MET-TP 1st Course

A Guide to TOSHIBA PLC

"If a man doesn't plant a tree for himself, he plants it for the next generation. Learn and give knowledge."

ALMAWARED ENGINEERING & TRADING SAE

1st edition 11/2011

MET-TP 1st Course

A Guide to TOSHIBA PLC

"If a man doesn't plant a tree for himself, he plants it for the next generation. Learn and give knowledge."

ALMAWARED
ENGINEERING
&TRADING SAE

Table of Contents

1. Introduction

Introduction

Welcome to the first course in the **MET-TP** series. Al-**M**awared **E**ngineering and **T**rading **T**raining **P**rogram offers a training service so that you can plan over the time the growth of the device knowledge, from the frequency inverters to the soft starters for asynchronous motors up to the PLC "Programmable Logic Controller"; Touch screens and SCADA system. Courses are targeted to engineers, technicians, users and installers and to the service personal as well. MET suggests courses that are mainly oriented to the use of the drives and of the automation system.

Our highly trained engineers will guide you step by step through each training course, allowing you to perform each step by yourself from small examples to large applications to help you practice everything you learn during the training course.

This course covers the **Basics of TOSHIBA PLC.** Upon completion of this course you will be able to:

- Connect any hardware components to TOSHIBA PLC
- Choose the best configuration for TOSHIBA PLC modules (analog, digital or communication) based on the number and type of points needed.
- Open existing projects in TOSHIBA programming tool (T-PDS)
- Develop new software or modify an old one.
- Download and upload any software.
- Implement all the basic functions needed in software to perform a certain sequence.
- Know all the essential data needed to put you on the first step of learning HMI software like Hakko Touch Screen (V-SFT) or Indusoft SCADA.

After you have completed this course, if you wish to determine how well you have retained the information covered, you can complete all the exercises described later in this course.

2. Company Overview

Experience and high responsiveness

Year 2001 is not only a date for ALMAWARED ENGINEERING AND TRADING S.A.E (MET). It represents the starting of a company that is today specialized in the industrial automation, mechanical and electrical power transmission field, thanks to the entrepreneurial capabilities from foundation members Mr. Abdel Aziz Aboul Atta, Ms. Bahia Khairy, Eng. Mohamed Abdel Aziz, Eng. Khalid Abdel Aziz and Eng. Khalid Ateya, expertise and firm commitment of the promoting partners.

MET partners achieved the quality certificates, with the aim to grant a good quality system for the different market needs to satisfy most of their customers' needs by providing complete solutions or individual tailor-made solutions. Only a long experience allows a company to reach in a flexible and wide way to market demands, with a complete range of products and services.

Flexibility and rapidity of product range

Our structure is composed of real problem solvers who study the customer requirements and orient him towards simple and innovative solutions thanks to the structure of MET product range that gives our highly professional technical engineers a wide area of solutions. MET product range comprises a whole variety of automation, electrical and mechanical equipment such as PLCs, touch screens, SCADA systems, flow meters, density meters, BMS components, inverters, soft starters, AC/DC motors, Servo systems, gearboxes, clutches and brakes. Distributed Control System (DCS) is the latest product that MET provides to our customers.

A strong presence in the market

Our experienced sales and marketing team is highly technical and customer-oriented. They combine premium customer support with years of industry experience to develop and sustain long-term relationships.

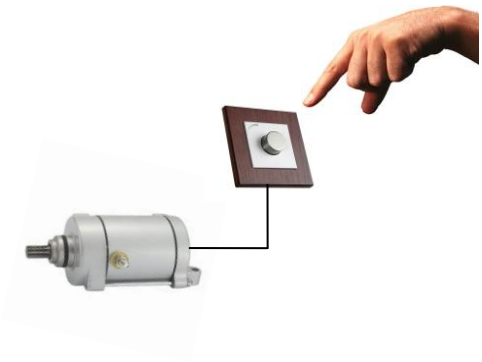
Our team ensures each customer is treated with professionalism and integrity as we work hard to understand our customers' needs and to provide solutions to meet those needs, so we serve a wide variety of market fields such as soup and fats, sugar industries, petroleum and refineries, cement factories, iron and steel mills, pharmaceutics and cosmetics, textiles and dyeing, paints and chemicals, plastics and petrochemicals, food & beverages industries, paper and printing, packing and wrapping, pump & water treatment plants, sewage & water treatment plants and commercial HVAC.

3. Control Overview

Control circuits

Control, as it relates to machines and processes, is a broad term that means anything from a simple toggle switch to a complex system.

