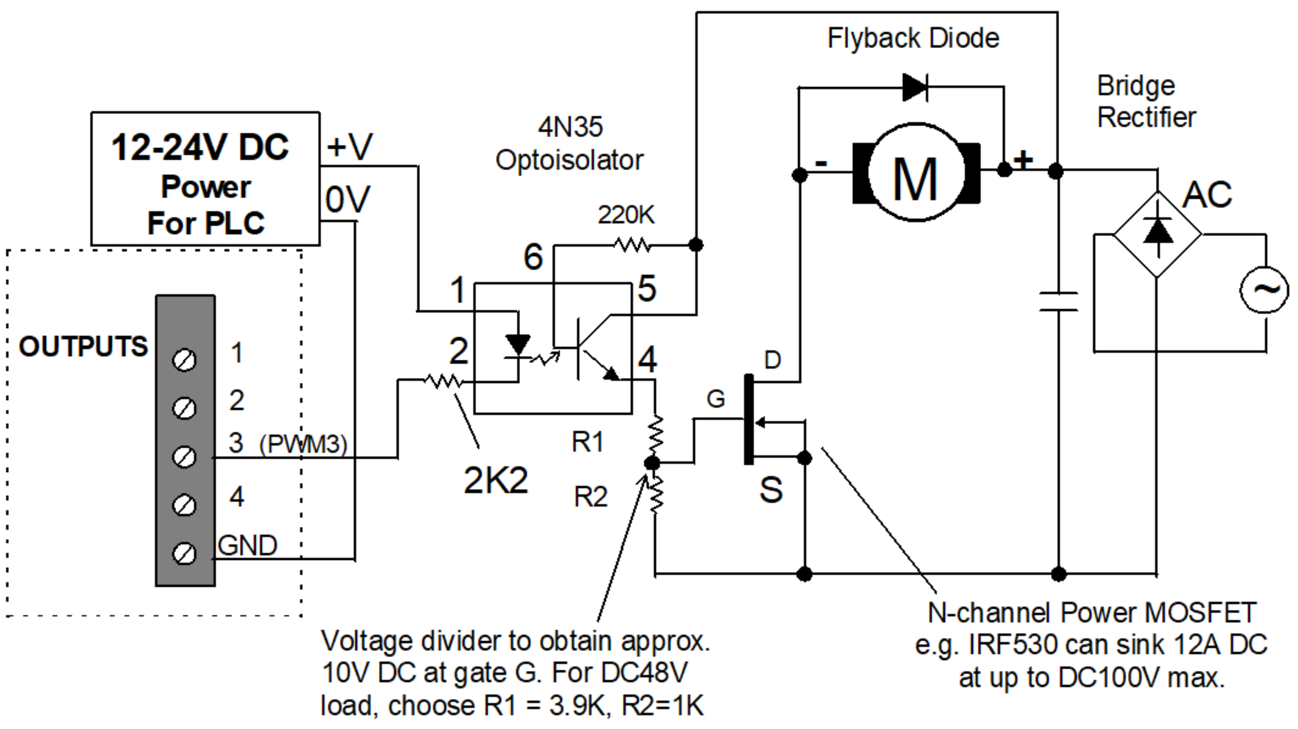
The TBASIC **SETPWM** statement controls the frequency and duty-cycle settings of the PWM channel. The Wx100 PLC features six channels of PWM on its digital outputs #1 (PWM channel #1) to output #6 (PWM channel #6).

The PWM channels can generate pulses with frequency ranging from 10Hz to 50KHz. At the lower frequency range, the output frequency can be extremely accurate (less than 0.01% error). Even at 10KHz the output frequency error is less than 1% for channel 1 & 2, and < 0.1% for channel 3 to 6. This makes it possible to use the PWM channels to generate square wave pulses of a certain frequency.

Usually it is better to select as high a frequency as possible because the resulting effect is smoother at higher frequencies. However, some systems may not respond properly if the PWM frequency is too high, in such cases a lower frequency should be selected.

Since all 6 PWM outputs are high voltage, high current outputs (24V, 0.5A) they can be used to directly control the speed of a small DC motor. They can also directly drive proportional (variable position) valves whose opening is dependent on the applied voltage. **Note:** When using the PWM output to drive a motor or solenoid valve, please take note of the need to add a bypass diode to absorb the inductive kick that will occur when the output current to the load is turned OFF, as mentioned in Section 1.7c

11.2 Increasing PWM Current



Flyback Diode
Bridge Rectifier
4N35 Optoisolator
12-24V DC Power For PLC
+V
0V
220K
6
1
5
2
4
OUTPUTS
1
2
3 (PWM3)
4
GND
2K2
R1
R2
G
D
S
M
-
+
AC
Voltage divider to obtain approx. 10V DC at gate G. For DC48V load, choose R1 = 3.9K, R2=1K
N-channel Power MOSFET e.g. IRF530 can sink 12A DC at up to DC100V max.

PWM Speed Control of a large DC Motor.

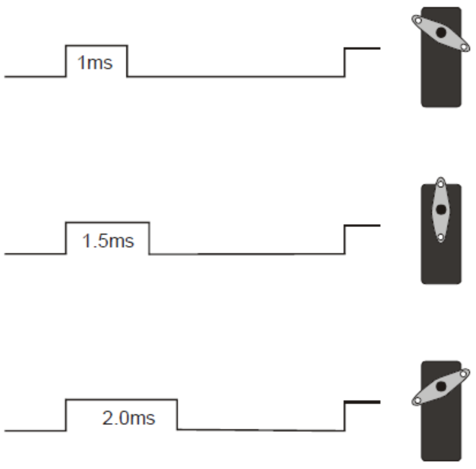
The advantage of using the PWM is that you can easily amplify the drive current to a larger load such as a larger permanent magnet DC motor by using a power transistor or power MOSFET to boost the current switching capability. If the load is of a different voltage and the load current is high, you should use an opto-isolator to isolate the PLC from the load, as shown the above diagram.

Note:

1. The opto-isolator must be able to operate at a frequency matching that of the PWM frequency, otherwise the resulting output waveform will be distorted and effective speed control cannot be attained.
2. The simple PWM speed control scheme described above is the open-loop type and does not regulate the speed with respect to changing load torque. Closed-loop speed control is attainable if a tachometer (either digital or analog) is used which feeds back to the CPU the actual speed. Based on the error between the set point speed and the actual speed, the software can then adjust the PWM duty cycle accordingly to offset speed variation caused by the varying load torque. A PID function may also be invoked to provide sophisticated PID type of speed control.
3. The Fx2424's PWM can be used to control the speed of small motors only (up to the maximum current limit that the PLCs output can safely drive). For larger motors, industrial-grade variable-speed drives should be used instead.

11.3 RC Servo Motor Control

RC Servo is a class of DC servo motor commonly used in remote control (hence the term RC) for positioning a device at a desired location. It is often termed “proportional control” because the position where the motor will turn to is directly proportional to the pulse width of the control signal. When chosen appropriately, RC Servo can provide an extremely inexpensive and versatile solution for positioning a device. For example, for controlling the percentage opening of a HVAC damper, or to rotate the angle of window blinds or to position a solar panel to track the sun light. There are many sizes of RC Servo available in the market, from those that weigh just a few grams to those for controlling an industrial scale unmanned vehicle (e.g. UAV or unmanned submarine). A small, self-contained RC servo typically cost **less than $20** retail price and is incredibly easy to control using the PWM output on the Wx100.



RC Servo typically only have 3 wires: Power “ + ” (typically 4.8 to 6V), “ – “ and a “Control” input.

To position the RC Servo to a position within its range of travel (Some are 0 to 90 degree and there are those that can go from 0 to 180 degree), you send a positive pulse with pulse width between 1.00 ms to 2.00 ms to the “Control” input once every 20ms. The servo will position the actuator to one end when it receives a pulse of 1.00ms and the other end when it receive a pulse of 2.00ms. If you send it a pulse of 1.50ms it will position the actuator to the center.

In PWM term, 1.00ms pulse width every 20ms means a duty cycle of 1.00/20.00 = 5.00%. 2.00ms pulse width every 20ms means a duty cycle of 2.00/20.00 = 10.00% duty cycle. So to control the position of the actuator one only needs to send it a positive PWM signal with frequency = 1/0.02 = 50Hz and duty cycle between 5% and 10%.

The gear and servo feedback mechanism within the RC Servo produces a huge amount of torque relative to its weight for positioning the actuator to the position determined by the pulse width at the “control” input. Yet once it is in the correct position the servo draws only minimum current required to maintain its position. Hence being a closed-loop controller an RC Servo is actually a much more efficient and effective positional control device (for a limited range) than a stepper motor which relies on a constant current in its winding to provide the “holding torque” for positioning. The open loop nature of stepper motor means that it does not know if the device is actually being knocked out of its desired location whereas in the Servo any deviation from its desired location is instantly being corrected by the servo mechanism.

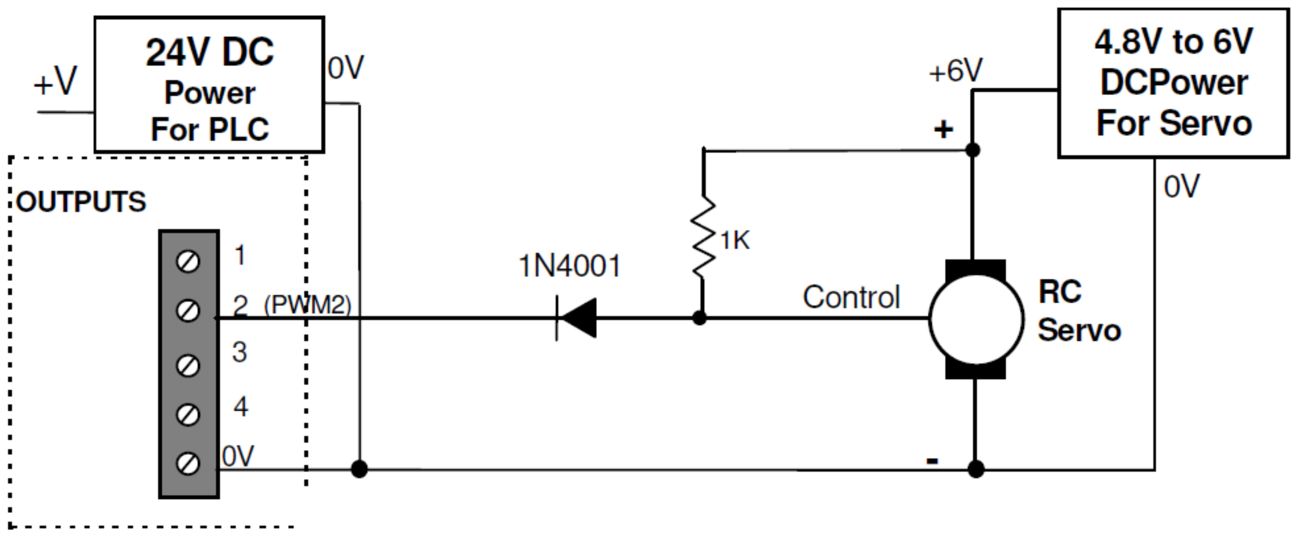
RC Servo Positioning Accuracy

Since the resolution on the Wx100 PLC’s PWM output is precise to 0.01% at 50Hz, this means that you can get a maximum of **500** discrete positions within the travel range of the RC Servo. This should be sufficiently accurate for many applications such as HVAC damper control or control of proportional valves. The actual positioning accuracy and precision, however, will depend on the quality of the RC Servo.

a) PWM to Servo Interface (Non isolated)

As you probably have realized by now that you can use a single PWM output to very easily position the RC Servo to whatever position within its range of travel.

However, since the Wx100 power supply is 12 to 24V (vs 4.8 to 6V on the typical RC Servo) and the PWM output is NPN (current sink) type (on the WxTERM8 board), the signal to the servo is actually inverted if you connect the PLC’s output to the Servo’s “control” input directly.



Non-isolated Interace to RC Servo

As shown in the above figure, we use a 1K ohm resistor to pull up the “Control” input to the RC Servo’s power supply so that when the PLC’s output is OFF, the Control input is +6V and when the PLC output is ON, the “Control” input is pulled to low.

The 1N4001 diode is to prevent the 24V weak pullup signal at the PLC output from entering the servo's "Control" input. As such, when the PLC output is ON, the "Control" input is pulled down to about 1 diode drop (about 0.7V).

In other words, the RC Servo connected above are controlled by the PLC's PWM output but the duty cycle is inverted. I.e. To send a 5% positive PWM control pulse to the Servo, you can run the following statement:

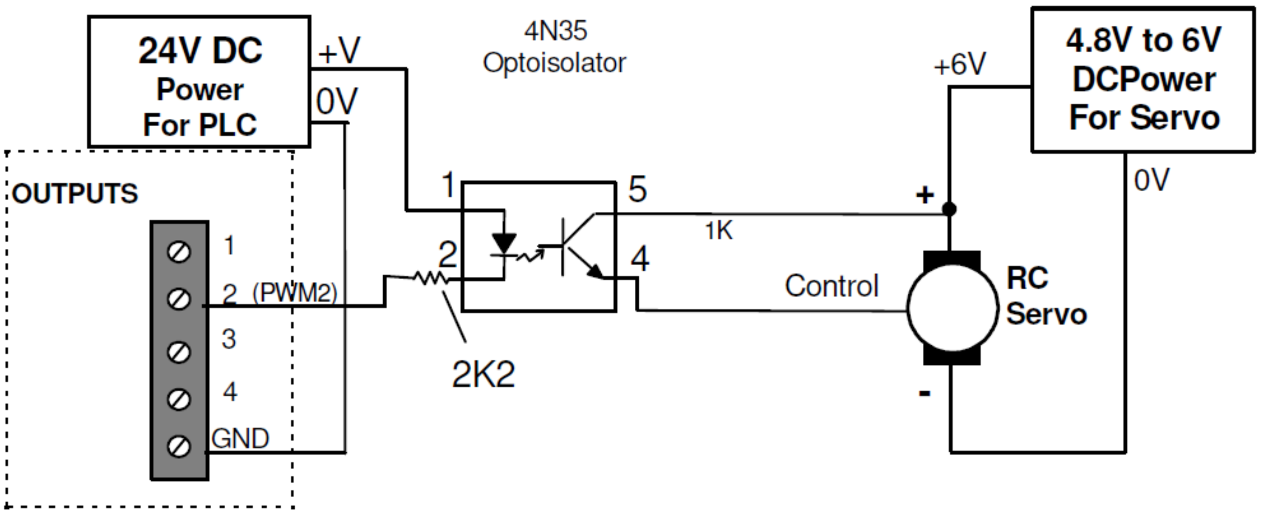
```
SETPWM 2, 9500, 50     ' Set PWM 2 output to 95% duty cycle at 50 Hz
```

Likewise, to send a 10% positive PWM control pulse to the Servo, you will need to run the following statement:

```
SETPWM 2, 9000, 50     ' Set PWM 2 output to 90% duty cycle at 50 Hz
```

b) PWM to Servo Interface (Opto-isolated)

You can also use the PLC's PWM output to drive an optocoupler (such as 4N35) and the output from the optocoupler is use to provide control pulse to the RC Servo, as shown below:



Opto-Isolated Interace to RC Servo

In the opto-isolated interface shown above, the power supply to the PLC and that to the RC Servo are completely isolated (no common ground required). Also since the interface inverts the output signal from the PLC, no software inversion is necessary. i.e When the PLC NPN output is OFF, the “control” input to the RC Servo will be OFF, and when the PLC NPN output is ON, the “control” input to the RC Servo = +6V.

Hence, to send a 5% positive PWM control pulse to the Servo, you can run the following statement:

```
    SETPWM 2, 500, 50     ‘ Set PWM 2 output to 5% duty cycle at 50 Hz
```

Likewise, to send a 10% positive PWM control pulse to the Servo, you can run the following statement:

```
    SETPWM 2, 1000, 50     ‘ Set PWM 2 output to 10% duty cycle at 50 Hz
```

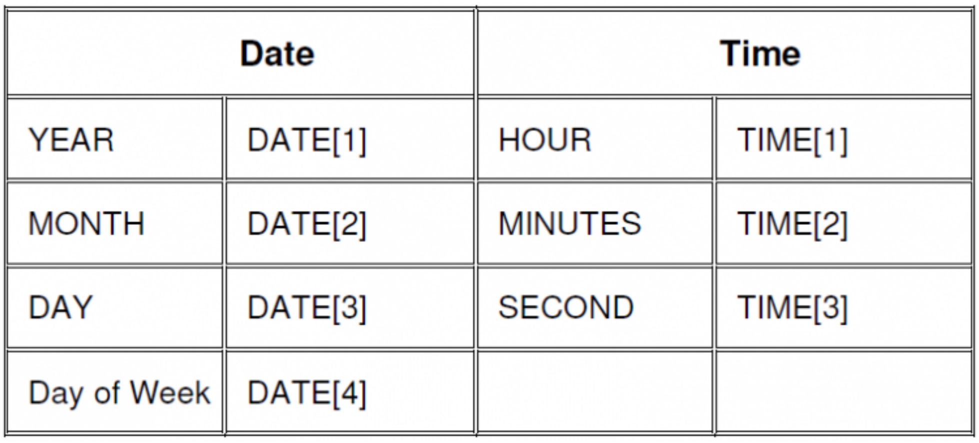
Of course you can replace the duty cycle with a value (e.g. DM[1]) and run statement such as:

```
    SETPWM 2, DM[1], 50   ‘ Set PWM 2 output to DM[1]/100% duty cycle at 50 Hz
```

Chapter 12 - Real Time Clock

A Real Time Clock (RTC) is a device that keeps accurate date and time information down to the second. The RTC in the Wx100 is not battery-backed so the clock settings will be lost if the PLC loses power.

a) TBASIC variables Used for Real Time Clock



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]		

There are 7 registers available in TBASIC that are used to access and configure the date and time. These registers, which are shown above, can be read from and written to just like any other integer variable. The data for these registers are in integer format.

```
DATE[1] : may contain four digits (e.g. 1998, 2003 etc).
DATE[4] : 1 for Monday, 2 for Tuesday, .... 7 for Sunday.
```

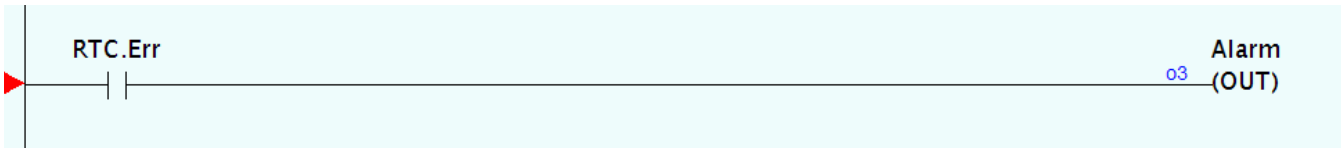
b) Reading RTC Time As A Single BCD Number

The function STATUS(18) can be used to read the RTC time as a single BCD (Binary-Coded-Decimal) number in the following format:

```
A = STATUS(18)    ' A will be returned as HHMMSS
                  ' where HH is from 00 to 23
                  ' MM is from 00 to 59
                  ' SS is from 00 to 59
```

c) RTC Error Status On Ladder Logic

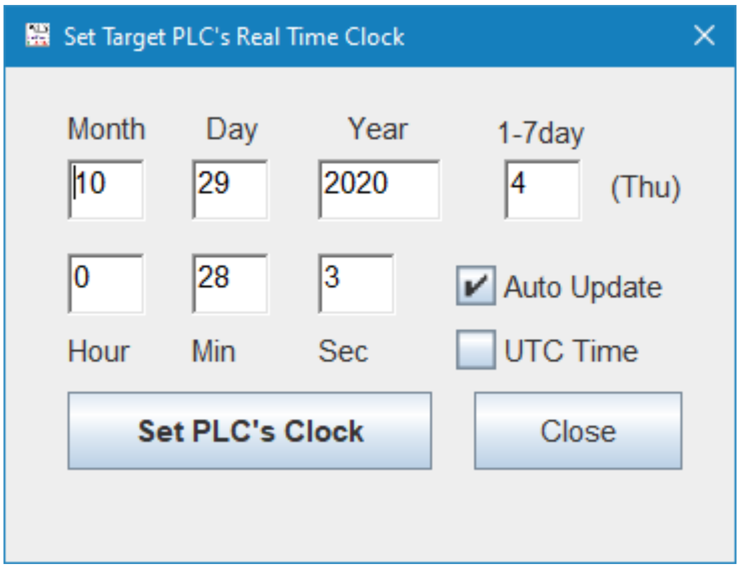
There is a special bit available in TRiLOGI that allows you to notify the PLC program if the RTC Error event occurs and the RTC Error status light is turned on. The RTC Error event occurs if the RTC is corrupted or damaged. The special bit is called RTC.Err and can be obtained from the "Special Bits" I/O Table. The RTC.Err contact can be used to activate an alarm of some kind. The following ladder logic circuit is an example of this using the RTC.Err bit as an input that controls an output called RTC_Alarm:



In the circuit above, RTC.Err is a special bit that cannot be renamed in your program. The output Alarm is a user-defined output that could be named to anything. If the RTC Error event occurs for any reason, the RTC.Err bit would activate the Alarm output.

d) Setting the RTC Using TRiLOGI Software

The RTC date and time can be easily set within TRiLOGI by selecting "Set PLC's Real Time Clock" from the "Controller" menu. A window will pop up with default values entered as shown below. All of these values can be edited and then written to the PLC by clicking on "Set PLC's Clock"



Month 10 Day 29 Year 2020 1-7day 4 (Thu)

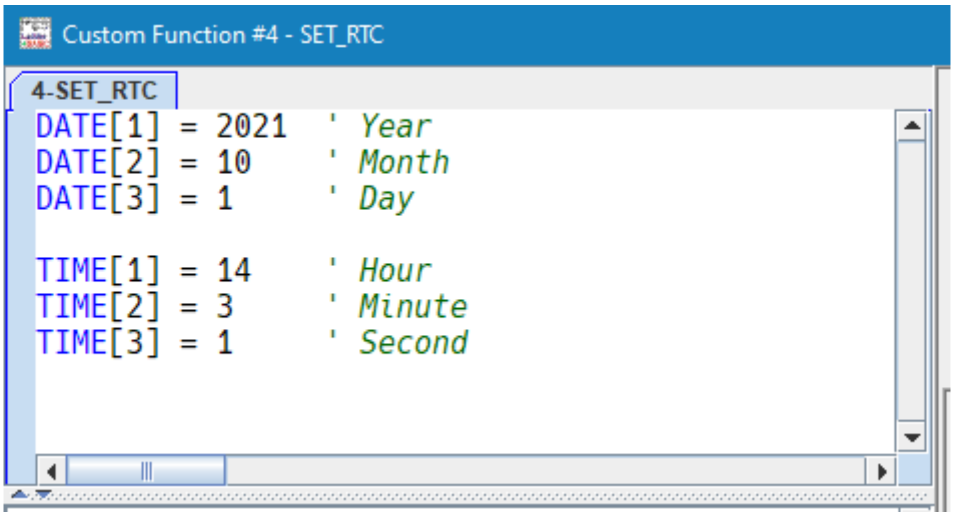
Hour 0 Min 28 Sec 3

[✓] Auto Update

[ ] UTC Time

e) Setting the RTC in TBASIC

The PLCs RTC can be set from TBASIC using the DATE[] and TIME[] registers shown in section 12.2. Figure 12.2: Set Date & Time in TBASIC, shown below, is an example of a custom function where the date is set to October 1st 2021, and the time is 14:30:01 (2:30:01 pm). Note that you do not need to set the DATE[4] ("Day of the week") variable as the RTC will automatically calculate and set the DATE[4] variable and override whatever value you try to set in DATE[4].



```
DATE[1] = 2021    ' Year
DATE[2] = 10      ' Month
DATE[3] = 1       ' Day

TIME[1] = 14      ' Hour
TIME[2] = 3       ' Minute
TIME[3] = 1       ' Second
```

f) Setting the RTC from Internet Time Server

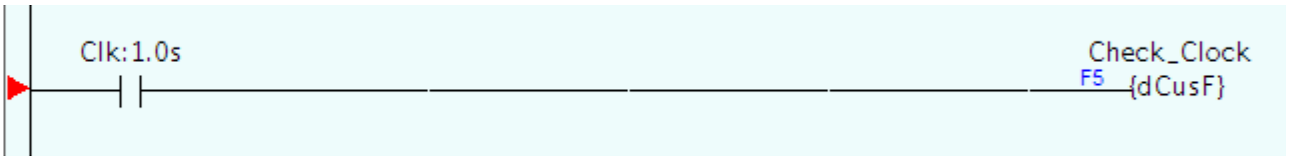
Please refer to Section 2.6 (https://docs.triplc.com/wx100-um/#3022) for more detailed information on how your PLC may be able to automatically set its own real time clock using the timeserver data available on the Internet. A sample program is also included in that section.

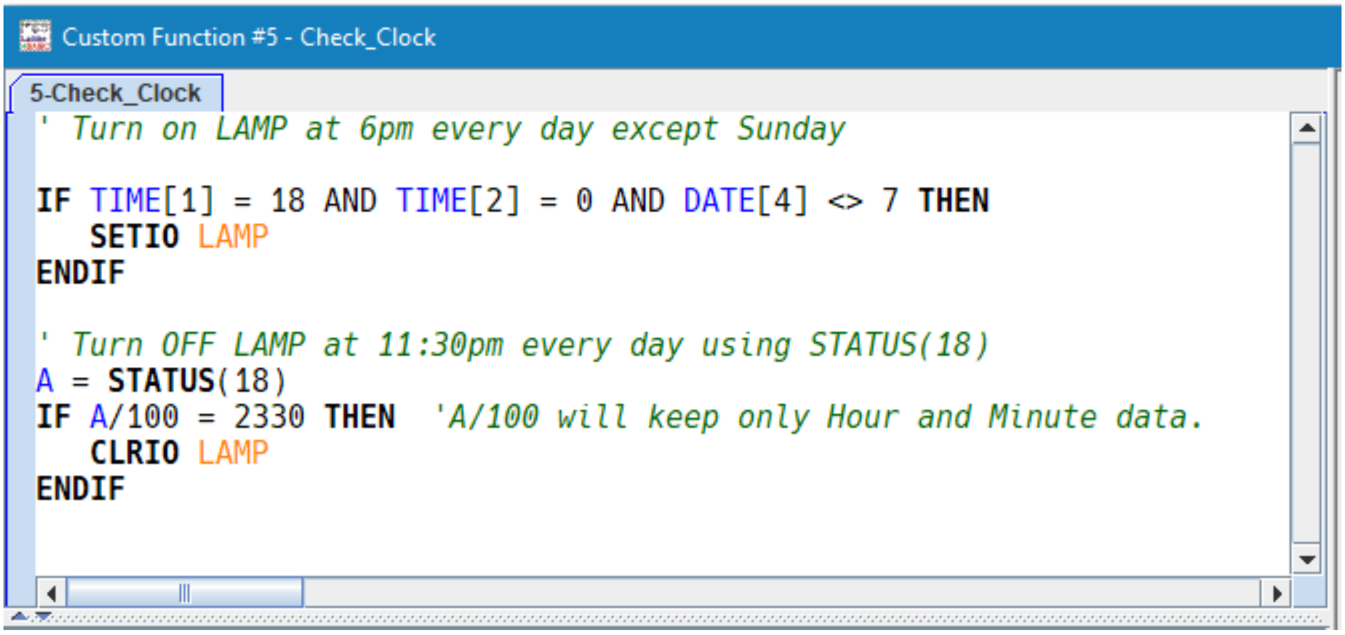
g) Setting up an Alarm Event in TBASIC

To the TBASIC program, the RTC data are simply integer arrays DATE[1] to DATE[4] and TIME[1] to TIME[3], they are fully accessible at any time by your PLC program. Therefore, if you want your program to execute a certain routine on a specific date and or/time, you would need to periodically check these variables against the desired settings and activate the action when the RTC variable(s) reach the set value.

You can also check the time data as a single BCD number using the STATUS(18) and compare against the target value.

Example: Set up a 1 sec clock pulse to monitor the RTC as follows:





```
' Turn on LAMP at 6pm every day except Sunday

IF TIME[1] = 18 AND TIME[2] = 0 AND DATE[4] <> 7 THEN
   SETIO LAMP
ENDIF

' Turn OFF LAMP at 11:30pm every day using STATUS(18)
A = STATUS(18)
IF A/100 = 2330 THEN  'A/100 will keep only Hour and Minute data.
   CLRIO LAMP
ENDIF
```

Chapter 13 - HMI Programming

The single biggest difference of the Wx100 PLC from the other TRi Super PLCs is the inclusion of Human Machine Interface (HMI) which comprises a keypad and a 128 x 64 pixels, graphical OLED display as a standard on every PLC.

We have also updated the i-TRiLOGI software to directly support the keypad and the OLED display and thus can greatly simplify the programming effort for OEM to provide a great user interface for either end-users or the service technicians alike.

13.1 Draw Text with SETLCD

Users of the FMD and Fx PLC may be familiar with the **SETLCD** command that allows the PLC to display text strings on a 4 lines x 20 characters, optional LCD display that are supported by these PLCs.

Although the Wx100 PLC does not have a built-in LCD port to connect to the external LCD, it does support the **SETLCD** command in emulated mode. Wx100 will run the **SETLCD** command and store the result inside an internal “virtual LCD” memory buffer which is visible from the iTRiLOGI software as well as displayed on web browser via the webapp interface. This allows the programmer to use the virtual LCD to store some debugging information even without a real LCD.

However, to support previous and current users of FMD or Fx PLC users so that they can more easily port their original programs to the Wx100 with minimum effort, the Wx100 by default will automatically display the result of the **SETLCD** command on the OLED screen by emulating it as a 4 lines x 21 characters LCD display.

Notes:

1. It is possible to de-link the virtual LCD from the OLED so that executing the **SETLCD** command will only affect the virtual LCD and will not be affect the OLED screen. Run the following command to de-link or link the OLED from the SETLCD command:

```
SETSYSTEM 30, n      ' n = 0: De-link OLED from SETLCD
                     ' n = 1: Link OLED to SETLCD
```

   You may want to de-link the OLED from the **SETLCD** so that you can use the virtual LCD to display info for debugging or monitoring purposes that you do not want to show on the OLED screen for the end users.

2. The content on the OLED display screen is not currently displayable on a webApp. So if you want to display information that is viewable from a web browser you need to display the info using the **SETLCD** command

Wx100 simultaneously supports both the **SETLCD** command as well as newly added TBASIC commands such as **DRAW_TEXT, DRAW_TEXTPOS, DRAW_RECT, FILL_CIRCLE** etc. We will discuss these new graphical commands in the next section.

a) SETLCD *y, x, string*

TBASIC command allows you to easily display any string of up to 20 characters on the *y*th line starting from the *x*th column. E.g., to display the message "Wx100" on the 3rd line starting from the 5th character position from the left end of the screen, you use the command:

```
SETLCD 3, 5, "Wx100 Super PLC"
```

Normally, y = 1,2,3, 4; x = 1, 2, ... 20. Integers must be converted to strings using the STR$() or HEX$() function before they can be displayed using SETLCD. You can use the concatenation operator "+" to combine a few components together in the command. E.g.

```
SETLCD 1,1,"Rm Temp = "+ STR$(ADC(1)/100,3)+" deg C"
```

The function STR$(ADC(1)/100,3) reads the content of ADC channel #1, divides it by 100 and converts the result into a 3-digit string. The final string being displayed will be :

```
Rm Temp = 012 deg C.
```

b) Special Commands For LCD Display

If you use the SETLCD command with line #0, then the strings will be treated as special "instructions" to be sent to the LCD module to program it for various modes of operation. For real external LCD this includes: blinking cursor, underline cursor or no cursor, as well as display shift mode. You have to refer to the LCD manufacturer's data sheet for the detailed commands.

Since Wx100 does not support the actual LCD, only one special SETLCD command is supported as shown below:

```
SETLCD 0, 1, CHR$(1)   ' clear LCD screen. Will also clear OLED screen if linked.
```

c) Displaying Numeric Variable With Multiple Digits

The "SETLCD *y, x, string"* command only overwrites the exact number of characters in the *string* parameter to the LCD display, and thereafter the cursor is placed back to the location specified by the *x* and *y* parameters. Thus if there exists some old characters right after the last character it can cause confusion, especially if you are displaying a number. E.g. If you first display the following string at row 1 and column 1:

```
Pressure = 12345
```

And if you subsequently display the string "Pressure = 983" at the same location without first clearing the line, then you will see the following string being displayed:

```
Pressure = 98345
```

What happens is that the string “Pressure = 983” is correctly displayed but the two old characters “45” left over from previous display would appear to be part of the new data. This can cause confusion.

There are several ways you can eliminate such a display problem:

1. Clear the line first before overwriting with a new string. You can create a custom function just to clear a particular line. E.g. if you pass the parameter DM[100] to the custom function and inside the custom function you do the following:

   ```
   SETLCD DM[100],1, "                    "
   ```

   Hence calling this custom function with DM[100] = 1,2,3, or 4 would clear the corresponding line.

2. A “ quick and lazy” way to do it is to add a few more characters to the back of the string to be displayed which will wipe out old characters that could be present adjacent to the new string.

   ```
   SETLCD 1,1, "Money = $"+ STR$(D) + "       "
   ```

3. If there is data to the right of the currently displayed string that you cannot overwrite with spaces, then you can restrict the number of digits that a numeric variable may be converted to using the two parameters from the STR$ or HEX$ command.

   ```
   SETLCD 1,1, “Temp = ”+ STR$(T,4)
   ```

   The STR$(T,4) function will always return 4 digits of the data stored in T. If T is less than 4 digits, then one or more preceding “0”’s will be added. E.g. if T = 12 then this function will return the string “0012. Note that for negative numbers the negative sign is counted as part of the digit count so you need to provide enough spaces to take care of the sign if you need to handle both positive and negative numbers.

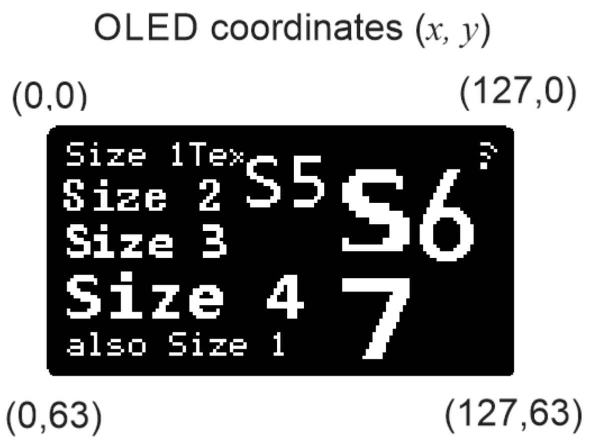
d) Displaying Floating Points

The STR$(x#) and STR$(x#,n) can convert a floating point number x# into a text string which can therefore be displayed immediately using the SETLCD command. Please refer to the help file for the STR$() (https://triplc.com/TRiLOGI/Help/tbasic/str$.htm) command.

If you wish to control the number of digit being displayed then you could use the MID$ command to extract the number of significant digits and the exponents from the string returned by STR$(x#, n) function.

13.2 DRAW_TEXT & DRAW_TEXTPOS

Wx100 has a 128 x 64 pixel graphical OLED display and hence is capable of displaying text at any coordinates (x, y) as shown in the following diagram:



Wx100 supports up to 7 text sizes as shown in the following table:

Text Size	Width x Height (pixels)
1	6 x 8
2	8 x 12
3	8 x 15
4	12 x 21
5	13 x 24
6	18 x 32
7	26 x 32

Commands for Drawing Text On The OLED

Command	Documentation
DRAW_TEXTPOS	https://triplc.com/TRiLOGI/Help/tbasic/draw_textpos.htm (https://triplc.com/TRiLOGI/Help/tbasic/draw_textpos.htm)
DRAW_TEXTSIZE	https://triplc.com/TRiLOGI/Help/tbasic/draw_textsize.htm (https://triplc.com/TRiLOGI/Help/tbasic/draw_textsize.htm)
DRAW_TEXT	https://triplc.com/TRiLOGI/Help/tbasic/draw_text.htm (https://triplc.com/TRiLOGI/Help/tbasic/draw_text.htm)

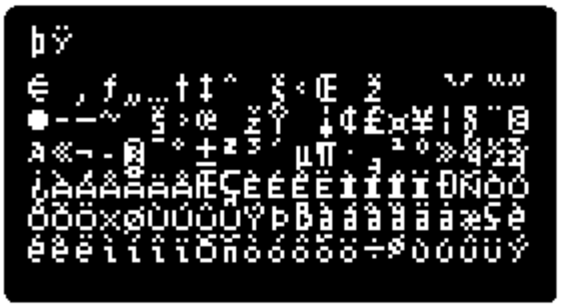
13.3 Display Character Sets

Please download the following example program to help demonstrates what will be discuss below

https://triplc.com/TRiLOGI/Examples/Wx_Specifics/DisplaytExtendedAscii_And_Custom_Bitmap.zip (https://triplc.com/TRiLOGI/Examples/Wx_Specifics/DisplaytExtendedAscii_And_Custom_Bitmap.zip)

a) Default Character Set

Using TBASIC, Wx100 can display all basic ASCII characters from ASCII code 32 to 127 (decimal) . In addition, it can display extended ASCII based on Windows-1252 (https://www.ascii-code.com/) (code page 1252), aka ISO-8859-1 character set, which includes many European characters as shown below:

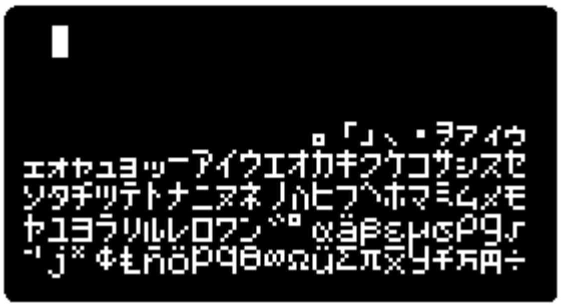

Extended ASCII – CP1252

The default character set are supported for all text sizes from 1 to 7.

b) Hitachi HD44780 LCD character set.

TBASIC also supports the Hitachi HD44780A00 character set (https://triplc.com/documents /LCDasiiTable.pdf) which is very common among character LCD displays that are based on the Hitachi HD44780 controller chip or one of its licensed derivatives. (Note: this chip is used in the LCD216 and LCD420 that are used with our other lines of PLC such as the FMD88-10 and Fx-2424 PLCs).

The Hitachi HD44780 includes a full set of Japanese Kana characters. If you need to display these characters then you need to specify the **text size = 11** before calling the DRAW_TEXT , DRAW_TEXTPOS or SETLCD (https://docs.triplc.com/#3646) commands. Only a single size (6 x 8 pixels) is supported.

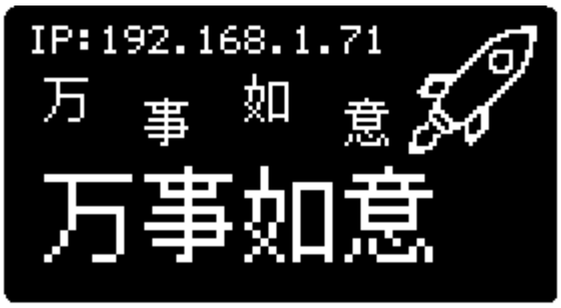


Extended ASCII – HD44780A00

c) Customized Bitmap Graphics

It is possible to display characters that are unsupported by the Windows-1252 or Hitachi HD44780 character by creating them as custom bitmap graphic (https://docs.triplc.com/#6140).

For example, you can display Chinese Characters or special symbol created as custom bitmap graphic to display the following:



For more details, please refer to Section 13.7 – Draw Custom Bitmap (https://docs.triplc.com /#6140)

13.4 Reading Key Pad Entry

a) Reading last Key Pressed

When a user pressed a key on the Wx100 built-in keypad, the event is captured and the key code of the last key pressed can be retrieved by the following TBASIC function:

```
A = KEYPRESSED(1)
```

The following are the key codes returned by KEYPRESSED(1) for each key:

Keys	Key codes
0 to 9	0 to 9
10 .	10
11 ↵	11
12 ≡	12
13 ▲	13
14 ▼	14
15 ⌫	15

Notes:

1. When no key is pressed, KEYPRESSED(1) function returns a -1.

2. KEYPRESSED(1) will return the key code of the last key pressed even if the key has already been

released by the time the program runs the KEYPRESSED(1) function.

3. To simplify programming for number entry purpose, the KEYPRESSED(1) function will return the keycode **ONLY ONCE** when it is first called. i.e. Even if you hold down the same key continuously, subsequent call of the KEYPRESSED(1) function will always return a -1. If you release the key and then press it again then the KEYPRESSED(1) function will again return the actual keycode only once when it is called.
4. This is to prevent a custom function from repeatedly executing a program sequence even if you just meant for it to execute once. For example if you press the '1' key when entering a number, you do not want the custom function to record it as a continuous streams of 1 being entered. The PLC program runs much faster than human can release the key in time so having KEYPRESSED(1) returns the keycode only just once each time the key is pressed will ensure that only a single digit is entered.

b) Reading Continuous Key Pressed

If you need to read the same key code as long as the key is being pressed and held down, then you can use the following function instead:

```
A = KEYPRESSED(0)
```

You may need to use this capability for example if you want to be able to press on a key and have a number scroll up or down rapidly and stop when the key is released.

c) Using Keypad In Ladder Logic

You can easily translate the actual key pressed into open or close contacts of some internal relay bit so that the keypad can be used in place of physical push buttons normally wired to the inputs of a PLC. The following function call automatically transfer the keycode 0 to 15 into the corresponding bit position in the 16-bit variable RELAY[2] :

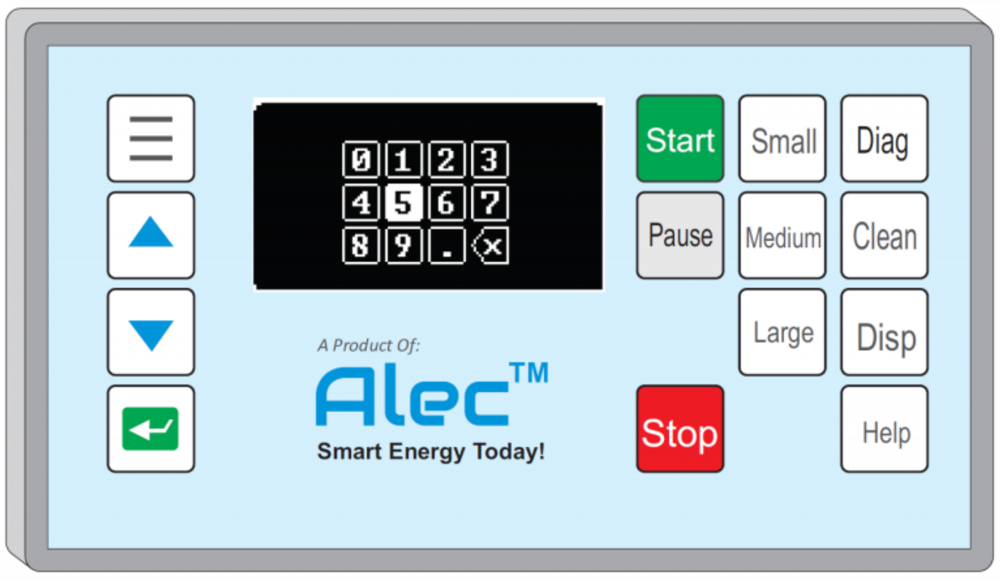
```
  RELAY[2] = KEYPRESSED(2)  ' transfer the 16 keys into internal relay 17 to 32
```

The above statement essentially turn the KEYPAD into 16 digital inputs and their logic '0' and '1' are transferred to internal relay #17 to #32. The resulting relay bit can then be readily used by the ladder logic part of the program.

One effective way to use this capability is to use a clock pulse to drive a differentiated custom function that periodically runs the above single line statement so that the internal relay #17 to #32 is regularly updated which can then be used to trigger different parts of the ladder logic.

d) Customized Keypad Consideration

You probably already figure it by now – that although the built-in keypad appears to be arranged and labeled conveniently for entering numbers into the PLC, the key legends on the keypad are not hard-coded for a specific purpose only. Since the key pressed actions are returned as numeric codes. the PLC program can easily use any key for any purpose. Hence if you do not need to enter numbers into your equipment then the key pads can be re-purposed as push button inputs to trigger different parts of the program. This can have tremendous cost savings for OEMs as physical push buttons are quite expensive and also result in high labour cost to wire them. For mid to large OEM customers it is possible to order the PLC with customized legends so that these keys can be used for other purposes than as numeric keypad entry. For example, it is totally conceivable that you could custom-order a Wx100 with the following keypad legends:



13.5 Menu Programming

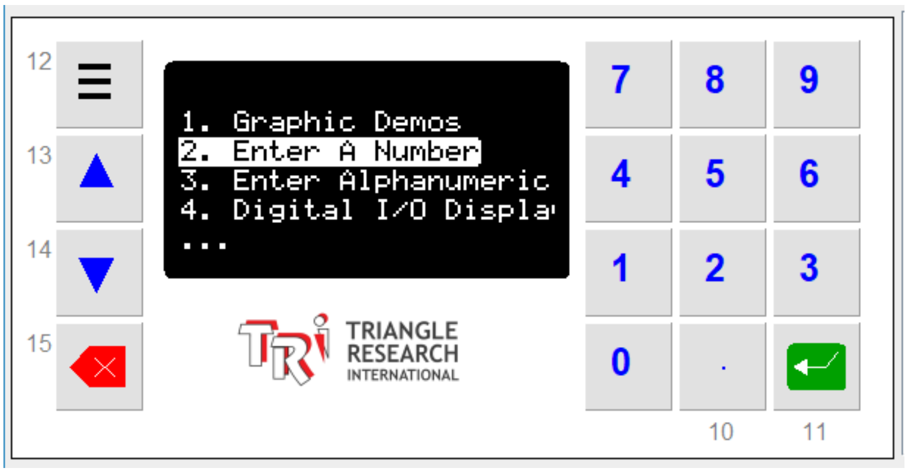
TBASIC makes it extremely simple for the programmer to create a menu system where the user can select a function that he/she wants the PLC to run by picking it from a menu. You start first by defining the menu structure using the “MENU_DEFINE” command. Then the program simply runs the MENU_SHOW command and the Wx100 firmware will automatically highlights the selected item.

Commands for Displaying MENU On The OLED

Command	Documentation
MENU_DEFINE	https://triplc.com/TRiLOGI/Help/tbasic/menu_define.htm (https://triplc.com/TRiLOGI/Help/tbasic/menu_define.htm)
MENU_SHOW	https://triplc.com/TRiLOGI/Help/tbasic/menu_show.htm (https://triplc.com/TRiLOGI/Help/tbasic/menu_show.htm)

Example: The best way to quickly experiment with the Wx100 menu system is to download the WxDEMO.PC7 (https://www.tri-plc.com/download/WxDemo.zip) program and then open it using the iTRiLOGI 7.4 software. It You can test the menu system using the iTRiLOGI simulator to simulate the function and click on the up down arrow keys or click the 1 to 8 button on the virtual HMI (see figure below) to jump to the desired menu item and then click on the green "Enter" key to select the menu. There are a great numbers of demos that you can experiment with that showcase the capability of the HMI.

Feel free to modify the program and immediately simulate it to see the result of the changes you have made. When you are satisfied with your simulated program, transfer it into Wx100 PLC and test using the real push button and display on the OLED screen.


```
1. Graphic Demos
2. Enter A Number
3. Enter Alphanumeric
4. Digital I/O Displa
...
```

13.6 Draw Vector Graphics

The following graphical drawing commands are supported:

Command	Documentation

DRAW_COLOR	https://triplc.com/TRiLOGI/Help/tbasic/draw_color.htm (https://triplc.com/TRiLOGI/Help/tbasic/draw_color.htm)
DRAW_CIRCLE	https://triplc.com/TRiLOGI/Help/tbasic/draw_circle.htm (https://triplc.com/TRiLOGI/Help/tbasic/draw_circle.htm)
DRAW_LINE	https://triplc.com/TRiLOGI/Help/tbasic/draw_line.htm (https://triplc.com/TRiLOGI/Help/tbasic/draw_line.htm)
DRAW_RECT	https://triplc.com/TRiLOGI/Help/tbasic/draw_rect.htm (https://triplc.com/TRiLOGI/Help/tbasic/draw_rect.htm)
DRAW_ROUNDRECT	https://triplc.com/TRiLOGI/Help/tbasic/draw_roundrect.htm (https://triplc.com/TRiLOGI/Help/tbasic/draw_roundrect.htm)
DRAW_TRIANGLE	https://triplc.com/TRiLOGI/Help/tbasic/draw_triangle.htm (https://triplc.com/TRiLOGI/Help/tbasic/draw_triangle.htm)
FILL_CIRCLE	https://triplc.com/TRiLOGI/Help/tbasic/fill_circle.htm (https://triplc.com/TRiLOGI/Help/tbasic/fill_circle.htm)
FILL_RECT	https://triplc.com/TRiLOGI/Help/tbasic/fill_rect.htm (https://triplc.com/TRiLOGI/Help/tbasic/fill_rect.htm)
FILL_ROUNDRECT	https://triplc.com/TRiLOGI/Help/tbasic/fill_roundrect.htm (https://triplc.com/TRiLOGI/Help/tbasic/fill_roundrect.htm)
FILL_TRIANGLE	https://triplc.com/TRiLOGI/Help/tbasic/fill_triangle.htm (https://triplc.com/TRiLOGI/Help/tbasic/fill_triangle.htm)

13.7 Draw Custom Bitmap

Besides displaying ASCII text and vector graphics via DRAW_XXX command, TBASIC is also able to display custom-designed bit map graphic block of up to 32 x 32 pixels. The procedure begins by designing the graphic using a special graphic editor software, then export the generated bit pattern as data points to be assigned to the PLC’s DM[n] variables. Finally, run the new TBASIC

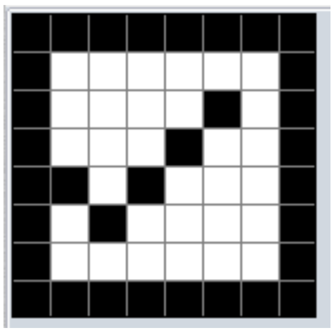
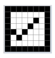
DRAW_BITMAP command to display the bitmap graphics on the OLED display. We will explain the procedure below.

```
  **DRAW_BITMAP** x, y, dmIndex, magnification
 - x and y are the coordinates to display the bitmap character
 - dmIndex is the starting index of the DM that stores the bitmap. It can start anywhere
 from
      1 to (4000 - number of bytes needed to represent the bitmap).
 - magnification factor.  E.g. if = 2 means that the character is displayed at double th
 e size.
       So a 6 x 8 character will be displayed as 12 x 16.
```

Using the DM is convenient as it allows you to load the characters either from EEPROM or from a data file or even from an online source and then deploy it as and when needed.

Now what do you store in the DM? If the bitmap is less than 8 pixel tall, then a single byte is used to represent a column of dots where a ‘1’ means that the dot is ON and ‘0 means that the dot is OFF

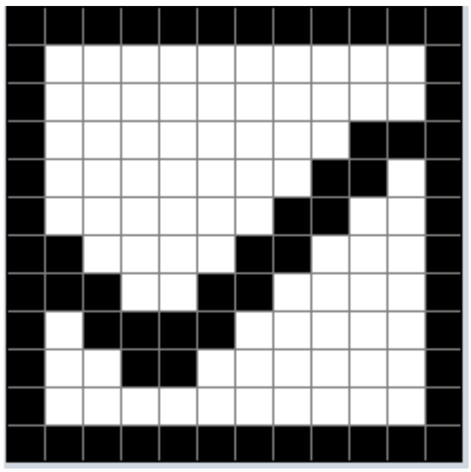
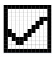
For example, the following graphics can be represented as an array of bytes (hex): &HFF, &H91, &HA1, &H91, &H89, &H85, &H81, &HFF

 (https://docs.triplc.com/wp-content/uploads/2021/03/ejqc27pc.bmp) 

(https://docs.triplc.com/wp-content/uploads/2021/03/ejqc27pc.bmp)

If the bitmap is larger than 8 but less than 16 pixel tall, then 2 bytes are used to represent each column. A 32 pixel tall bitmap requires 4 bytes to represent the column and as many 4-byte words as is needed to represent the number of columns.

E.g. A 12 x 12 pixel bitmaps as following :

 (https://docs.triplc.com/wp-content/uploads/2021/03/ivybyw21.bmp)  (https://docs.triplc.com/wp-content/uploads/2021/03/ivybyw21.bmp)

can be represented by &HFF, &H0F, &HC1, &H08, &H81, &H09, &H01, &H0B, &H01, &H0B, &H81, &H09, &HC1, &H08, &H61, &H08, &H31, &H08, &H19, &H08, &H09, &H08, &HFF, &H0F

Note: it is written in Little Endian format. So the leftmost column can be thought of as &H0FFF but is represented by the first two byte: &HFF, &H0F.

Likewise, second column should be &H08C1 but is represented by the 3rd and 4th byte: &HC1, &H08
.. and so on.

It will be a real pain to try to do this by hand but fortunately there is a free program call MicroElektronika GLCD Font creator which you can download from:

https://www.mikroe.com/glcd-font-creator (https://www.mikroe.com/glcd-font-creator)

MikroElektronika GLCD Font Creator
VERSION 1.2

REGISTERED TO: Free version

CREDITS: Ahmed Lazreg

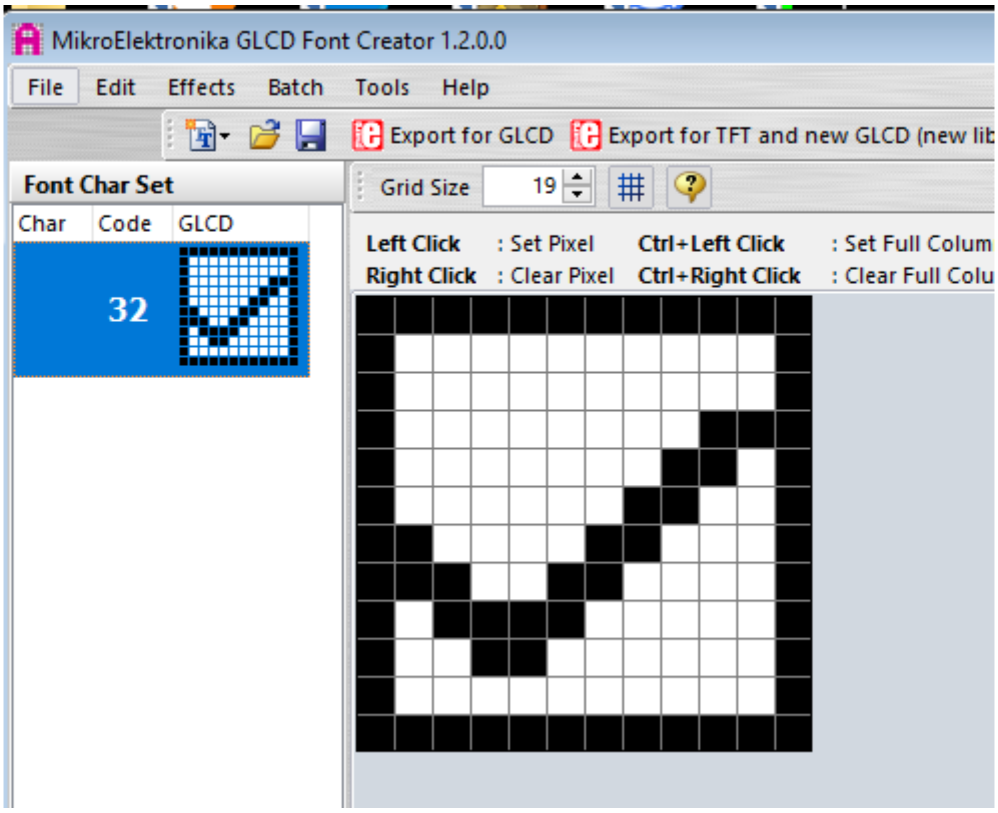
WEBSITE: http://www.mikroe.com
EMAIL: office@mikroe.com
FORUM: http://www.mikroe.com/forum

© 2002 - 2011 MikroElektronika

(https://docs.triplc.com/wp-content/uploads/2021/03/s1yczqm3.bmp)

The program is meant for you to create bitmpa font for ASCII characters or Unicode characters. You can import some system font for simple graphics (e.g. WingDing) or create from scratch.

When creating from scratch you define the number of width and height for your bitmap and assign it a code number (you can use the 32 to 32 which is actually a space character but won't matter. Then just use left click to draw pixels and right click to erase pixels inside the bitmap.

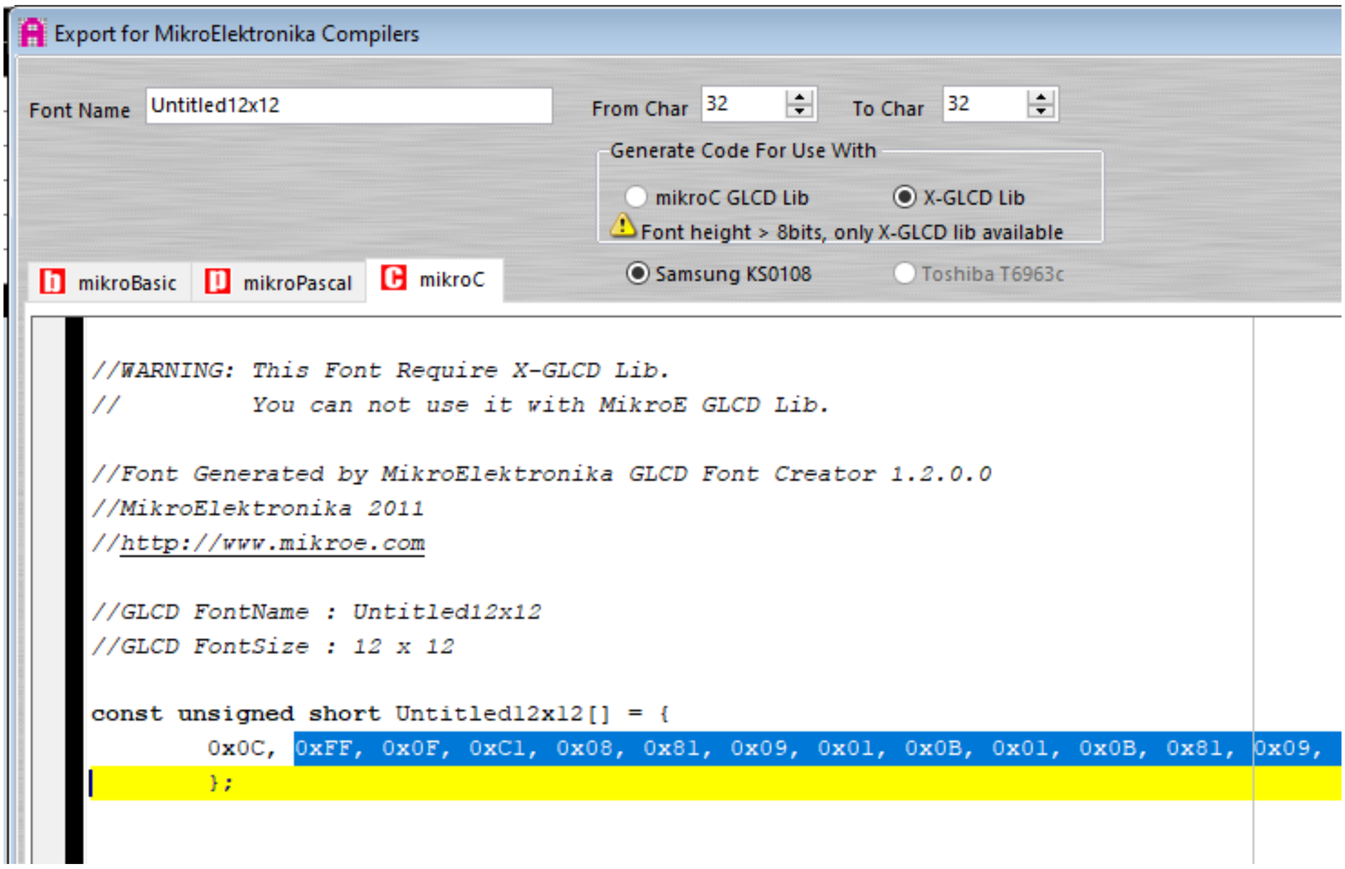

MikroElektronika GLCD Font Creator 1.2.0.0
File
Edit
Effects
Batch
Tools
Help
Export for GLCD
Export for TFT and new GLCD (new lib
Font Char Set
Char
Code
GLCD
32
Grid Size
19
Left Click : Set Pixel
Right Click : Clear Pixel
Ctrl+Left Click : Set Full Colum
Ctrl+Right Click : Clear Full Colu

(https://docs.triplc.com/wp-content/uploads/2021/03/cqnq5xgk.bmp)

Once you are done with creating the bitmap, click on the icon on the left (the one next to 32) to SAVE the character

After you have saved the character, generate the bitmap data by clicking on "Export for GLCD" to generate the Basic, Pascal or C codes for the character bitmap.

If you click on the "mikroC" tab you see the following array. **Discard the very first byte (0x0C as shown below)** as we don't use it.



Font Name: Untitled12x12
From Char: 32
To Char: 32
Generate Code For Use With: ( ) mikroC GLCD Lib [✓ X-GLCD Lib]
Font height > 8bits, only X-GLCD lib available
[✓ Samsung KS0108] ( ) Toshiba T6963c

```
//WARNING: This Font Require X-GLCD Lib.
//         You can not use it with MikroE GLCD Lib.

//Font Generated by MikroElektronika GLCD Font Creator 1.2.0.0
//MikroElektronika 2011
//http://www.mikroe.com

//GLCD FontName : Untitled12x12
//GLCD FontSize : 12 x 12

const unsigned short Untitled12x12[] = {
        0x0C, 0xFF, 0x0F, 0xC1, 0x08, 0x81, 0x09, 0x01, 0x0B, 0x01, 0x0B, 0x81, 0x09,
        };
```

(https://docs.triplc.com/wp-content/uploads/2021/03/zo3ab9kk.bmp)

```
const unsigned short Untitled12x12[] = {
0x0C, 0xFF, 0x0F, 0xC1, 0x08, 0x81, 0x09, 0x01, 0x0B, 0x01, 0x0B, 0x81,
0x09, 0xC1, 0x08, 0x61, 0x08, 0x31, 0x08, 0x19, 0x08, 0x09, 0x08, 0xFF, 0x0F
// Code for char
};
```

The data after the first byte will be what we want to fill in the DM array starting from DM[dmIndex + 4] onwards until the last byte in the array is reached.

You need to fill in DM[dmIndex] to DM[dmIndex+3] the following:

```
DM[dmIndex] = 0   ' reserved, always 0
DM[dmIndex+1] =   bitmap width in number of pixels
DM[dmIndex+2] = bitmap Height in number of pixels
DM[dmIndex + 3] = ' reserved, always 0
```

For example, in the 12 x 12 bitmaps example above, if we starts with dmIndex = 1

```
    DM[1] = 0
    DM[2] = 12
    DM[3] = 12
    DM[4] = 0
    DM[5] = &HFF
    DM[6] = &H0F
    DM[7[ = &HC1
    DM[8] = &H08
    ....
    DM[27] = &HFF
    DM[28] = &H0F
```

Now if you run

```
 DRAW_BITMAP 0, 30, 1, 1
```

it will draw the bitmap checkbox graphic at coordinate x = 0, y = 30 with x1 magnification.

Note: Please download the following example program that shows the creation and display of custom bitmap graphics:

https://triplc.com/TRiLOGI/Examples/Wx_Specifics/DisplaytExtendedAscii_And_Custom_Bitmap.zip (https://triplc.com/TRiLOGI/Examples/Wx_Specifics /DisplaytExtendedAscii_And_Custom_Bitmap.zip)

Chapter 14 - RS485 Serial Port

There is a single, two-wire RS485 serial ports on the Wx100 that have full Modbus ASCII/RTU and Host Link Protocol drivers. The RS485 port can also be programmed to accept or send ASCII or binary data using the TBASIC built-in commands such as **INPUT$**(n), **INCOMM**(n), **PRINT** #n, **OUTCOMM** n, d.

The half-duplex RS485 port is meant for serial bus type networking or for connecting to optional peripherals such as a serial LCD message-display unit (e.g. MDS100-BW), external analog modules (e.g. I-7018), touch panel large screen HMI, or for inter-communication between PLCs. Up to 255 RS232/RS485 peripheral devices may be linked together in an RS485 network.

Please refer to Chapter 1.3 (https://docs.triplc.com/wx100-um/#2440) for help on physical wiring of the RS485 port

14.1 SETBAUD Command

The Wx100 RS485 is highly configurable. It can be set to a wide range of baud rates. You can also program it to communicate in either 7 or 8 data bits, 1 or 2 stop bits, odd, even or no parity. The baud rate and communication formats of the serial ports are set by the following command:

```
SETBAUD  ch, baud_no
```

ch represents the COMM port number (1, 2, or 3 only). The baud_no parameter takes a value from 0 – 255 (&H0 to &HFF), which allows for additional configuration of the communication format. The upper 4 bits of baud_no specify the communication format (number of data bits, number of stop bits, and parity) and the lower 4 bits represent the baud rate. Hence the baud_no for 8 data bit, 1 stop bit, and no parity is the same as the old models, providing compatibility across the PLC families.

Once the new baud rate has been set, it will not be changed until execution of another SETBAUD statement. The baud rate is not affected by a software RESET but the settings will be lost when the power is turned OFF. The available baud rates and their corresponding baud rate numbers for COMM1, 2, and 3 are shown below:

Format	*baud_no*
8, 1, n	0000 xxxx
8, 1, e	0100 xxxx
8, 1, o	0110 xxxx
7, 1, n	1000 xxxx
7, 1, e	1100 xxxx
7, 1, o	1110 xxxx

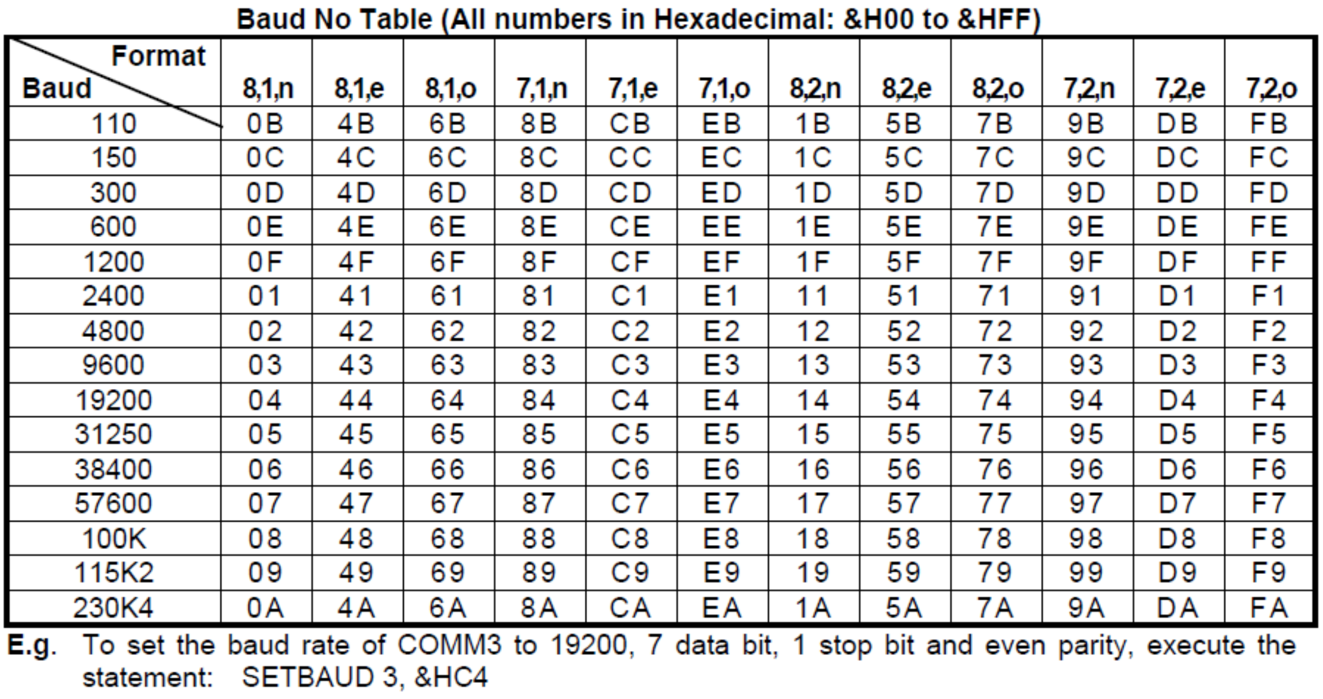
Format	*baud_no*
8, 2, n	0001 xxxx
8, 2, e	0101 xxxx
8, 2, o	0111 xxxx
7, 2, n	1001 xxxx
7, 2, e	1101 xxxx
7, 2, o	1111 xxxx

Where xxxx represents the baud rate of the comm port, as follow:

x x x x	0000	0001	0010	0011	0100	0101	0110	0111
Baud Rate	2400	2400	4800	9600	19200	31250	38400	57600

x x x x	1000	1001	1010	1011	1100	1101	1110	1111
Baud Rate	100K	115.2K	230.4K	110	150	300	600	1200

A table of all the available baud rates and COMM formats is shown in the following page. The communication format written as “7,2,e”, which means 7 data bits, 2 stop bits and even parity. Likewise, “8,1,n” means 8 data bits, 1 stop bit and no parity. You can use the table to select the baud number for a certain baud rate and COMM format.


Baud No Table (All numbers in Hexadecimal: &H00 to &HFF)

Format Baud	8,1,n	8,1,e	8,1,o	7,1,n	7,1,e	7,1,o	8,2,n	8,2,e	8,2,o	7,2,n	7,2,e	7,2,o
110	0B	4B	6B	8B	CB	EB	1B	5B	7B	9B	DB	FB
150	0C	4C	6C	8C	CC	EC	1C	5C	7C	9C	DC	FC
300	0D	4D	6D	8D	CD	ED	1D	5D	7D	9D	DD	FD
600	0E	4E	6E	8E	CE	EE	1E	5E	7E	9E	DE	FE
1200	0F	4F	6F	8F	CF	EF	1F	5F	7F	9F	DF	FF
2400	01	41	61	81	C1	E1	11	51	71	91	D1	F1
4800	02	42	62	82	C2	E2	12	52	72	92	D2	F2
9600	03	43	63	83	C3	E3	13	53	73	93	D3	F3
19200	04	44	64	84	C4	E4	14	54	74	94	D4	F4
31250	05	45	65	85	C5	E5	15	55	75	95	D5	F5
38400	06	46	66	86	C6	E6	16	56	76	96	D6	F6
57600	07	47	67	87	C7	E7	17	57	77	97	D7	F7
100K	08	48	68	88	C8	E8	18	58	78	98	D8	F8
115K2	09	49	69	89	C9	E9	19	59	79	99	D9	F9
230K4	0A	4A	6A	8A	CA	EA	1A	5A	7A	9A	DA	FA

E.g. To set the baud rate of COMM3 to 19200, 7 data bit, 1 stop bit and even parity, execute the statement: SETBAUD 3, &HC4

Important:

Please note that if your program changes the baud rate or communication format to something that is different from that set in the iTRiLOGI serial port setting, then i-TRiLOGI will no longer be able to communicate with the PLC via this COMM port. You will have to either configure the iTRiLOGI's serial port setting using its "Serial Communication Setup" routine to match the PLC, or you can cycle the power to the PLC to reset the COMM port to the default format (38,400, 8,n,1).

If you had used "1st.Scan" contact to activate the SETBAUD command than you will need to cycle the power to the PLC with DIP switch #4 set to ON to halt the execution of the SETBAUD command. When the PLC is reset this way, its COMM1 port will power up at the default format of 38,400, 8,n,1 only so you will need to configure iTRiLOGI's serial port also to 38,400bps, 8, 1, n to communicate with it.

14.2 Multiple Comm Protocols

The Wx100 has been designed to understand and speak many different types of communication protocols, some of which are extremely widely used de facto industry standards, as follows:

a) NATIVE HOST LINK COMMAND
b) MODBUS ASCII (Trademark of Modbus.org)
c) MODBUS RTU* (Trademark of Modbus.org)
d) OMRON C20H protocols. (Trademark of Omron Corp of Japan)

The command and response formats of the “NATIVE” protocols are described in detail in Chapter 15 and 16. The other protocols and their address mapping to the Wx100 are described in Section 14.5. By default Wx100 works in “automatic protocol” mode. There is no DIPswitch to set and no special configuration software to run to configure the port for a specific communication protocol. The following describes how the automatic protocol recognition scheme works:

1. When the PLC is powered ON, its RS485 serial port is set to the “AUTO” mode, which means that it is open-minded and listen to all serial data coming through the COMM ports. The CPU tries to determine if the serial data conforms to a certain protocol and if so, the COMM mode is determined automatically.

2. Once the protocol is recognized, the CPU sets that COMM port to a specific COMM mode, which enables it to process and respond only to commands that conform to that protocol. Error detection data such as the “FCS”, “LRC” or CRC are computed accordingly which method is used to verify the integrity of the received commands. If errors are detected in the command, the CPU responds in accordance with the action specified in the respective protocols.

3. When the COMM port enters a specific COMM mode, it will regard commands of other protocol as errors and will not accept them. Hence, for example, if COMM #1 has received a valid MODBUS RTU command (which puts it in an “RTU” mode), it will no longer respond to i-TRiLOGI’s attempts to communicate with it using the “NATIVE” mode. You will receive a communication error if you try to use i-TRiLOGI to access a PLC COMM port that has just been communicating in other protocol modes.

4. To improve the flexibility of switching from one COMM mode to another, The Wx100 incorporates a COMM mode self-reset timer such that a specific COMM mode will time out automatically and enters into “AUTO” mode after 10 seconds if no more commands are received from that COMM port. When a user wants to switch from one COMM mode to another, he/she often will be changing the serial connector from one device to another. During this time there is no data received by the COMM port, which presents an opportunity for it to reset its COMM mode. However, the surest way to reset the specific COMM mode is to cycle the power to the PLC so that its COMM port will be reset to “AUTO” mode and ready to communicate with any supported protocols.

5. If you wish to use the COMM port for serial data input only, you can use the SETPROTOCOL command to set the COMM port to NO PROTOCOL. This can prevent the PLC from erroneously treating some serial data as the header of an incoming communication protocol and respond to it automatically.

The SETPROTOCOL command can also be used to set the PLC to a specific protocol. This may be desirable if the COMM port has a specific role and you do not want it to enter other modes by mistake.

Please refer to the TBASIC Programmer’s Reference manual for a detailed description of the

SETPROTOCOL command.

Note:

If you fix a COMM port to a non-native, non-auto mode, TRiLOGI will not be able to communicate with the PLC anymore. You may have to power-cycle the PLC to reset the COMM mode. If you use the “1st.Scan” contact to activate the SETPROTOCOL command, then you will need to cycle the power to the PLC with DIP switch #4 set to ON to halt the execution of the SETPROTOCOL command. (Also remember that when the PLC is reset this way, its COMM1 will power up at default format of 38,400, 8,n,1 only so you will need to configure iTRiLOGI’s serial port to 38,400bps to communicate with it.)

14.3 Access RS485 From TBASIC

Besides responding automatically to specific communication protocols described in section 14.5, The WX100 RS485 serial ports is fully accessible by the user program using the TBASIC commands: INPUT$, INCOMM, PRINT # and OUTCOMM. It is necessary to understand how these commandsinteract with the operating system, as follow:

When a COMM port receives serial data, the operating system of the Wx100 automatically stores them into a 256 bytes circular buffer so that user programs can retrieve them later. The serial data are buffered even if they are incoming commands of one of the supported protocols described in section 14.3. In addition, processing of a recognized protocol command does not remove the characters from the serial buffer queue so these data are still visible to the user’s program.

The RS485 COMM port has its own separate 256-byte serial-in buffer. As long as the user-program retrieves the data before the 256-byte buffer is filled up, no data will be lost. If more than 256 bytes have been stored, the buffer wraps around and the oldest data is overwritten first and so on.

The following describes how INCOMM and INPUT$, PRINT # and OUTCOMM functions interact with the serial buffer:

a) INCOMM (n)
Every execution of the INCOMM(n) function removes one character from the circular buffer. When no more data is available in the buffer this function returns a -1. The data removed by INCOMM will no longer be available for the INPUT$ command.

b) INPUT$(n)
When the INPUT$(n) function is executed, the CPU checks the COMM #n buffer to see if there is a

byte with the value 13 (the ASCII CR character) which acts as a terminator for the string. If a string is present, all the characters that make up the string will be removed from the COMM buffer. If a completed string is not present, then the COMM buffer will not be affected, and INPUT$(n) returns a null string. This ensures that before a complete string is received, the serial characters will not be lost because of the unsuccessful execution of the INPUT$(n) function.

c) INPUT$(n) in Blocking Mode

INPUT$(n) is designed to be non-blocking and to return immediately – i.e. either it returns a complete string or it returns an empty string. This means that INPUT$(n) will not suspend the CPU and wait for a valid string from the COMM port. However, in real world communication very often you will need to send a command to a device and it takes a while for the device to be able to send back a response string.

The most efficient way of handling such serial communication exchange with other devices is to send a command string using the PRINT #n and then start a timer and let the PLC program continue to scan the ladder program. When timer times out, the timer contact is used to activate a custom function and use INPUT$(n) to read the return message from the device. This is most efficient use of the CPU time since the program will not have to waste time to wait for response string from the device and that's the reason for INPUT$(n) to default to non-blocking mode.

Another way to deal with this is to use the NETCMD$ function (terminated with "~"). NETCMD$ sends a command string and will return immediately when it receives a response string or if more than 0.15s has passed. If the device is slow to response the CPU basically sits in a loop to wait for the response for up to a maximum of 0.15s. NETCMD$ function also re-tries the communication several times if it does not receive a response in the first try.

A third way of handling serial exchange with other devices is to use the INPUT$(n) command in blocking mode. Starting from CPU firmware r72 and above, if you run the command

```
    SETSYSTEM 19, t
```

This will configure the INPUT$(n) command to block the CPU for up to maximum of (t x 10) milliseconds to wait for a valid string from 3rd party device. Although this command also wastes CPU cycles to wait for a response and hence it is not as efficient compared to using the timer-activated
method mentioned above, it nonetheless is very simple to implement and can be used for non time-critical applications.

You only need to execute SETSYSTEM 19, t once and INPUT$(n) will block for the same t x 10 millisecond every time it is executed until SETSYSTEM 19,t is run again. To return the INPUT$(n) to non-blocking mode, run SETSYSTEM 19, 0.

d) PRINT #n

The PRINT statement transfers its entire argument to a 256 byte serial-out buffer, which is separate from the serial-in buffer. The PRINT statement, therefore, does not affect the content of the serial buffer for incoming characters. The operating system handles the actual transfer of each byte of data out of the serial-out buffer in a timely manner.

Note that the PLC automatically enables the RS485 transmit driver when it sends serial characters out of its COMM 2 and 3 ports. When the stop bit of the last character in the serial-out buffer has been sent out, the operating system immediately disables the RS485 driver and enables the receiver. This greatly eases the use of the RS485 port since there is no need for a user to bother with the often-critical timing of controlling the RS485 driver/receiver direction.

e) OUTCOMM

This command sends only a single byte out of the serial COMM port without going through the serial out buffer. For RS485 port, it enables the RS485 transmitter before sending the character and disables it immediately after the stop bit has been sent out.

14.3 RS485 Primer

a) RS485 Network Interface Hardware

The built-in RS-485 interface allows the Wx100 PLC to be networked together using very low cost twisted-pair cables. Since the PLCs are fitted with a 1/8-power RS485 driver such as the 65HVD1785, up to 256 devices can be connected together. The twisted-pair cable goes from node to node in a daisy chain fashion. When communicating at the default baud rate of 38400 bps, the twisted pair cable should be terminated by a 120-ohm resistor if the cable length is longer than 300m (1000 ft). Terminating resistor is not needed for short distance communication.

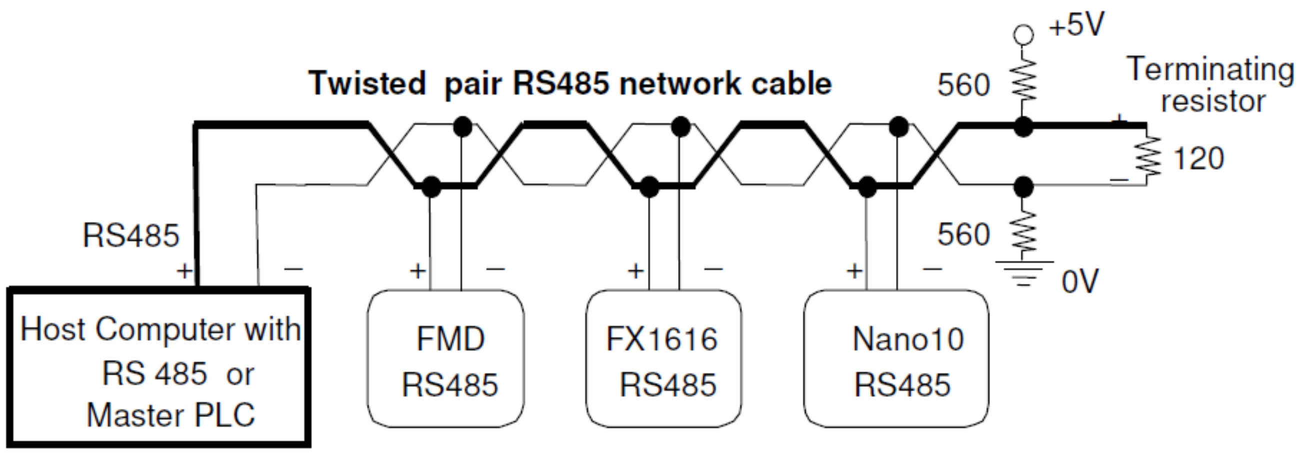


Figure 15.3.1

Note that the two wires are not interchangeable so they must be wired the same way to each controller. The maximum wire length should not be more than 1200 meters (4000 feet). RS-485 uses balanced or differential drivers and receivers, which means that the logic state of the transmitted signal depends on the differential voltage between the two wires and not on the voltage with respect to a common ground.

As there will be times when no transmitters are active (which leaves the wires in “floating” state), it is good practice to ensure that the RS-485 receivers will indicate to the CPUs that there is no data to receive. In order to do this, we should hold the twisted pair in the logic ‘1’ state by applying a differential bias to the lines using a pair of 560R to 1K biasing resistors connected to a +9V (at least +5V) and 0V supply as shown in Figure 15.3.1. Otherwise, random noise on the pair could be falsely interpreted as data.

The two biasing resistors are necessary to ensure robust data communication in actual applications. Some RS485 converters may already have biasing built-in so the biasing resistors may not be needed. However, if the master is a Wx100 PLC then you should use the biasing resistor to fix the logic states to a known state. Although in a lab environment the PLCs may be able to communicate without the biasing resistors, their use is strongly recommended for industrial applications.

Care should be taken to ensure that the power supplies for all the controllers are properly isolated from the main so that no large ground potential differences exist between any controllers on the network.

b) Single Master RS485 Networking Fundamentals

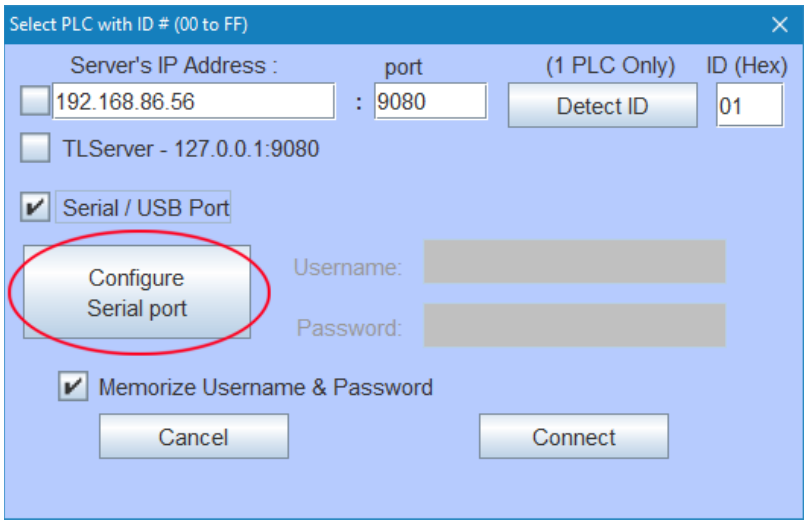
RS485 is a half-duplex network, i.e., the same two wires are used for both transmission of the command and reception of the response. Of course, at any one time, only one transmitter may be active. The Wx100 implements a master/slave network protocol. The network requires a master controller, which is typically a PC equipped with an RS485 interface. In the case of a PC, you can purchase a USB to RS-485 adapter card or an RS232C-to-RS485 converter and connect it to the RS232C serial port. A Wx100 can also be programmed to act as the master, it can communicate with other PLCs by executing the “NETCMD$” function or the “READMODBUS” or the “WRITEMODBUS” commands (the latter two are for communicating using MODBUS protocols only)

Only the master can issue commands to the slave PLCs. To transmit a command, the master controller must first enable its RS-485 transmitter and then send a multi-point command to the network of controllers. After the last stop bit has been sent, the master controller must relinquish the RS485 bus by disabling its RS485 transmitter and enabling its receiver. At this point the master will wait for a response from the slave controller that is being addressed. Since the command contains the ID of the target controller, only the controller with the correct ID would respond to the command by sending back a response string. For the network to function properly, <u>it is</u>

obvious that no two nodes can have the same ID.

There are two ways to change the PLC ID:

i) If you are connecting to the PLC via serial port only, you can click on the "Configure Serial Port" button screen in the login screen and then click on the "Change PLC ID" button to change the ID from 01 to FF (01 to F7 for Modbus)



Select PLC with ID # (00 to FF)

Server's IP Address : 192.168.86.56 : port 9080 (1 PLC Only) Detect ID ID (Hex) 01

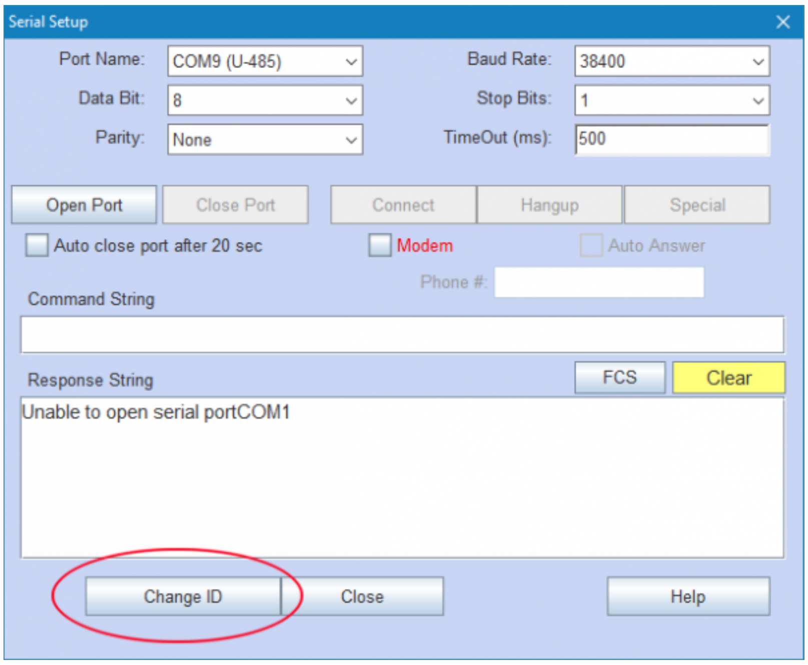
TLServer - 127.0.0.1:9080

[✓] Serial / USB Port

Username:

Password:

[✓] Memorize Username & Password



Serial Setup

Port Name: COM9 (U-485)
Baud Rate: 38400
Data Bit: 8
Stop Bits: 1
Parity: None
TimeOut (ms): 500

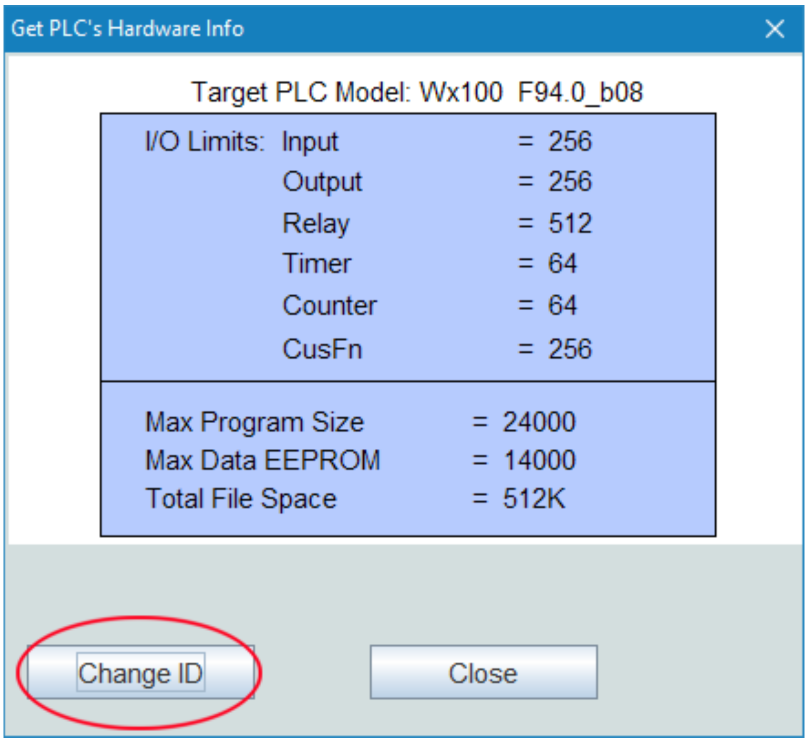
Auto close port after 20 sec
Modem
Auto Answer
Phone #:

Command String

Response String

Unable to open serial portCOM1

ii) If you are connected to the PLC via WiFi. Click "Controller->Get PLC Hardware Info" command in i-TRiLOGI software and then click on the "Change ID" button to make changes as shown below:


Get PLC's Hardware Info

Target PLC Model: Wx100 F94.0_b08

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

Max Program Size     =  24000
Max Data EEPROM      =  14000
Total File Space     =  512K
```

c) Multi-Master RS485 Networking

Since the Wx100 is capable of sending out network commands, the obvious question is whether multiple masters are allowed on the RS485 network? It is possible to have multiple masters on a single RS485 network provided the issues of collision and arbitration are taken care of.

Some RS485 protocols have arbitration or token passing features built-in, which automatically allow multiple masters over a single RS485 network. The 3 serial communication protocols that Wx100 currently supports, namely Hostlink protocol, Modbus ASCII and Modbus RTU do not have built-in arbitration and are hence are not suitable for multi-master type communication. Although there are ways and means to introduce such arbitration method into the standard protocol, you probably should instead consider communicating using TCP/IP via the WiFi if you need to have multiple PLCs talking to each other at arbitrary time.

TCP/IP communication is inherently multi-master, so any PLC can talk to any other PLC by establishing a virtual connection and then talk as if it has exclusive direct connection to the other PLCs. The Wx100 PLC can use either the Hostlink and Modbus TCP protocol to talk to another PLC, allowing true machine-to-machine communication easily and at a speed that is MUCH faster than serial RS485 communication. The TCP/IP communication capability therefore makes multi-master RS485 communication somewhat redundant today.

d) Trouble-Shooting An RS485 Network

Single faulty device

If a single device on the RS485 network becomes inaccessible, problems can be isolated to this particular device. Check for loose or broken wiring or wrong DIP switch settings. Also double check the device ID using the host-link command “IR*” sent via the serial link when only a single PLC is connected to the PC. If all attempts fail, either replace the PLC and try again.

Multiple faulty devices

If all the PLCs are inaccessible by the host computer, it may possibly be due to a faulty U-RS485 converter at the PC. If this is the case, disconnect the USB-RS485 converter from the network and check it using a single PLC. Replace the converter if it is confirmed to be faulty. Next check the wire from the converter to the beginning of the network. A broken wire here can lead to the failure of the entire network.

Since an RS485 network links many PLCs together electrically and in a daisy chain fashion, problems occurring along the RS485 network sometimes affect the operation of the entire network. For example, a broken wire at the terminal of one node may mean that all the PLCs connected after this node become inaccessible by the master. If the RS485 interface of one of the PLCs has short-circuited because of component failure, then the entire network goes down with it too. This is because no other node is able to assert proper signals on the two wires that are also common to the shorted device.

Hence, when trouble-shooting a faulty RS485 network, it may be necessary to isolate all the PLCs from the network. Thereafter, reconnect one PLC at a time to the network, starting from the node nearest to the host computer. Use the TRiLOGI program to check communication with each PLC until the faulty unit has been identified.

14.5 RS485 Applications

The following describes some possible uses of the RS485 ports.

a) Programming And Network Configuration

Besides programming via Wi-Fi, a Wx100 can also be programmed via its RS485 port on a one-to-one or multi-drop manner. Since most PCs do not have built-in RS485 port, you will need to purchase an optional USB-RS485 adapter (e.g. the U-485 (https://www.triplc.com/u485.htm) adapter that TRi supplies) in order to program the PLC via its RS485 port.

The i-TRiLOGI programming software now directly supports programming of the PLC via serial port. (In older version of iTRiLOGI software you will need to run the TLServer software to be used as a TCP-to-serial port gateway). This is especially useful should you encounter problems connecting the PLC to a WiFi network (e.g. you accidentally messed up the wireless key settings), or if you are unable to connect your PC's WiFi to the PLC even when it boots up in AP mode, then serial connection may be the last resort for you to connect to the PLC for quick programming or configure the network parameters.

b) Acting As a Modbus TCP WiFi to Modbus RTU Gateway

Wx100 PLC has a built-in Modbus TCP to Modbus RTU Gateway capability! What it means is that you can connect a Wx100 to a network of Modbus RTU slave devices that have only RS485 port and enable them to be accessible by a Modbus TCP master over WiFi. The Modbus TCP master could be a SCADA master or another intelligent device that speaks Modbus TCP but may have no direct physical connection to the Modbus RTU slave via RS485. Since a Wx100 can be connected to a WiFi network it provides a convenient means for the Modbus TCP master to accessible these slave devices over the air.

To enable the Modbus TCP to RTU features in the Wx100, run the following command once during initialization:

```
    SETSYSTEM 12, 1    ' relay Modbus TCP messages through COMM1 on the PLC
```

Then set the PLC ID to be different from all the slave Modbus RTU devices connected to its RS485 port. That's all!

Once enabled, any Modbus TCP packet that is addressed to an ID which does not belong to the PLC itself will be relayed out of the RS485 serial port to be received by slave Modbus RTU devices. A response packet received from the RS485 port will in turn be sent back to the Modbus TCP master via the WiFi port and thus completing the command/response packet.

Note: The Modbus TCP to RTU gateway operates in the background without needing any direct intervention by the PLC program at all. Best of all, the PLC can continue to execute its control program even when the Modbus TCP-to-RTU gateway function is ongoing in the background. So you essentially get a free gateway while still using the full power of the PLC to control your equipment.

c) Interfacing non-Mobdus Devices to Modbus Host or to the Internet

Since the Wx100 supports MODBUS protocols, it can operate as a master PLC and serve as a gateway to interface non MODBUS-enabled PLCs (such as the E10+ PLCs, or the I-7000 analog modules) to third party SCADA masters or MMI hardware that speaks MODBUS TCP or MODBUS RTU. The Wx100 also makes it easy for these devices to be controlled or monitored over the Internet. The master Wx100 will use its RS485 port to pull data from these devices into its data-memory. The data memory in the Wx100 is in turn accessible by a SCADA program using the Modbus serial or Modbus/TCP protocol. Through the Wx100, these other connected devices are also accessible from the Internet.

d) Accessing 3rd Party RS485-based Devices

There are more and more industrial devices such as electric power meters, analog I/O modules (e.g. the I-70xx modules by ICPDAS), variable frequency drives, etc that allows data communication via their RS485 port. The Wx100 PLC has many built-in commands for reading/writing to these serial ports, including built-in commands for communicating with devices that use the MODBUS ASCII or RTU protocols.

e) Distributed Control

Another important use of the RS485 port will be to connect a Wx100 to other TRi PLCs. One Wx100 will act as the master and all other PLCs will act as slaves. Each PLC must be given a unique ID. The master will send commands to all the slaves using the “NETCMD” or READMODBUS, WRITEMODBUS, READMB2, WRITEMB2 statements and coordinate information flow between the PLCs. In this way, a big system can be built by employing multiple units of Wx100, FMD, Fx or Nano-10 PLCs connected in a network. This results in more elegant implementation of complex control systems and simplifies maintenance jobs.

14.6 Wx100 Modbus/Omron RTU

The Wx100 PLC supports a subset of the OMRON™ C20H and MODBUS™ (Both ASCII and RTU modes are supported) compatible communication protocols so that it can be easily linked to third-party control software/hardware products such as SCADA software, LCD touch panels etc. The PLC automatically recognizes the type of command format and will generate a proper response. These are accomplished without any user intervention and without any need to configure the PLC.

Both MODBUS and Omron protocols use the same device ID address (00 to FF) as used by the native “Hostlink Command” protocol described in Chapter 15. Since the addresses of I/O and internal variables in the Fx2424 are organized very differently from the OMRON or Modicon PLCs, we need to map these addresses to the corresponding memory areas in the Modbus address space so that they can be easily accessed by their corresponding protocols.

All I/Os, timers, counters, internal relays, data memory DM[1] to DM[4000], and floating point data FP[1] to FP[1000] are mapped to the Modbus Holding Registers space. The Inputs, Outputs, Relays, Timers and Counters bits are mapped to the MODBUS Bit address space as shown in Table 14.1. Note that input and output bits are always mapped according to Table 14.1 whether it is MODBUS function 01, 02 or 05.

However, 32 bit variables and string variables are not mapped since they are fundamentally quite different in their implementation among different PLCs. Internal variables that are not mapped can be still be accessed by copying the contents of these variables to unused data memory DM[n] (this can be easily accomplished within a CusFn) so that they can be accessed by these third party protocols.

a) MODBUS ASCII Protocol Support

The Wx100 supports MODBUS ASCII protocols with the following command and response format:

START	Address	Function	Data	LRC Check	CRLF
:	2 chars	2 chars	# chars	2 chars	2 chars

The following Function Codes are supported:

01/02	Read I/O bit (Use Bit Address Mapping in Table 14.1)
03/04	Read I/O Word registers
05	Force I/O Bit (Use Bit Address Mapping in Table 14.1).
06	Preset Single Word Register
16	Preset Multiple Word Registers

The exact command/response format of the MODBUS protocol can be found at http://www.modbus.org. However, if your only purpose is to interface the PLC to other MODBUS hosts such as an LCD touch panel or SCADA software then there is no need to know the underlying protocol command format. All you need to know is which PLC’s system Variable is mapped to which MODBUS register.

Please refer to Section 2.5 (https://docs.triplc.com/wx100-um/#2979) and Table 2.5.1 (https://docs.triplc.com/html/table251.htm) for the memory mapping of Wx100 I/Os and internal data to MODBUS Register.

Bit Address Mapping

All the Wx100 I/O bits are mapped identically to both the Modicon “0x” and 1x space. The Modicon bit register offset is shown in the last column of Table 2.5.1. Although Modicon names the “0x” address space as “Coil” (which means output bits) and the “1x” address space as “Input Status” (which means input bits only), the Wx100 treats both spaces the same. Some Modbus drivers only allow a “read” from 0x space and a “write” to 1x space but you still use the same offset shown below on Table 2.5.1 (https://docs.triplc.com/html/table251.htm)

Example:

1. To read from the PLC Input 5 via Modbus, you select the Modicon bit address 0-0005.
2. To read from the PLC’s output #2, you may have to specify Modicon register address 0-0258 and to write to PLC output #2, you may have to specify Modicon register address 1-0258. However, if your Modbus TCP client allows reading and writing to either 0-xxxx and 1-xxxx space, then you can use either 0-258 or 1-258 and the action will be identical.

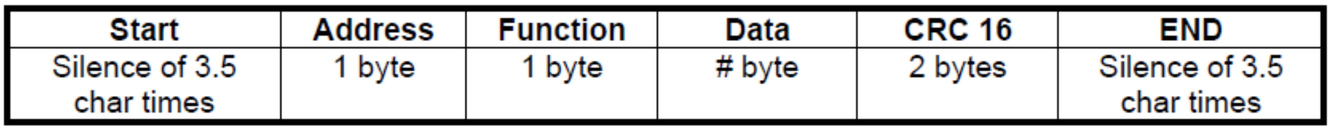
Word Address Mapping

As shown in Table 2.5.1, to access the PLC’s DM[1], you use MODBUS address space 4-1001 (The address naming scheme by Modicon. For binary addressing use 1000 (dec, or &H03E8) and so on. To access the Real Time Clock Hour data (TIME[1]), use 4-0513. The I/O channels can also be read or written as 16-bit words by using the addresses from 4-0001 to 4-0320.

Some MODBUS drivers (such as National Instruments “Lookout” software) even allow you to manipulate individual bits within a 16-bit word. So it is also possible to map individual I/O bits to the “4x” address space. E.g. Input bit #1 can be mapped to 4-0001.1 and output bit #2 is mapped to 4-0257.2, etc. This is how it is shown in Table 2.5.1. However, if you do not need to manipulate the individual bit, then you simply use the address 4-0001 to access the system variable INPUT[1] and address 4-0257 to access the system variable OUTPUT[1]. Note that INPUT[1] and OUTPUT[1] are TBASIC system variables and they each contain 16 bits that reflect the on/off status of the actual physical input and output bits #1 to #16.

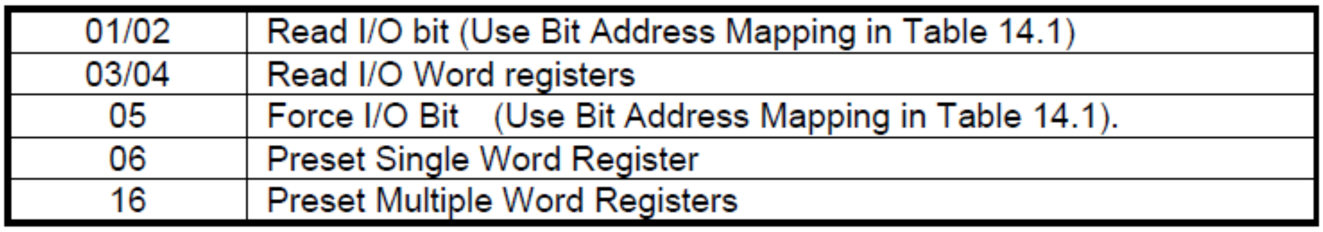
b) MODBUS RTU Protocol Support

The Wx100 also support the MODBUS RTU protocol. The difference between the ASCII and RTU protocols is that the latter transmits binary data directly instead of converting one byte of binary data into two ASCII characters. A message frame is determined by the silent interval of 3.5 character times between characters received at the COMM port. Other than that, the function codes and memory mappings are identical to the MODBUS TCP and Modbus ASCII protocol.



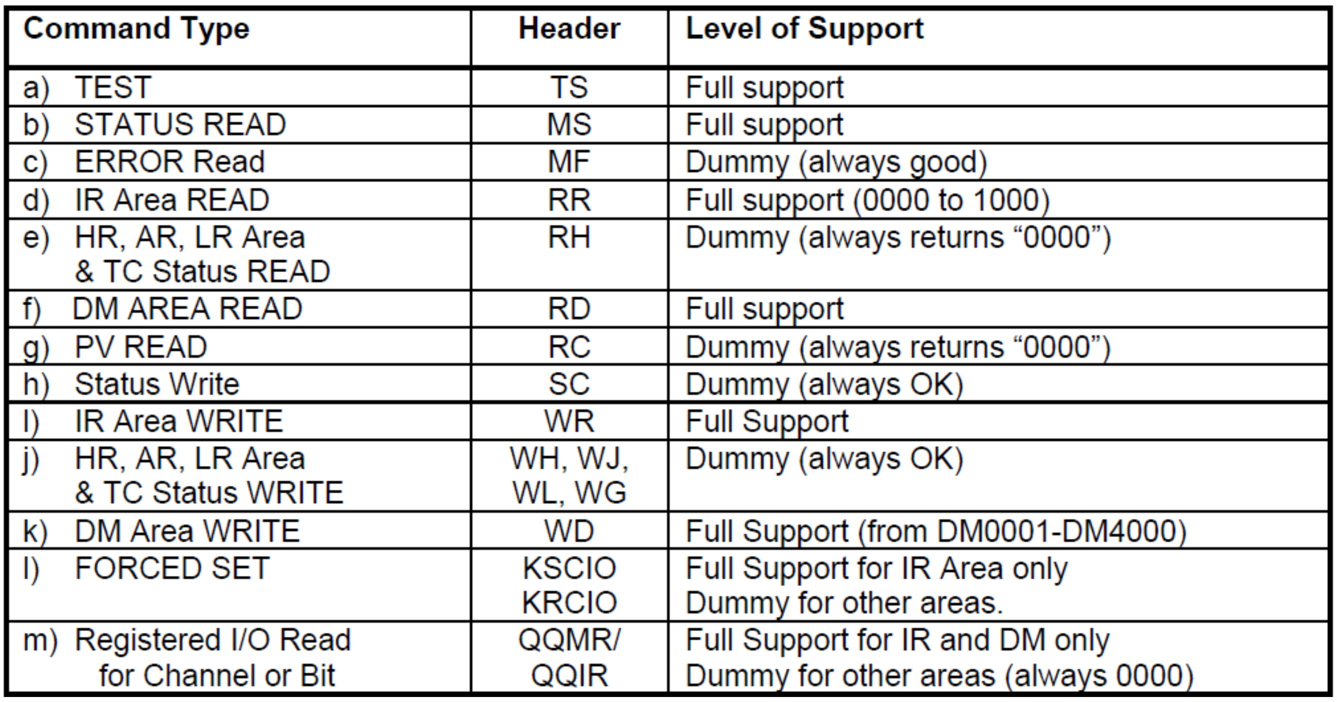
Start	Address	Function	Data	CRC 16	END
Silence of 3.5 char times	1 byte	1 byte	# byte	2 bytes	Silence of 3.5 char times

The following Function Codes are supported:



01/02	Read I/O bit (Use Bit Address Mapping in Table 14.1)
03/04	Read I/O Word registers
05	Force I/O Bit (Use Bit Address Mapping in Table 14.1).
06	Preset Single Word Register
16	Preset Multiple Word Registers

c) OMRON Host Link Command Support



Command Type	Header	Level of Support
a) TEST	TS	Full support
b) STATUS READ	MS	Full support
c) ERROR Read	MF	Dummy (always good)
d) IR Area READ	RR	Full support (0000 to 1000)
e) HR, AR, LR Area & TC Status READ	RH	Dummy (always returns "0000")
f) DM AREA READ	RD	Full support
g) PV READ	RC	Dummy (always returns "0000")
h) Status Write	SC	Dummy (always OK)
I) IR Area WRITE	WR	Full Support
j) HR, AR, LR Area & TC Status WRITE	WH, WJ, WL, WG	Dummy (always OK)
k) DM Area WRITE	WD	Full Support (from DM0001-DM4000)
l) FORCED SET	KSCIO KRCIO	Full Support for IR Area only Dummy for other areas.
m) Registered I/O Read for Channel or Bit	QQMR/ QQIR	Full Support for IR and DM only Dummy for other areas (always 0000)

Some OMRON host link commands are described in Section 16.40. For other commands please refer to the Omron C20H/C40H PLC Operation manual published by OMRON Corporation. If your purpose is only to use the PLC's OMRON mode with SCADA or HMI, then there is no need to learn the actual command/response format.

d) Wx100 As a Modbus Master – Getting Data From Power or Flow Meters

The Wx100 support for the MODBUS protocol goes beyond being a MODBUS slave only. You can use the TBASIC READMOBUS and WRITEMODBUS commands, as well as READMB2 and WRITEMB2 to send out MODBUS ASCII or RTU commands to access any other Wx, Fx, FMD or Nano-10 PLCs or any third party MODBUS slave devices. The READMODBUS or READMB2 commands use MODBUS Function 03 (this can be changed to function 04 using SETSYSTEM 6,4 command) to read from the slave, and WRITEMODBUS or WRITEMB2 use MODBUS Function 16 to write to the slave.

Note that when using the READMODBUS or WRITEMODBUS commands, the 40001 address stated in Table 2.5.1 (https://docs.triplc.com/html/table251.htm) should be interpreted as address 0000, and 40002 as address 0001, and 41001 as address 1000, etc. This is in accordance with the specifications stated in MODBUS protocol. MODICON defined zero offset addresses for the MODBUS protocol, yet in their holding register definition these are supposed to start from address 40001 – hence the unusual correspondence. But to maintain compatibility with the MODBUS specifications, we have to adhere to their definitions.

Sending Modbus ASCII or a Modbus RTU Commands

The Wx100 PLC can act either as a MODBUS **ASCII** or MODBUS **RTU** master.

You use the same READMODBUS and WRITEMOBUS commands to send and receive MODBUS ASCII or MODBUS RTU commands. The only difference being the comm port number you use with these commands. The RS485 port on Wx100 is assigned COMM port #1. If you run the MODBUS command via COMM port #1 then it will send out MODBUS ASCII command by default.

To send MODBUS RTU command, what you need to do is add 10 (decimal) to the COMM port number to signal to the processor that you wish to use MODBUS RTU instead of MODBUS ASCII to talk to the slaves.

In other words, you should specify port #11 to use RTU commands on COMM1 port..

E.g. the statement DM[10] = READMODBUS (11, 8, 16) will access, via COMM1, the slave with ID = 08 and read the content of register #16. This register corresponds to MODICON address 40017 and is the OUTPUT[1] of the slave PLC. The ability to speak MODBUS RTU greatly extends the type of peripherals that can be used with a Wx100 PLC. You can now make use of many off-the-shelf, third party RTU devices to extend the PLC
capability.

Chapter 15 - HostLink Protocol

While a Wx100 is running, it may receive ASCII string commands that read or write to its inputs, outputs, relays, timers, counters, and all the internal variables from a host computer or another Wx, Fx, FMD or Nano-10 PLC. These ASCII commands are known as the “Host-link commands” and are to be serially transmitted (via RS232C or RS485 port) to and from the controller. The default serial port settings of the Wx100 PLC for host-link communication are: 38400 baud, 8 data bit, 1 stop bit, no parity. The baud rate and the communication format may be changed using the “**SetBAUD**” TBASIC command described in the i-TRiLOGI Programmer’s Reference.

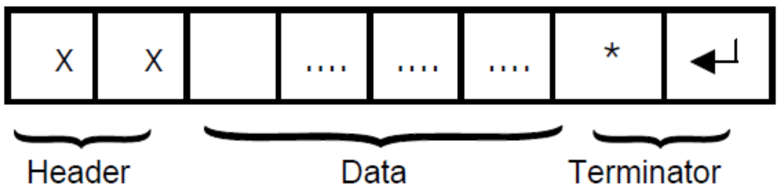
15.1 Point-To-Point Protocol

When the Wx100 receives a native host-link command via its RS485 port, it will automatically send a response string corresponding to the command. This operation is totally transparent to the user and does not need to be handled by the user’s program.

All Wx100 support both the point-to-point (one-to-one) and multi-point (one-to-many) communication protocols. Each protocol has a different command structure as described below.

In a point-to-point communication system, the host computer’s RS232C serial port is connected to the PLC’s COMM1. At any one time, only one controller may be connected to the host computer. The hostlink commands do not need to specify any controller ID code and are therefore of a simpler format, as show below:

Command/Response Frame Format (Point to Point)



Each command frame starts with a two-byte ASCII character header, followed by a number of ASCII data and ends with a terminator which is comprised of a ‘*’ character and a carriage return (ASCII value =13 decimal). The header denotes the purpose of the command. For example, RI for Read Input, WO for Write Output, etc. The data is usually the hexadecimal representation of numeric data. Each byte of binary data is represented by two ASCII characters (00 to FF).

To begin a communication session, the host computer must first send one byte of ASCII character: Ctrl-E (=05Hex) via its serial port to the controller. This informs the controller that the host computer wishes to send a (point-to-point) host-link command to it. Thereafter, the host computer must wait to receive an echo of the Ctrl-E character from the controller. Reception of the echoed Ctrl-E character indicates that the controller is ready to respond to the command from the host computer. At this moment, the host computer must immediately send the command frame to the controller and then wait to receive the response frame from the controller. The entire communication session is depicted in Figure 15.1.

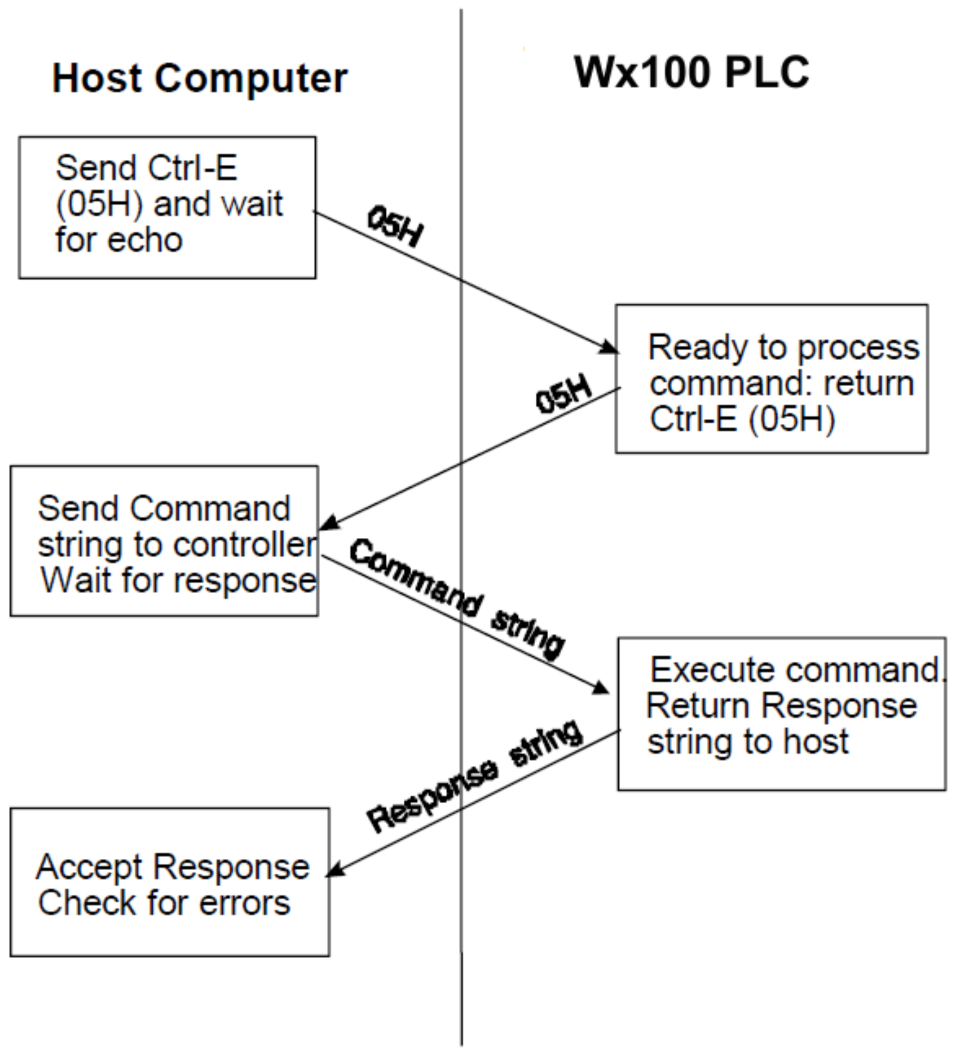

Host Computer
Wx100 PLC
Send Ctrl-E (05H) and wait for echo
05H
Ready to process command: return Ctrl-E (05H)
05H
Send Command string to controller Wait for response
Command string
Execute command. Return Response string to host
Response string
Accept Response Check for errors

Figure 15.1

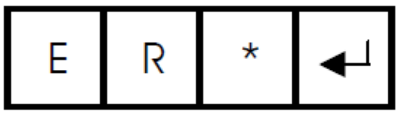
After the controller has received the command, it will send a response frame back to the host computer and this completes the communication session. If the controller accepts the command, the response frame will start with the same header as the command, followed by the information that has been requested by the command and the terminator.

As you can probably see, proper handshaking using the Ctrl-E character between the host and the PLC is important to communicate successfully using the Point-to-point protocol. Although the "Multi-point" format discussed in the next section seems more complex, the communication exchange using multi-point protocol is actually simpler than point-to-point since it involves only a single exchange of command/response string. We therefore recommend using the multi-point format if you are writing your own communication program.

Note: TBASIC has a built-in command "NETCMD$" that lets a Wx100 PLC access another slave PLC using the multipoint Host-link protocol format very easily.

If an unknown command is received or if the command is illegal (such as access to an unavailable output or relay channel), the following error response will be received:

Error Response Format (Point-to-point)

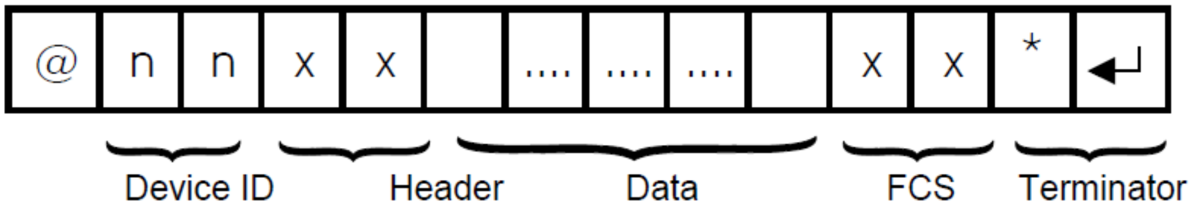


The host computer program should always check the returned response for possibilities of errors in the command and take necessary actions.

15.2 Multi-point Protocol

In this system, one host computer may be connected to either a single PLC (via either RS232 or RS485) or multiple PLCs on an RS485 network.

a) Command/Response Frame Format (Multi-point)

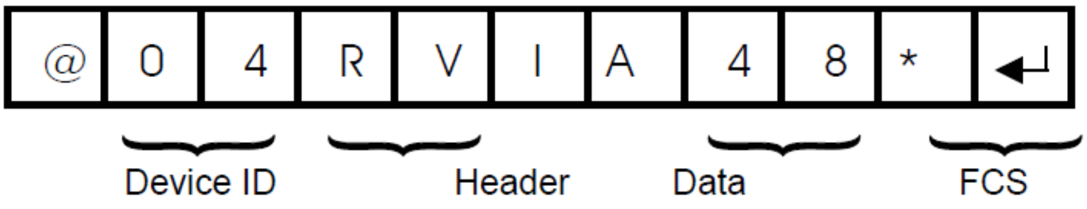


Each command frame starts with the character “@” and two-byte hexadecimal representation of the controller’s ID (00 to FF), and ends with a two-byte “Frame Check Sequence” (FCS) and the terminator. FCS is provided for detecting communication errors in the serial bit-stream. If desired, the command frame may omit calculating the FCS simply by putting the characters “00” in place of the FCS.

Note: we call “00” the “wildcard” FCS, which is available when the PLC is in “auto protocol” mode. This is to facilitate easy testing of the multi-point protocol. However, the wildcard FCS can be disabled if the PLC has executed the SETPROTOCOL n, 5 to put its COMM port n into pure native mode. In that case you will have to supply the actual FCS to your command string.

b) Calculation of FCS

The FCS is 8-bit data represented by two ASCII characters (00 to FF). It is a result of performing an Exclusive OR on each character in the frame sequentially, starting from @ in the device number to the last character in the data. An example is as follows :



```
@     0100 0000
      XOR
0     0011 0000
      XOR
4     0011 0100
      XOR
R     0101 0010
      XOR
V     0101 0110
      XOR
 I    0100 1001
      XOR
A     0100 0001
_______________________________
      0100 1000   = 48₁₆
```

Value 48_{16} is then converted to ASCII characters ‘4’ (0011 0100) and ‘8’ (0011 1000) and placed in the FCS field.

FCS calculation program example

The following C function will compute and return the FCS for the “string” passed to it.

```
unsigned char compute_FCS(unsigned char *string){
unsigned char result;
result = *string++; /*first byte of string*/
while (*string)
    result ^= *string++; /* XOR operation */
    return (result);
}
```

A **Visual Basic** routine for FCS computation is included in the source code of a sample communication program you can download from:

http://www.tri-plc.com/applications/VBsample.htm#VB6sample (http://www.tri-plc.com /applications/VBsample.htm#VB6sample)

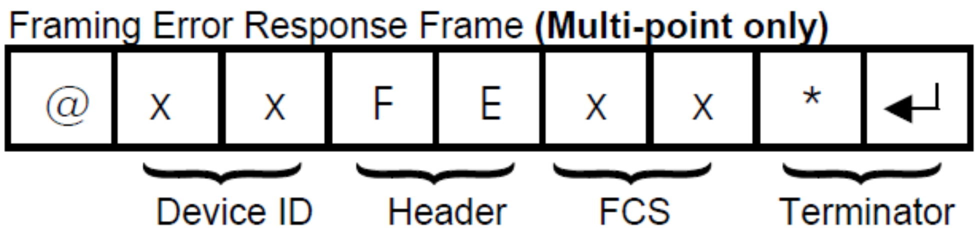
c) Communication Process

Unlike the point-to-point communication protocol, the host computer must **NOT** send the CTRL-E character before sending the command frame. After the host computer has sent out the multi-point host-link command frame, only the controller with the correct device ID will respond. Hence it is essential to ensure that every controller on the RS485 network assumes a different ID (If a master PLC is used, then the master PLC should also have a different ID from all the slaves). Otherwise, contention may occur (i.e., two controllers simultaneously sending data on the receiver bus, resulting in garbage data being received by the host). On the other hand, if none of the controller IDs match that specified in the command frame, then the host computer will receive no response at all.

If the PLC is operating in "auto-protocol" mode (default), it will automatically recognize the type of command protocols (point-to-point or multi-point) sent by the host computer and it will respond accordingly. If a multi-point command is accepted by the controller, the response frame will start with a character '@', followed by its device ID and the same header as the command. This will be followed by the data requested by the command, a response frame FCS and the terminator.

d) Framing Errors

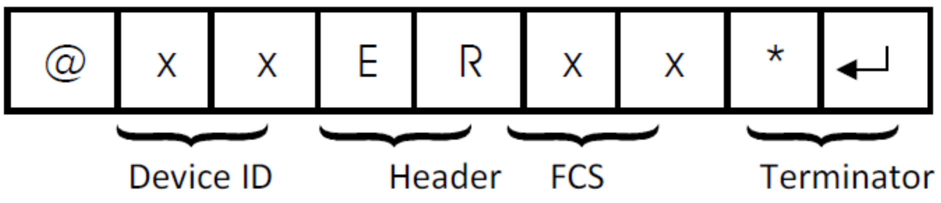
When the controller receives a multi-point host-link command frame, it computes the FCS of the command and compares it with the FCS field received in the command frame. If the two do not match, then a "framing error" has occurred. The controller will send the following Framing Error Response to the host:



e) Command Errors

If an unknown command is received or if the command is illegal (such as an attempt to access an unavailable channel), the following **error response** will be received:

Error Response Format



d) SHOULD YOU USE POINT-TO-POINT OR MULTI-POINT PROTOCOL?

Although at first the point-to-point protocol appears simpler in format (having no ID and no FCS computation), the communication procedure is actually more complex since it involves the need to synchronize the two communicating devices by exchanging the Control-E character. The lack of error checking also makes the protocol less reliable especially in noisy environment.

Hence, if you were to write your own communication program to talk to the PLCs, we certainly strongly recommend using only the multi-point protocol exclusively due to its simplicity and built-in error checking capability. The "IR*" point-to-point protocol may still need to be used to retrieve the ID of a PLC with unknown ID (and you can only run this with a single PLC attached to the PC).

Chapter 16 - Hostlink Reference

Please visit the HostLink Protocol Reference Manual at:

https://docs.triplc.com/hostlink/ (https://docs.triplc.com/hostlink/)

Chapter 17 - File System Support

17.1 Introduction

We have earlier described in Chapter 2.9 (https://docs.triplc.com/wx100-um/#3059) that a Wx100 PLC has 512K bytes of file space that can be used for storing control web pages that a web browser can load to perform control operation.

In addition, the PLC can also open or create a data file for reading, writing or appending. This allows the PLC to log operational- or maintenance-related data into data file for non-volatile storage. Once a data file has been updated, there are two ways to retrieve the stored data files from the PLC:

1. Download the file from the PLC's built-in web server: The file created by the PLC can be downloaded from the PLC's built-in web server using any web browser. This allows the user to access the data file at any time of the day.
2. Automatic FTP upload from the PLC to an external web-server: You can program the Wx100 PLC to make a FTP client connection to any web server on the local network or on the Internet/Cloud and upload the data file it has created to the web-server using any filename.

Imagine what the Wx100 can do with this file uploading capability! The ability to log data locally and automatically upload the data to a web server transforms the PLC into a potent data-logger! The PLC can be programmed to capture daily, weekly or monthly data and then periodically upload the data file to an Internet web server with a unique, time-stamped filename (E.g. "temperaturelog2012-01-01.csv"). This allows the PLC to log data completely unattended.

The data uploaded by the PLC to the external web server can therefore be viewed or downloaded into a PC using any web browser, anywhere in the world. This allows you to carry out analysis of past logged data file for performance or diagnostic analysis at any time without having to physically access the PLC to retrieve the logged data.

17.2 Data Uploading Benefit

1. Although it is possible to directly access the PLC's internal web server to download the data file it has created, this does require active action by the user and to ensure that the data are retrieved before the file is full and deleted by the PLC to create space to log new data.

2. By programming the PLC to upload the data periodically the PLC can delete the file after it has successfully uploaded the data file to free up space to accept new data. In other words the PLC will never run out of data space to log data since it can store the logged files on any server including the Cloud!

3. To directly access the file stored on the PLC from outside of the LAN, you will need to setup the router or firewall to "forward" the PLC's server port (e.g. 9080) to the PLC. If you have multiple PLCs logging data, then each PLC will need to have a different port number in order to properly forward the data. This not only complicates the setup, but also is often frown upon by System Administrator and may not even be permitted by the corporate network security policy.

4. The PLC is designed to upload data to any web server via FTP passive mode by providing the

login username and password. Using FTP passive mode allows the PLC to open a network connection to an external web-server to upload a file and then close the connection immediately. It does not require opening a port on your router to permit external access to the PLC from the Internet. Hence there is no complicated router setup involved as there is no port forwarding required. It also eliminates the security risk from someone trying to take control of the PLC from outside of your LAN and is generally much more acceptable to the System Administrator.

If you have multiple PLCs in use, you can program each PLC to upload data to a different directory or append a different file name prefix, or to a different server, and once programmed all PLCs will happily log data unattended indefinitely!

17.3 File Structure & File Name

Wx100 does not support directory structure. All files are therefore always stored in the root directory. File name on a Wx100 can be a combination of alphanumeric characters **excluding white space** and not more than 31 characters in length. You can store multiple files with different names as long as the total space used up by the file is less than 512K bytes.

Wx100 FTP uploading capability is quite flexible and you can program the PLC to upload the file to a FTP server using a completely different name from the actual name of the file to be uploaded.

Note:

> The file naming system on the F-series Super PLCs family, namely Nano-10, FMD and SmartTILE-Fx based PLCs is more restrictive which limits the file names to only “A.xxx” to “Y.xxx” and the files that PLC can directly read/write to only between “Z000.xxx” to “Z399.xxx”. The file system on Wx100 is more advanced and have no such restrictions. It also mean that the file names that work on the FMD, Fx PLC will work without modification on Wx100.However, if you intend to make your Wx100 program to also work on the other PLC models mentioned above, then you should keep the file naming method backward compatible across all PLC models, which means you should understand the file naming structure of the other PLC models as well.
>
> Most of the file-related sample programs you find in your “\TRiLOGI\TL6\usr\samples” and “\TRiLOGI\TL7\usr\samples” folders are written to work with the F-series PLC with their file naming scheme in mind but will work without modification the Wx100.

17.4 File Access By TBASIC

The PLC can open a new file for writing new data (essentially deleting the old file content), or open an existing file and append data to the end of the file. It can also open a file and read data from the file as ASCII strings. It can achieve this by using the **PRINT #8** and **INPUT$(8)** functions, which will be described in details in the following sections. We have created a sample program: "ExtendedFileSystem.PC6" (click here: http://www.tri-plc.com/trilogi/ExtendedFileSystem.zip (http://www.tri-plc.com/trilogi/ExtendedFileSystem.zip) to download) that demonstrates all these capabilities.

a) Open A File For Writing New Data

Syntax: **PRINT #8 "<WRITE** *filename***>"**

If successfully executed, the "<WRITE>" command will open the file and set the file pointer to the beginning of the file. Thereafter the PLC can start writing ASCII data to the file using the PRINT #8 <string data> command. [Note: the PRINT #8 command automatically appends a carriage return to the end of the string data unless the string data is terminated with a semi-colon (';')].

When the PLC has completed writing data, it must close the file by executing the command: PRINT #8 "</>".

E.g.

```
PRINT #8  "<WRITE Z005.TXT>"
PRINT #8  "The current Greenwich Mean Time is"
PRINT #8  STR$(TIME[1]);":";STR$(TIME[2]);":";"00"
PRINT #8  "</>"
```

The CPU should use the STATUS(2) command to check whether the <WRITE> has been successfully executed before begin writing data to it. STATUS(2) command returns a 1 if "<WRITE>" operation is successful and returns a '0' if the operation failed. The CPU can only write or access to a single file at a time so any opened file must be closed by the PRINT #8 "</>" command before another file can be opened for writing.

b) Open A File For Appending Data To The End Of The File

Syntax: **PRINT #8 "<**APPEND *filename***>"**

If successfully executed, the "<APPEND>" command will open the file and set the file pointer to the end of the file. Thereafter any string data following a PRINT #8 command will be appended to the end of the file. When the PLC has completed appending data, it must close the file by executing the command: **PRINT #8 "</>"**.

As per the "<WRITE>" command, the CPU should also use the STATUS(2) command to check whether the <APPEND> command has been successfully executed before begin writing data to it.

Example

```
PRINT #8  "">"
S = STATUS(2)       ‘ Status(2) returns 1 if successful.
IF S <> 1 RETURN: ENDIF
FOR I = 1 to 100
   PRINT #8  STR$(I,4)+":This is the Appended first line"
   PRINT #8  STR$(I,4)+":This is the Appended second line"
   SETLCD 1,1, "Append  #"+STR$(I,4)
NEXT
PRINT #8  "</>"     ‘ close the file
```

c) Open A File For Appending Data To The End Of The File

Syntax: **PRINT #8 "<DELETE** *filename***>"**

There is no need to close a deleted file.

d) Open A File For Reading Data

Syntax: **PRINT #8 "<READ** *filename***>"**

If the file has been successfully opened for reading after execution of the PRINT #8 "<READ>" command, the PLC can start to retrieve ASCII data from the file line-by-line using the INPUT$(8) command. A line is either a string that is terminated with a Carriage Return character (ASCII 13), or is a 70-character long string (which is the maximum length of any string variables A$ to Z$) without carriage return. In either case the return string does not contain the CR character itself.

The PLC can check if a file has been successfully opened for reading using the STATUS(2) function **AFTER** executing the PRINT #8 "<READ>" command. STATUS(2) will only return a 1 if a file has been successfully opened.

The PLC can determine if the End-of-File (EOF) has been reached using the STATUS(2) function after every INPUT$(8) command has been executed. STATUS(2) returns a 255 if the EOF has been reached. The PLC should then close the file by executing the "PRINT #8 "</>" command.

```
A$ = ""
S = STATUS(2)
IF S <> 0
   SETLCD 1,1, "Failed to Open File"
   GOTO @100
ENDIF
C = 0
PRINT #8 A$
SETLCD 1,1, A$
WHILE 1
   A$ = INPUT$(8)
   S = STATUS(2)
   IF S = 255 EXIT : ENDIF   ' S = 255 means EOF
   SETLCD 2,1,A$
   DELAY 50       ' So that reader can read from the screen.
   C = C+1
ENDWHILE

SETLCD 1,1, "Read ” +STR$(C) + " lines "
@100
PRINT #8 "</>"   ' close the opened file
```

17.5 Setup FileZilla FTP Server

One important capability of the new Wx100 PLC is the ability to upload file created by the PLC to an external server on a local area network or on the Internet via the FTP protocol. If you have access to a FTP username and password on your company's server (or if the SysAdmin is authorized to set up an account for you) you can certainly use your own account for testing. If not, you can download the free Filezilla FTP **Server** and set it up for testing. The "ExtendedFileSystem.PC6" has a FTP upload demo and it was configured to work with a FileZilla FTP server.

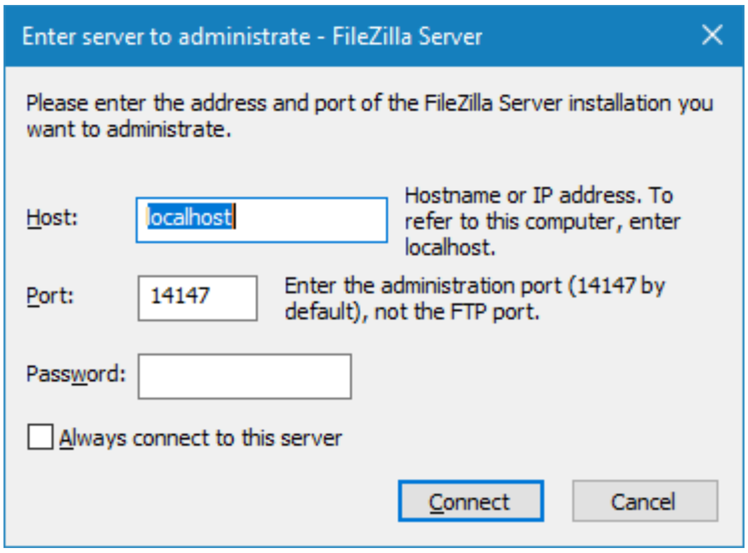
Using Filezilla has the advantage that you can see the login sequence performed by the PLC when it attempts to connect to the FTP Server so that it is easier to troubleshoot connection problem. (For professional grade troubleshooting, one handy program to have is the "Wireshark" program which is a TCP/IP packet sniffer that allows you to look at the actual TCP/IP packets sent between your PC and the PLC). However, it is important to setup the Filezilla server program properly to minimize connection trouble.

a) Download and Setup FTP Server

1. First download the FileZilla server installer from the following website:

`http://filezilla-project.org/download.php?type=server (http://filezilla-project.org/download.php?type=server)`

2. Run the “FileZilla Server Interface” program which is meant for managing the FTP Server settings.



Enter server to administrate - FileZilla Server

Please enter the address and port of the FileZilla Server installation you want to administrate.

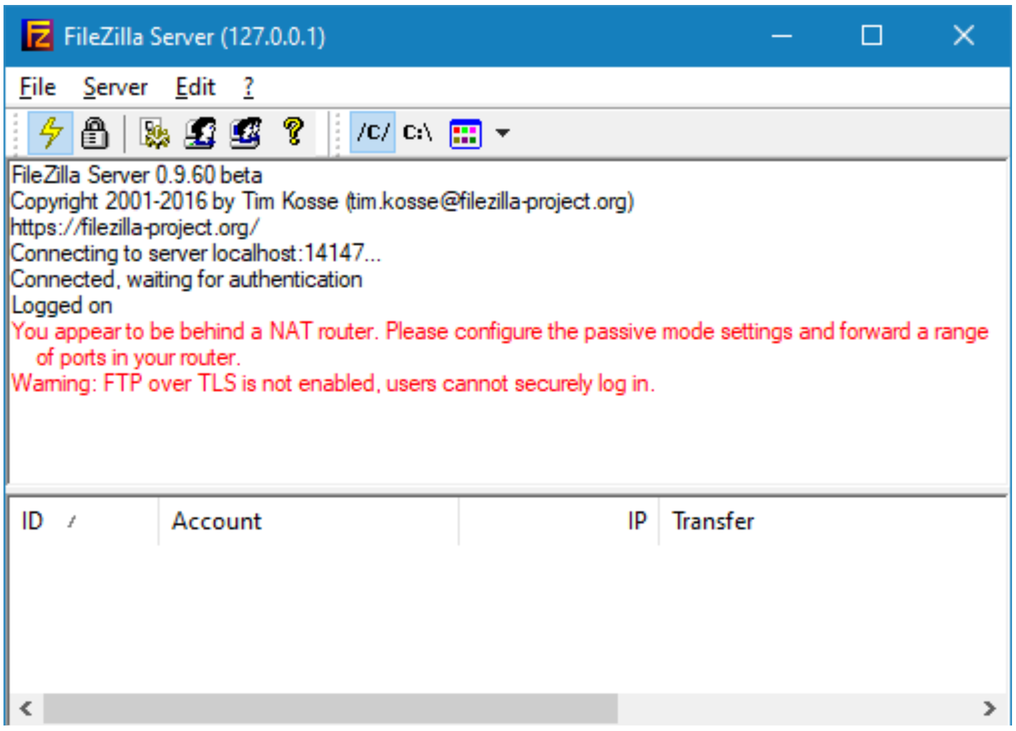
Host: localhost — Hostname or IP address. To refer to this computer, enter localhost.

Port: 14147 — Enter the administration port (14147 by default), not the FTP port.

Password:

3. If the FileZilla Server is running on the same PC that you are running the FileZilla Server Interface program then you can use the localhost IP address which is 127.0.0.1 – the Port can be anything since this is a client port that the Interface program is using to interact with the FTP Server (don’t be confused with the FTP **Server** listening port which is by default = 21).

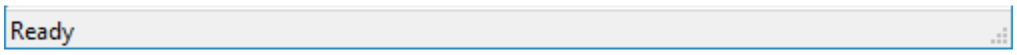
4. If this is the first time you run the program after setting up Filezilla FTP Server there will be no Administration password so you can leave it blank. Click OK to connect.



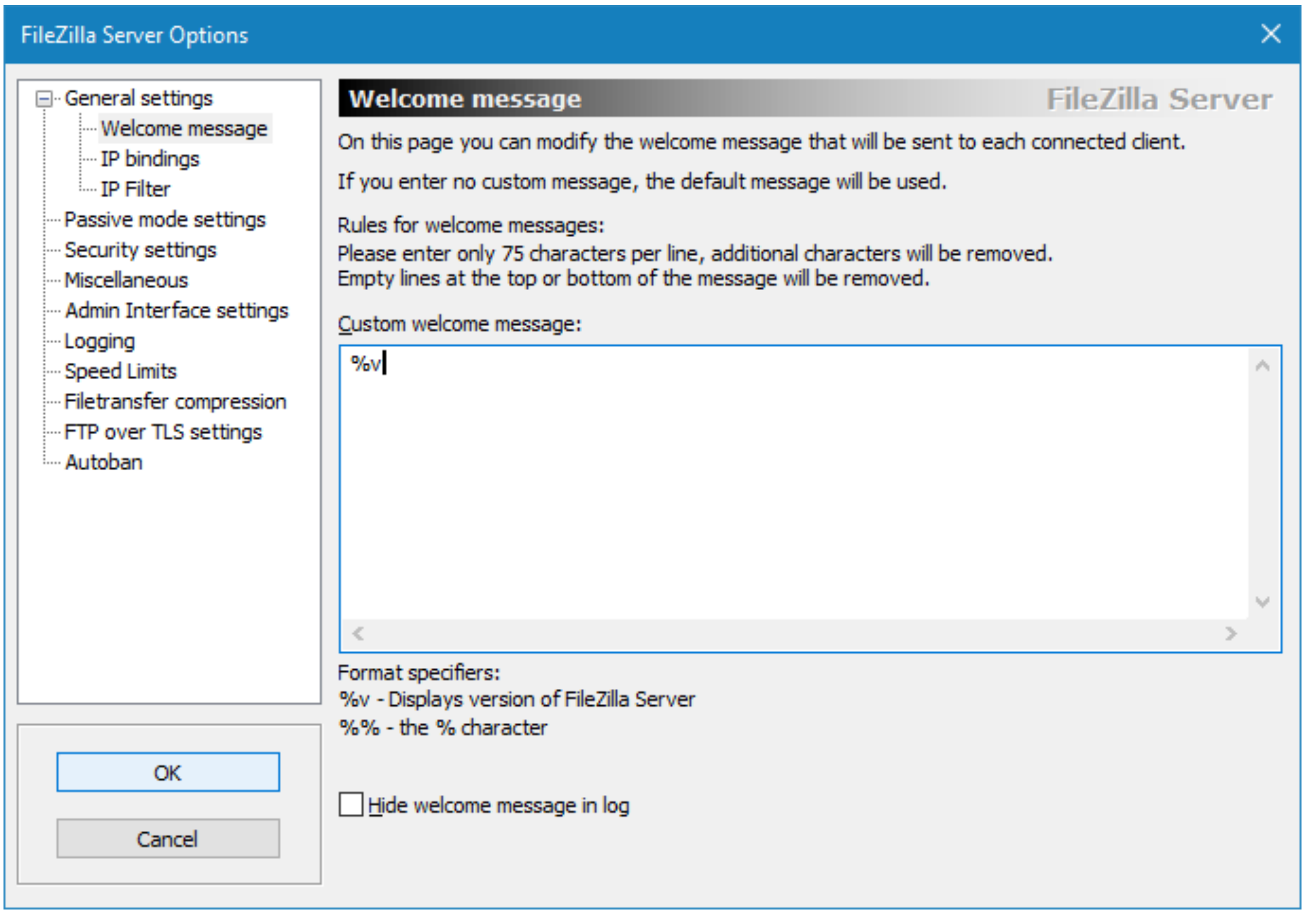
FileZilla Server (127.0.0.1)

```
FileZilla Server 0.9.60 beta
Copyright 2001-2016 by Tim Kosse (tim.kosse@filezilla-project.org)
https://filezilla-project.org/
Connecting to server localhost:14147...
Connected, waiting for authentication
Logged on
You appear to be behind a NAT router. Please configure the passive mode settings and forward a range
   of ports in your router.
Warning: FTP over TLS is not enabled, users cannot securely log in.
```

ID	Account	IP	Transfer


Ready

5. Click “Edit->Settings” and then “General Settings -> Welcome message” – leave only one line of welcome message so that the PLC has less work to do. Then click OK to accept the change.



On this page you can modify the welcome message that will be sent to each connected client.

If you enter no custom message, the default message will be used.

Rules for welcome messages:
Please enter only 75 characters per line, additional characters will be removed.
Empty lines at the top or bottom of the message will be removed.

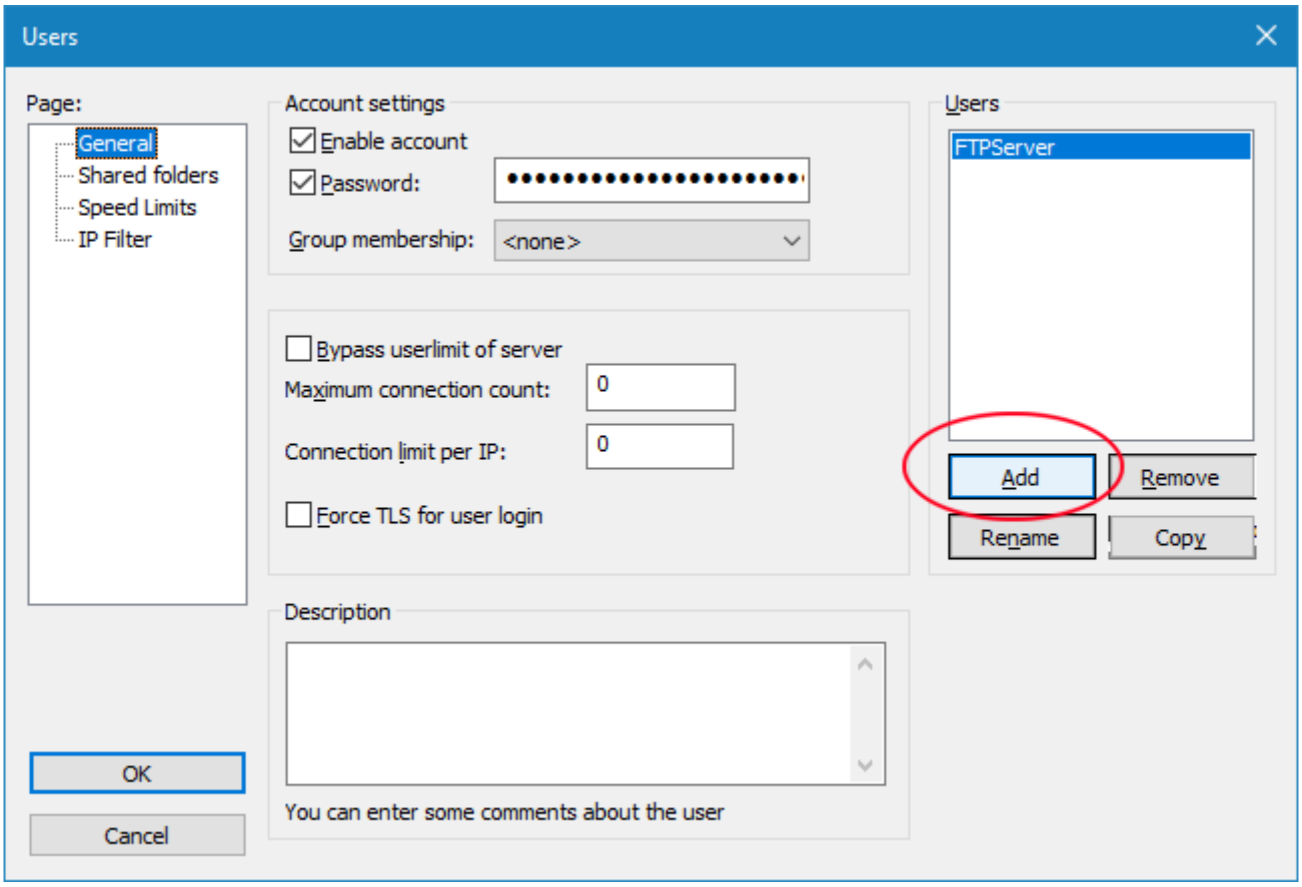
Custom welcome message:

```
%v
```

Format specifiers:
%v - Displays version of FileZilla Server
%% - the % character

☐ Hide welcome message in log

6. At the “General” page of the setup screen, click “Add” button at the “Users” pane to add a username “PLC”. Since no group has been defined simply leave the default as “<none>” in the second text box as shown in the following diagram.
7. At the “General” page of the Users setup screen, click “Add” button at the “Users” pane to add a username “PLC”. Since no group has been defined simply leave the default as “<none>” in the second text box as shown in the following figure.



☑ Enable account

☑ Password: ●●●●●●●●●●●●●●●●●●●●●

Group membership: <none>

☐ Bypass userlimit of server

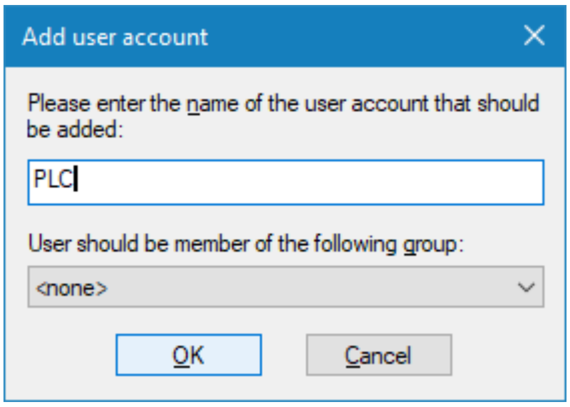
Maximum connection count: 0

Connection limit per IP: 0

☐ Force TLS for user login

Users: FTPServer

You can enter some comments about the user



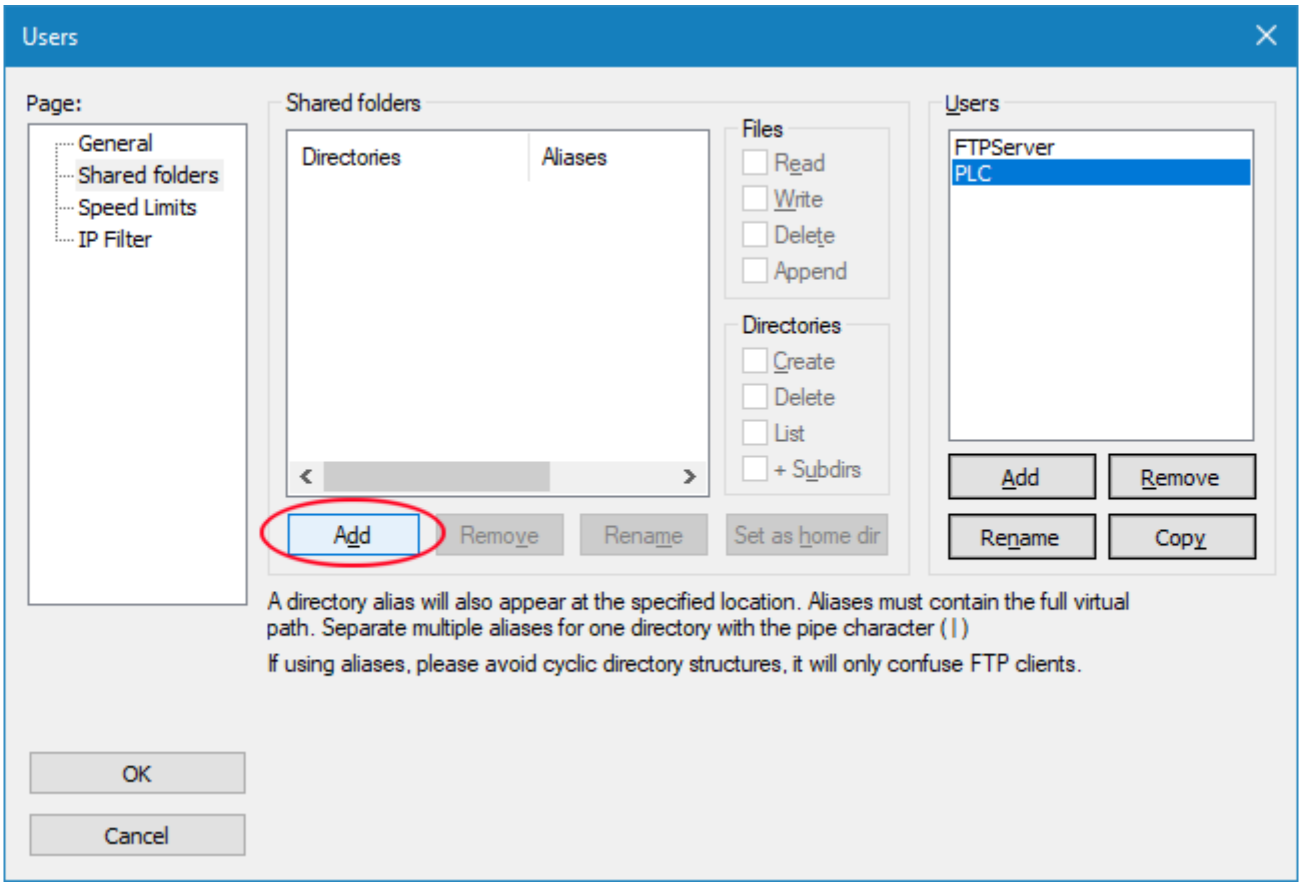
Please enter the name of the user account that should be added:

PLC

User should be member of the following group:

<none>

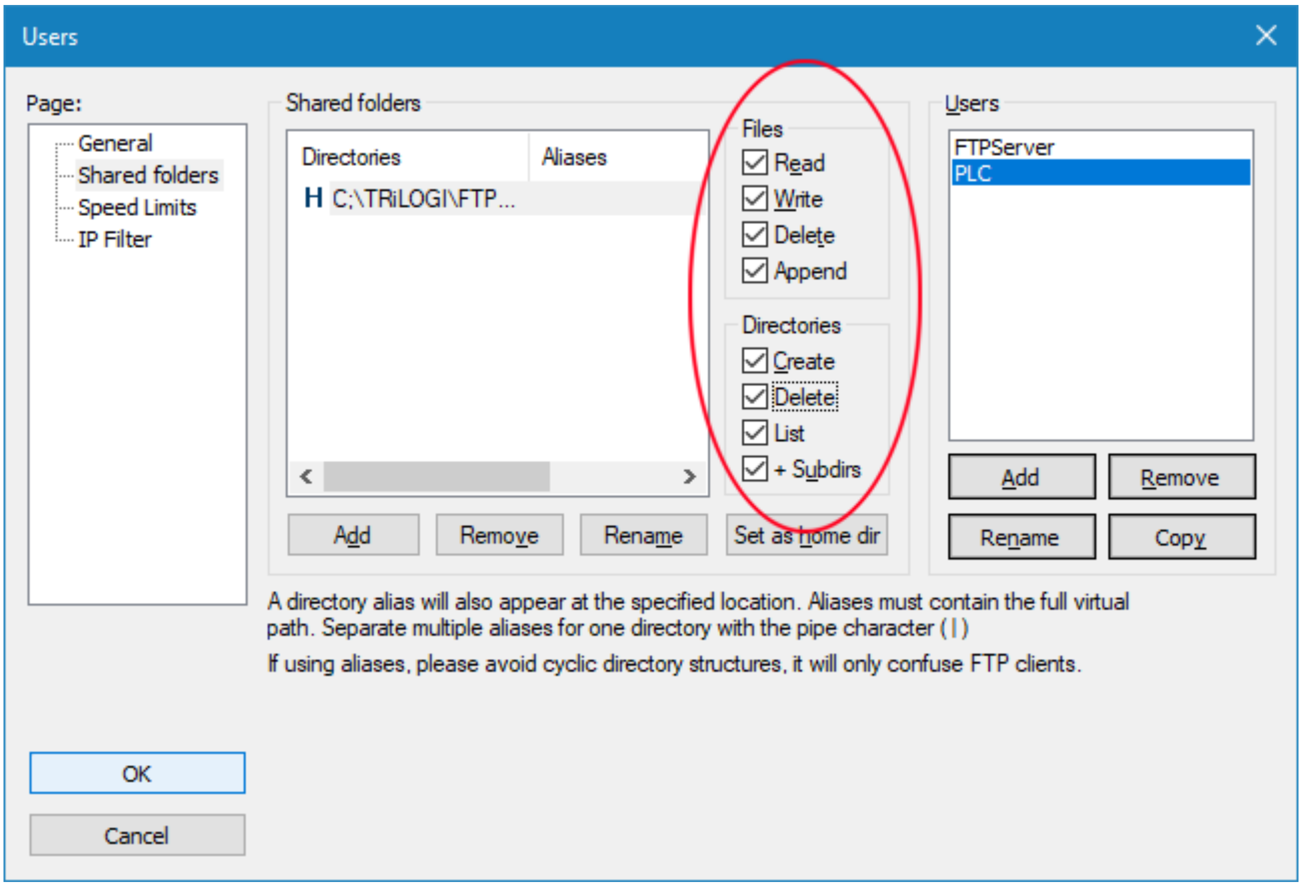
8. Next click “Shared folders” page and you must setup a folder that is to be used to receive uploaded file. Click “Add” at the “Directories” pane to add the folder.



A directory alias will also appear at the specified location. Aliases must contain the full virtual path. Separate multiple aliases for one directory with the pipe character (|)

If using aliases, please avoid cyclic directory structures, it will only confuse FTP clients.

9. You can choose any folder on your PC to be used for the FTP upload and you just have to remember the location so that you can look for the uploaded file later in the test. However, **make sure that you check all the check boxes** for “Read”, “Write”, “Append” etc as shown in the following figure:


Users

Page: General, Shared folders, Speed Limits, IP Filter

Shared folders

Directories	Aliases
H C:\TRiLOGI\FTP...	

Files: ☑ Read ☑ Write ☑ Delete ☑ Append

Directories: ☑ Create ☑ Delete ☑ List ☑ + Subdirs

Users: FTPServer, PLC

A directory alias will also appear at the specified location. Aliases must contain the full virtual path. Separate multiple aliases for one directory with the pipe character (|)

If using aliases, please avoid cyclic directory structures, it will only confuse FTP clients.

10. Click “OK” to complete the setup. The FTP server should now be ready and wait for incoming connection from any FTP client including the PLC.

17.6 Testing Filezilla FTP Server

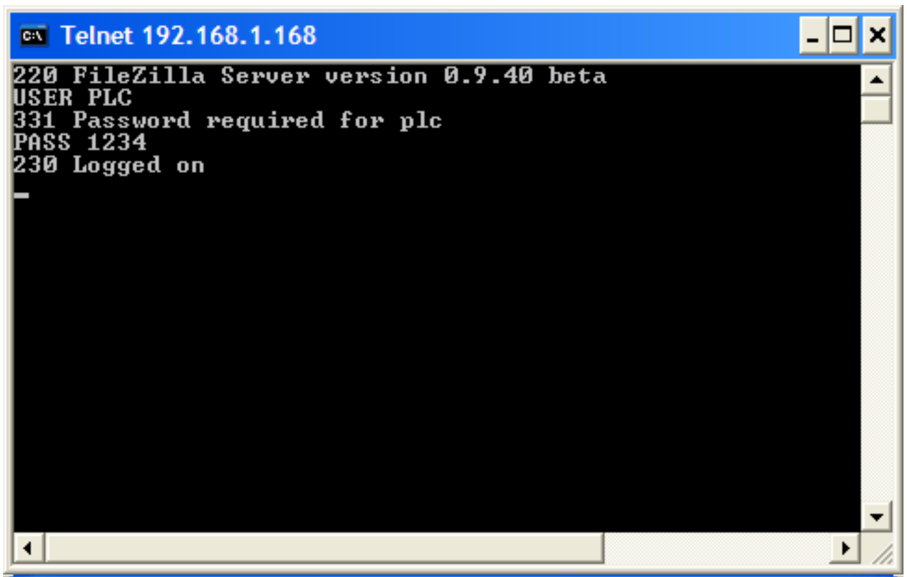
You can now test the FTP Server using the FileZilla client as mentioned in Chapter 2.9 (https://docs.triplc.com/wx100-um/#3059) in this Manual.

But a better way to test is to use the “Telnet” program on your PC (if you are running Windows Vista or Windows 7 you may need to enable the Telnet program since it is disabled by default – do a quick Google search on how to enable the Telnet client software on your PC).

Also you may want to find out the IP address of your PC that is running the FileZilla Server. If you have TLServer running on your PC your IP address is reported on the TLServer’s front panel. You can also get the IP address from the Windows “Network Connection Status” . Our test PC has an IP address = 192.168.1.168 which will be used in the following tests as well as used in the PLC program to connect to the FTP Server.

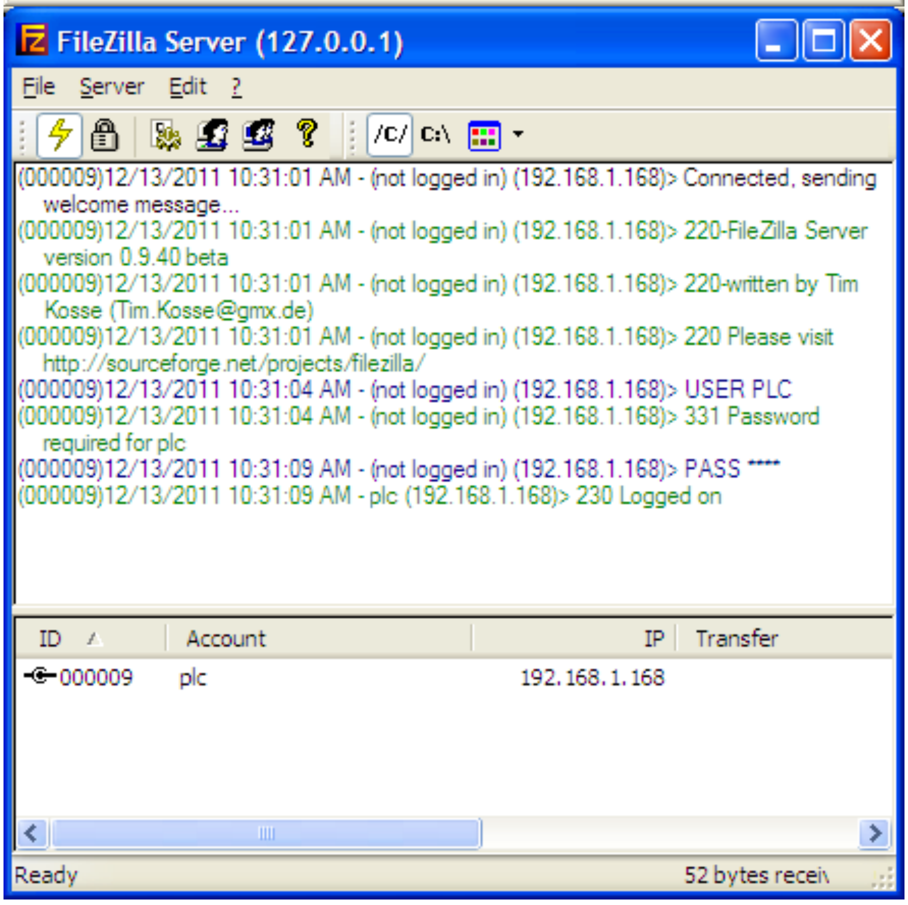
1. First open a command prompt window and then type “telnet 192.168.1.168 21” – this will open a telnet connection to the FTP server on our test PC with IP address 192.168.1.168 and listening at port 21. Please replace the IP address with the actual IP address of your The following screen shot captures the test sequence. Note that the same command/response

sequence with the server is also shown on the FileZilla Server Interface program front panel:



Telnet 192.168.1.168

```
220 FileZilla Server version 0.9.40 beta
USER PLC
331 Password required for plc
PASS 1234
230 Logged on
_
```



FileZilla Server (127.0.0.1)

```
(000009)12/13/2011 10:31:01 AM - (not logged in) (192.168.1.168)> Connected, sending
   welcome message...
(000009)12/13/2011 10:31:01 AM - (not logged in) (192.168.1.168)> 220-FileZilla Server
   version 0.9.40 beta
(000009)12/13/2011 10:31:01 AM - (not logged in) (192.168.1.168)> 220-written by Tim
   Kosse (Tim.Kosse@gmx.de)
(000009)12/13/2011 10:31:01 AM - (not logged in) (192.168.1.168)> 220 Please visit
   http://sourceforge.net/projects/filezilla/
(000009)12/13/2011 10:31:04 AM - (not logged in) (192.168.1.168)> USER PLC
(000009)12/13/2011 10:31:04 AM - (not logged in) (192.168.1.168)> 331 Password
   required for plc
(000009)12/13/2011 10:31:09 AM - (not logged in) (192.168.1.168)> PASS ****
(000009)12/13/2011 10:31:09 AM - plc (192.168.1.168)> 230 Logged on
```

ID	Account	IP	Transfer
000009	plc	192.168.1.168	

Ready 52 bytes recei

2. Once you get the “230 Logged On” message you know that the FTP setup is done correctly. Note that the welcome message from the FTP Server shows only one line “**220 FileZilla Server Version x.xx**” which is what we have set it up to be. You can now disconnect from the FTP server by typing “Quit” at the command prompt.

3. There is one more things you need to do before you proceed to test the FTP upload features of the PLC to avoid connection problem – that is to temporarily **TURN OFF the Windows Firewall**

and any software firewall setup by anti-virus software during your test. You can always re-enable your software firewall(s) after the test if you wish. PC operating system are designed to run client program normally instead of acting as a server so Windows Firewall by default is to block all incoming connections to the FTP Server. Thus it can give you a lot of headache when you are trying to connect to the FTP server operating behind the Windows Firewall.

Notes:

- The main purpose of Windows Firewall is to protect your PC when you are connected to say a public wi-fi network. But if your PC is connected to the Internet via a router at work or at home, the router hardware itself would act as a firewall to isolate your PCs and a software Firewall is actually redundant. (If a hacker tried to connect to a port to your public Internet IP address what he reached would be the port on the router and he would not be able to reach your PC, unless you have specifically set up to forward all TCP/IP messages sent to that port to a specific PC).

- If you really want to use the Filezilla FTP Server as a permanent server on a PC to receive files, and still want to have the Windows Firewall enabled, you can refer to the next section in this chapter which describes how to do it.

17.7 Windows Firewall For Filezilla

Most Windows PC today are shipped with Windows Firewall enabled, which will prevent inbound connection to the Filezilla server program from any external device. Hence in order for Filezilla server to work successfully you have to setup the Windows Firewall to allow inbound connections. For quick test you can temporarily disable the Windows Firewall if you do not wish to go through the process of setting up the inbound rules. If you want to setup Filezilla to work permanently to accept inbound FTP connection then you and follow the step by step instructions below:

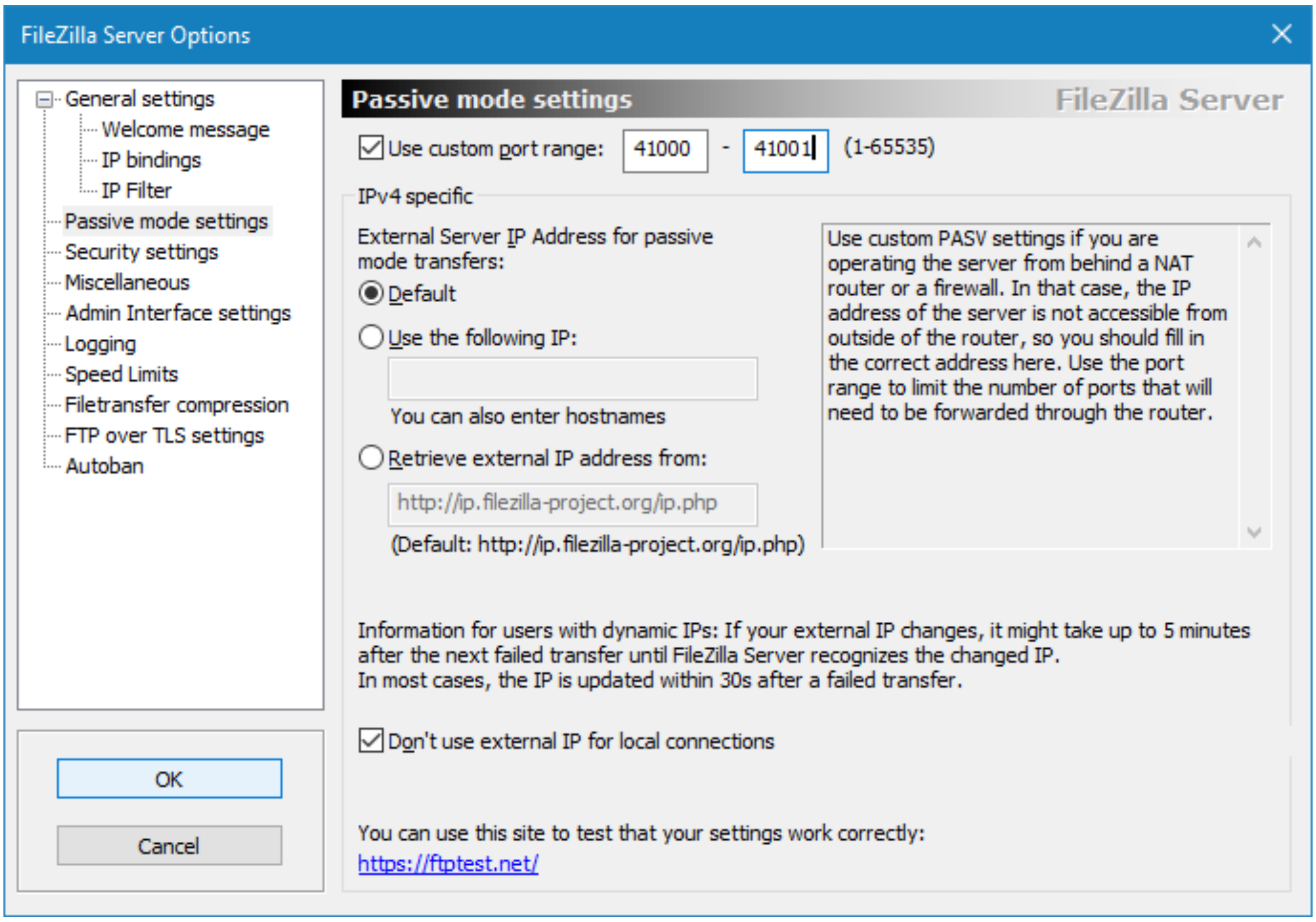
You can also refer to the following Microsoft document describing issues and solutions related to FTP server behind the Windows Firewall.

http://technet.microsoft.com/en-us/library/dd421710(WS.10).aspx (http://technet.microsoft.com/en-us/library/dd421710(WS.10).aspx)

Microsoft focuses mainly on the FTP server in their IIS server (for obvious reasons) instead of Filezilla. If you are setting Filezilla as a permanent FTP server behind a software firewall you can try to make the following configuration setup:

1) You must specifically setup a range of port number for Passive mode use. These are the port number that Filezilla will assign to the PLC to make a data channel connection when it attempts to transfer a file using passive mode. The following is an example where two port numbers are assigned so that two PLCs may connect to the Filezilla simultaneously. You can add a larger port

range if more PLCs may connect to the FTP Server simultaneously.


FileZilla Server Options

Passive mode settings

[✓] Use custom port range: 41000 - 41001 (1-65535)

IPv4 specific

External Server IP Address for passive mode transfers:

[●] Default

( ) Use the following IP:

You can also enter hostnames

( ) Retrieve external IP address from: http://ip.filezilla-project.org/ip.php

(Default: http://ip.filezilla-project.org/ip.php)

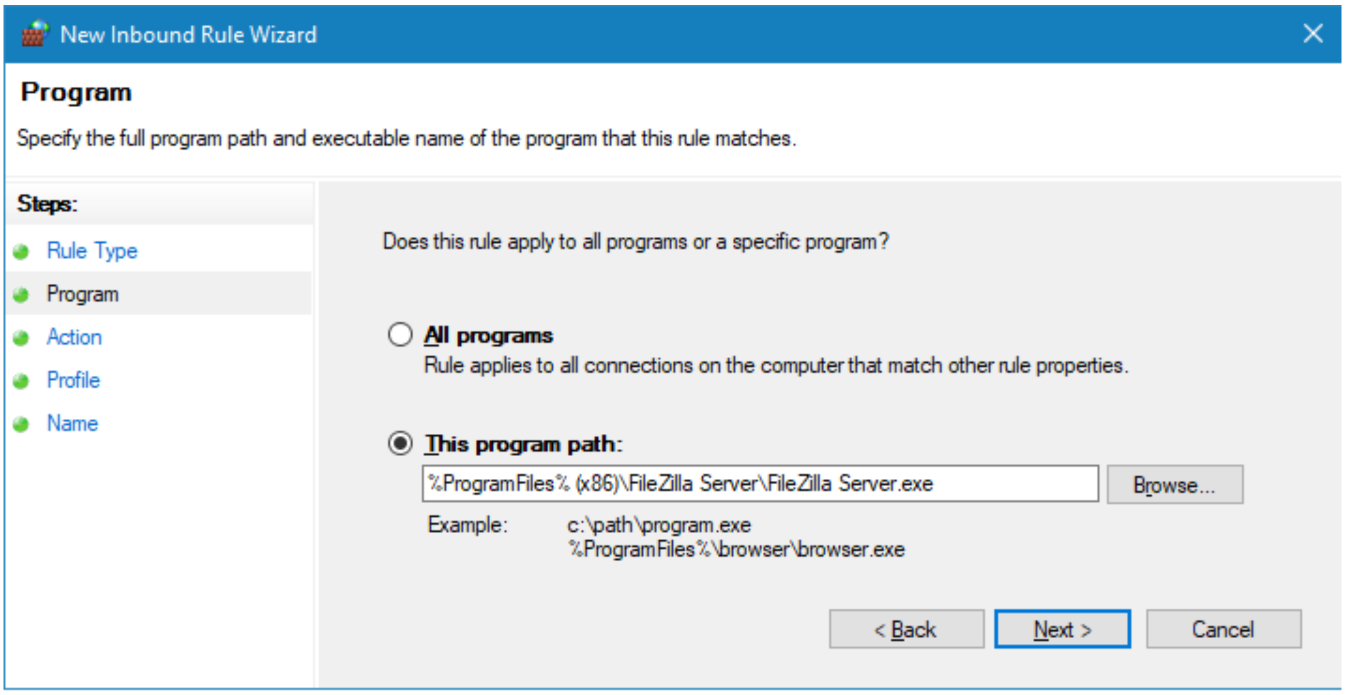
Use custom PASV settings if you are operating the server from behind a NAT router or a firewall. In that case, the IP address of the server is not accessible from outside of the router, so you should fill in the correct address here. Use the port range to limit the number of ports that will need to be forwarded through the router.

Information for users with dynamic IPs: If your external IP changes, it might take up to 5 minutes after the next failed transfer until FileZilla Server recognizes the changed IP.
In most cases, the IP is updated within 30s after a failed transfer.

[✓] Don't use external IP for local connections

You can use this site to test that your settings work correctly:
https://ftptest.net/

2) The next step is to allow FTP connections through the Windows Firewall. Open up Windows Defender Firewall and click on the “Advanced Settings” link. Then click on “Inbound Rules” and click “New Rule”. Click on “Program” and browse to the Filezilla FTP server executable.


New Inbound Rule Wizard

Program

Specify the full program path and executable name of the program that this rule matches.

Steps:
- Rule Type
- Program
- Action
- Profile
- Name

Does this rule apply to all programs or a specific program?

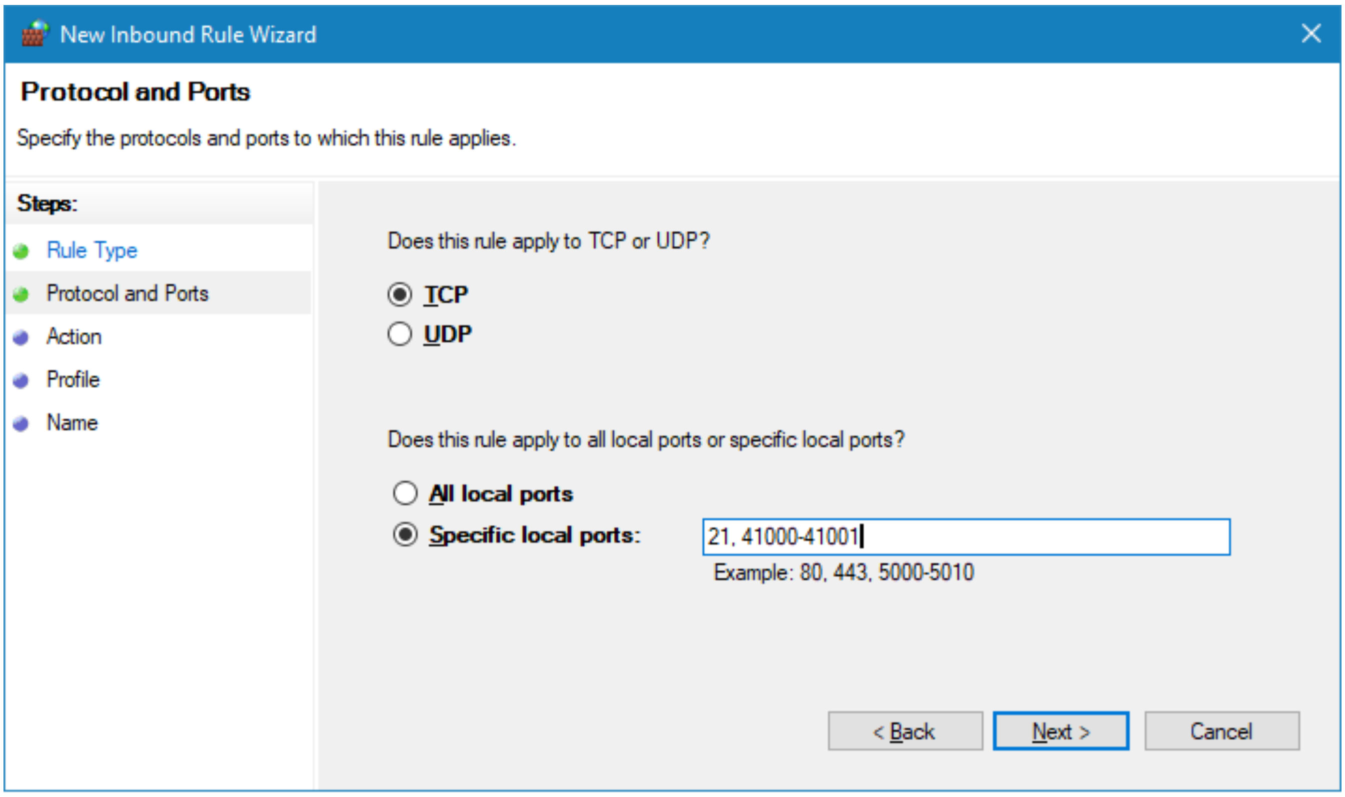
( ) **All programs**
Rule applies to all connections on the computer that match other rule properties.

(●) **This program path:**
%ProgramFiles% (x86)\FileZilla Server\FileZilla Server.exe

Example: c:\path\program.exe
%ProgramFiles%\browser\browser.exe

Click Next and make sure that “Allow Connection” option is selected and click next again and give a name to your new rule. Finally, click “Finish” button when you are done setting up the program exception rule.

3) Next we will add the FTP ports to allow inbound connection. Click on “New Rule”, select “Port” and select “TCP”, then either select “All local ports” (the easiest way, but less secure) or “Specific local ports” and then enter the port number 21 (FTP server port) and the port range 41000-41001 that we want to allow Filezilla Server to accept inbound connection, as shown below:


New Inbound Rule Wizard

Protocol and Ports

Specify the protocols and ports to which this rule applies.

Steps:
- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

Does this rule apply to TCP or UDP?

(●) TCP
( ) UDP

Does this rule apply to all local ports or specific local ports?

( ) All local ports
(●) Specific local ports: 21, 41000-41001

Example: 80, 443, 5000-5010

Click “Next” and make sure that “Allow Connection” option is selected and click next again and give a name to your new rule. Finally, click “Finish” button when you are done setting up the port exception rule.

17.8 PLC FTP Upload to FileZilla

a) Overview of The FTP Protocol

The FTP protocol requires **two** socket-connections between the devices performing the file transfer. One connection is the “command” channel where FTP commands such as STOR or DELE and the responses are sent as plain ASCII text strings between the FTP client and the FTP server. The second connection is the “data” channel where only the file content or the file directory data are being transferred.

There are two transfer modes: “Active” mode and “Passive” mode. Active mode requires that the server establish a data connection back to the client. Passive mode on the other hand, requires that the client also be the one to establish the data connection. i.e. For passive mode, both the command and the data connections are performed by the client (the PLC in this case).

The PLC has been designed to use only **passive** mode to transfer file to the FTP server. Passive mode is preferred because that is the only way to transfer file if the FTP Server is located on the Internet. The alternative active mode transfer requires the server to make a data connection back to the PLC that is sitting behind a router firewall, and that can be problematic unless the router is specifically configured to forward the data port to the PLC.

b) PLC FTP Upload Procedure

In order to upload file to the FTP Server, the PLC would use the PRINT #4 "<TCPConnect xxx.xxx.xxx.xxx:21>" command tag to connect to the FTP server port 21 to establish the "command" connection to the FTP Server. The PLC uses its PRINT #4 to send and INPUT$(4) to receive ASCII text strings from the FTP server via the command channel. The PLC would then send the command "PASV" to inform the FTP server that it wants to transfer a file in passive mode.

At this point you can use the PRINT #4 command to send any valid FTP command to the server, including changing directory (CWD command – make sure the directory exist), deleting a file (DELE command – beware of what you are deleting!) etc.

When the PLC is ready to start a file transfer to the FTP server, the server will in turn provide the PLC with the port number that it has opened for the PLC to connect to establish a data connection. Upon receiving this port number the PLC will make a second TCP connection to the given port and then the actual file transfer will begin.

A new, network service command tag named "FTPUPLD" handles the negotiation between the FTP Server and the PLC as well as handling of the data transfer from the PLC to the FTP server. The following is the syntax:

```
    PRINT #4 "<FTPUPLD Zxxx.yyy  [destination file name]>"
```

Where: Zxxx.yyy is the file name of the extended file that the PLC has access to. The [destination file name] can be any legal name acceptable to the server so you can attach a date or time stamp to the file name for easy identifications.

When the above command is run, the PLC will send the actual "STOR" command in the background to the FTP server and then obtain the port number from the server and it will then make a data connection to it, and file transfer can then begin.

c) Monitoring The FTP Upload Progress

Once the file transfer begins the PLC firmware will handle the rest of the file transfer until either the file has been completely transferred or the transfer is aborted due to a network or server trouble. You can monitor the progress of the file transfer using either the **STATUS(4)** or **STATUS(20)** functions.

```
STATUS(4) = 0  : FTP client was idle or last FTP failed
            1  : FTP data transfer just started
            2  : 1st FTP segment transferred, now transferring the rest
            3  : FTP data transfer completed.

STATUS(20) > 0 : Number of bytes uploaded to FTP Server. Transfer is in progres
s.
           < 0 : Total number of bytes uploaded. Transfer completed.
```

For example, If 2,345 bytes has been uploaded to the server and the transfer has ended, **STATUS(20)** will return the number = –2345.

Since file transfer can take substantial amount of time to complete, it is not wise to run a loop to wait for the file transfer to complete since this will block the PLC from processing any other part of the program. The demo “ExtendedFileSystem.PC6” shows you how to setup a monitoring function to periodically monitor the progress of the file transfer and report the transfer status on the LCD display.

Please refer to the comments in the custom function “fnConnFTP” and “fnMonFTP” of the “ExtendedFileSystem.PC6” program for more detailed descriptions of each command involved in the FTP file transfer.

Chapter 18 - ModbusTCP Gateway

18.1 Introduction

A Wx100, Fx or a Nano-10, FMD PLC with >= r77 firmware can be configured to act as the “MODBUS/TCP GATEWAY” for other Modbus serial devices while continue to function as a Super PLC!

A Modbus/TCP gateway essentially translates a Modbus/TCP command packet it receives from its Ethernet port or WiFi connection into a Modbus RTU serial command and sends it out of its serial port (RS232 or RS485). If the Modbus RTU slave sends back a serial response, the gateway will in turn translate the RTU response data back to Modbus/TCP response packet and return to the client via the Ethernet port.

As such the Modbus/TCP gateway enables any non-Ethernet and non-WiFi equipped, both TRi or 3rd party serial Modbus slave device to be directly accessible by a Modbus/TCP client via the Ethernet or the Internet. In addition, the user PLC program does not need to handle the gateway function at all as the CPU performs the translation functions automatically and transparently to the PLC’s program.

Best of all, while acting as a gateway, the Nano-10, FMD or F-series PLC program can **simultaneously act as a Modbus master PLC** and read/write to any registers inside any of the attached Modbus RTU slave! The CPU firmware automatically schedules the order of the command/response packets whether it is originated from the client or from the PLC itself so that the Modbus/TCP command from the client will be responded to in the correct order and in a timely manner.

18.2 Application Ideas for Modbus/TCP Gateway

The Modbus/TCP gateway function can be very useful for many large area control systems, such as a building automation system. A master Wx100 or a F-series PLC is linked to many Modbus RTU slave devices via RS485 bus distributed across an entire building. The master PLC can perform sophisticated control functions since it has read/write access to **ALL** the Modbus RTU slaves it connects to. At the same time, a Modbus TCP client software such as a Building Management System (BMS) can access any registers in the master PLC or **ANY** of the slave Modbus RTU controller directly via the master PLC acting as a gateway for the slave PLCs.

18.3 Configuring The PLC As Modbus/TCP Gateway

The command used to define a serial port to become Modbus/TCP Gateway serial port is as follow:

```
SETSYSTEM 12, n
```

Where n = 1, 2 or 3 for COMM1, COMM2 and COMM3 port.

For Wx100 there is only one RS485 serial port which is assigned as COMM1.

This statement must be run once (could be during start up via 1st.Scan pulse) to configure the serial port #n to be used to send out RTU commands and received RTU responses. If SETSYSTEM 12,n is not run then the Modbus gateway function will be disabled.

Here is how it works: A Modbus/TCP client (such as a SCADA, HMI etc) would send a Modbus/TCP request with a specific 8-bit Modbus slave ID to the gateway PLC. If the ID **matches** the native ID that is assigned to the gateway PLC then the PLC will react normally by sending its own Modbus/TCP response packet back to the client. The PLC will not send any RTU command out of the gateway serial port. This means that the client can access the master PLC's own register as per normal.

However, if the client send a Modbus/TCP packet with a different ID from that of the gateway PLC's ID, then the gateway PLC will translate the command into a Modbus RTU command and send it out of the gateway serial port #n defined by the “SETSYSTEM 12, n” statement. The PLC will also wait for a response from the RTU slave via the gateway serial port, and if it does receive a response it will translate it into Modbus/TCP response packet and return to the client.

Note:

- If the Modbus/RTU slave being addressed does not respond to the RTU command, then the gateway PLC will wait till time-out (default is about 150ms) and then resend the Modbus RTU command and wait for a response again. By default the gateway PLC will repeat this procedure twice and if it still doesn’t receive a response after repeated attempts, it will send back a “TARGET DEVICE FAILED TO RESPOND” error packet back to the Modbus/TCP client. (i.e. It will set the 7th bit of the function code to “1” and send back an exception code “0B” hex).
- The CPU firmware is designed to handle only one Modbus gateway translation for each Modbus/TCP connection per ladder logic scan. This is to ensure that the CPU is not completely bogged down by its Modbus gateway job and will have time to run its own program. If the Modbus/TCP clients attempt to overload the PLC with more requests than it can handle the PLC will frequently return “SLAVE DEVICE BUSY” error response packets back to the client. (i.e. It will set the 7th bit of the function code to “1” and send back an exception code “06”).

18.4 Fine-Tuning The Modbus/TCP Gateway Function

It is important to know that every time the PLC runs a Modbus/TCP to Modbus RTU translation its own program scan time increases due to the need for the CPU to wait for a response from the slave Modbus RTU devices. The delay becomes much more pronounced when a Modbus slave device fails to respond (e.g. not online). The system designer must therefore take this into consideration when designing the system:

1) The Modbus/TCP client should not overload the Modbus gateway with unnecessarily frequent Modbus requests. Whenever the PLC report a “TARGET DEVICED FAILED TO RESPOND” function the client program should back off for a while before re-trying communication with the Modbus/RTU slaves again.

2) You can change the number of wait states the gateway will wait for a response from the Modbus/RTU slave by running:

```
SETSYSTEM 1, w
```

Where w = number of serial port wait states for each COMM port as follow:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Ethernet		COMM3		COMM2		COMM1	

Thus each COMM port could be configured to wait from 0 to 3 wait states for a response. The default settings for w = &H55 (or 01010101 binary) which means each COMM port as well as the Ethernet port uses 1 wait state of about 150ms.

3) You can change the number of retries (default = 2) the gateway will attempt to get a response from the Modbus RTU slave by running:

```
    SETSYSTEM 2, n
```

Where n = number of retries for each COMM port as follow:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Ethernet		COMM3		COMM2		COMM1	

Thus each COMM port could be configured to re-attempt to send the command from 0 to 3 times if it does not get a response from a Modbus RTU slave.

The default settings for n = &HAA (or 10101010 binary) which means each COMM port as well as the Ethernet port will retry twice (making a total of 3 attempts). E.g. If you run SETSYSTEM 2, 0 the gateway will not resend the RTU command on any of its COMM port if does not receive a response on the first attempt. This can help to reduce the CPU scan time if a Modbus RTC slave does not respond.

18.5 Modbus/TCP Gateway Sample Program

Please download the following self-explanatory I-TRiLOGI sample program that demonstrates the new Modbus/TCP Gateway capability via any of the selected COMM port:

```
    http://www.tri-plc.com/trilogi/modbusgateway.zip (http://www.tri-plc.com/trilogi/m
odbusgateway.zip)
```

Chapter 19 - PLC Self-Update

Current users of TRi Super PLC may be aware that the PLC programs can be updated using one of the following 2 ways only:

1. Program directly using the i-TRiLOGI software. This is normally performed by the PLC programmer

2. Transfer a file with ".CO5" extension to the PLC using the "CO5_Uploader" program. The PLC programmer will use the i-TRiLOGI software to generate the CO5 file, which is the compiled version of the program. The programmer can then email the CO5 file to the end user to update their PLC program. Since the CO5 file does not contain any trace of the original source file, the IP of the programmer is protected against meddling.

We have now added a new way of updating the PLC program exclusive to the Wx100 PLC only, which is to allow the Wx100 PLC to automatically download a CO5 file from a password-protected web server and update its own program. This can be a very powerful tool for OEMs to minimize the support issue when updating hundreds or thousands of their equipment that have already been deployed in the field. The process is actually really simple, as described below:

1) After having thoroughly tested the new program for the equipment to perform self-update, the programmer shall use i-TRiLOGI to generates a CO5 file.

2) The CO5 file is uploaded to a web server (HTTP only, HTTPS is currently NOT supported). A special folder can be created to host the CO5 file and the folder can optionally be password-protected so that the CO5 file can only be downloaded after the client has been authenticated.

3) The programmer can hard-code into the PLC program the URL of the web server that is hosting the CO5 file, the (optional) username and password required to access the CO5 file. The PLC can be programmed to periodically check for a new version of the CO5 file and self-update when the new file is found. Alternatively, the programmer can add an option to the Wx100 HMI menu system so that the end user can navigate the menu and manually initiate a program self update only as and when needed. The following single command is all it takes to begin a CO5 download and self-update:

```
PRINT #4 "<DOWNLOAD_CO5 file_path>"
```

The PLC program uses this new network services command to initiate the self-update after making a TCP connection to the web server that is hosting the CO5 file. If provided, the <DOWNLOAD_CO5> command will automatically supply the username and password credentials to the webserver when it is queried by the web server and then begin downloading the CO5 file immediately.

The following is a full example to illustrate the process:

```
#DEFINELOCAL  download_username = "my_username"
#DEFINELOCAL  download_password = "my_password"
#DEFINELOCAL  download_URL = "triplc.com:80"
#DEFINELOCAL  download_path = "/~triplc/test/WxDemo.CO5"

SCREEN_CLEAR
C = 0
IF STATUS(3)
    PRINT #4 "</>"
ENDIF

' Upgrade from CO5 file stored on triplc.com (directory for the domain name translate t
o /~triplc)
' file path is:  /~triplc/test/WxDemo.CO5

PRINT #4 "<USERNAME ";download_username;">"
PRINT #4 "<PASSWORD ";download_password;">"

PRINT #4 "<TCPCONNECT ";download_URL; ">"
DELAY 200

IF STATUS(3) = 0
   SETLCD 1,1, "No connection"
   RETURN
ELSE
   SETLCD 1,1, "Upgrading PLC Prog."
ENDIF

PRINT #4 "<DOWNLOAD_CO5 ";download_path;">"
PRINT #4 "</>"
```

The Wx100 PLC’s operating system automatically computes the checksum of the downloaded content against the CO5 file headers to confirm if the downloaded content are valid before committing the program to flash memory. After the download is completed and the program has been successfully updated, the Wx100 PLC will automatically reboot and start running the new program. If the download fails for whatever reasons the PLC will display an error message on the OLED screen for a few seconds and then reboot. When that happens the PLC will be back to running the previously successfully updated program and the recently failed downloads should not affect the PLC.

The PLC automatically logs the time-stamped success or failure message into a local file named “Download_CO5.log” which can be retrieved any time later to confirm if the previous download has been successful.

Alternative command:

```
    PRINT #4 "<DOWNLOAD_CO5_PAUSE file_path>"
```

If you run the alternative command as shown above in place of the <DOWNLOAD_CO5 xxxx> command described earlier, then the PLC will PAUSE after the program download regardless of whether the self-update is successful. The user will need to manually reboot the PLC by power-cycling it in order to run the new program.

About TRi PLCs

For more than 27 years, TRi PLCs have been the leading choice of embedded PLCs for Original Equipment Manufacturers .

Support Links

FAQ (Https://triplc.com/faq.htm)

Forums (Https://triplc.com/smf/)

Documentation

Information

Products Page (Https://triplc.com/)

Testimonials (Https://triplc.com/testimonials.htm)

Resources (Https://triplc.com/resources.htm)