Control is considered to be **manually operated** when someone must initiate an action for the circuit to operate. For example, someone might have to flip the switch of a manual starter to start and stop a motor.



Although manual operation of machines is still common, many machines are controlled automatically. Frequently there is a combination of manual and automatic control. For example, a machine that is started manually may stop automatically when certain conditions are met.

Control Components

While many control components are used in circuits that involve motors, control components are also used with a variety of other equipment. Various types of control components are used for switching, starting, protecting, detecting, monitoring, communicating, and other functions. The full range of these capabilities is beyond the scope of this course, but examples of devices that perform these types of functions are discussed throughout this course.

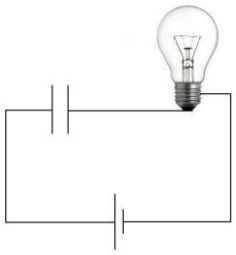
In some cases, the interaction of these components is dependent only on how they are wired to each other. This is sometimes referred to as **hard-wired logic**. Increasingly, however, these components are wired or networked to a control system, such as a programmable logic controller or variable frequency drive.
In such cases, the interaction of the circuit components is dependent both on wiring and the software stored in the controller.

Before discussing specific control devices; however, it is important to understand some basic symbols and diagrams. The symbols and diagrams described in this course are commonly used in North America. Other types of symbols and diagrams are also used.

Contact Symbols:

Various devices incorporate contacts to control the flow of current to other control components. When in operation, a contact may be either **open**, a condition which blocks current flow, or closed, a condition which allows current flow. Control logic diagrams, however, cannot show the dynamic operation of contacts. Instead, these diagrams show contacts as either **normally open (NO)** or **normally closed (NC)**.

The standard method of showing contacts is to indicate the circuit condition produced when the actuating device is in the **de-energized (off) state**. For example, in the following illustration, the contacts are part of a relay. The contacts are shown as normally open to indicate that, when there is no power applied to the relay's coil, the contacts are open. With the contacts open, there is no current flow to the light.

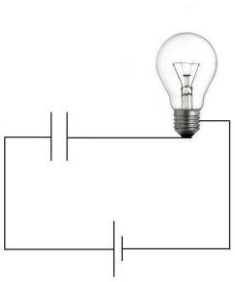


Relay De-energized

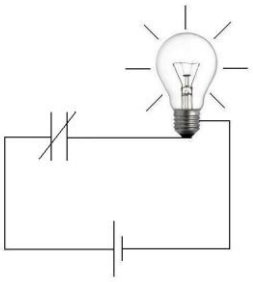
Symbols on a control logic diagram are usually not shown in their energized (on) state. However, in this course, contacts and switches are sometimes shown in their energized state for explanation purposes.

Normally Open Contact Example:

For example, in the following illustration, the circuit is first shown in the de-energized state, and the contact is normally open. When the relay energizes, the contacts close, completing the path for current and illuminating the light.
Note: This is not a standard symbol.



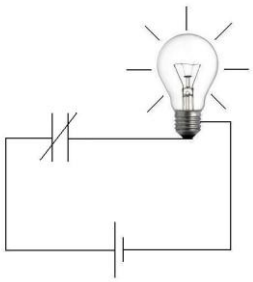
Relay De-energized



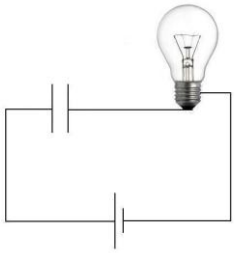
Relay Energized

Normally Closed Contact Example:

In the following illustration, when the relay is de-energized, the normally closed contacts are shown as closed. A complete path of current exists at this time, and the light is on. When the relay is energized, the contacts open, turning the light off.



Relay De-energized

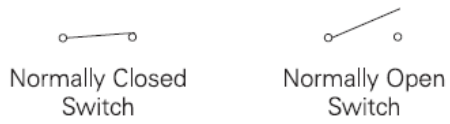


Relay Energized

Switch Symbols:

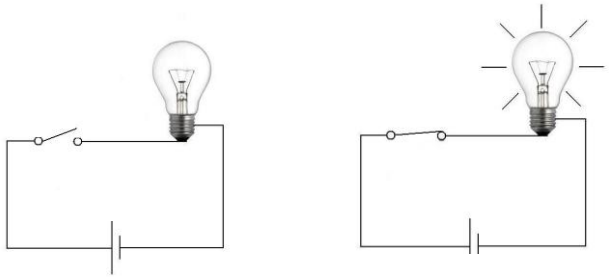
Various types of **switches** are also used in control circuits. Like the contacts just discussed, switches can also be normally open or normally closed and require another device or action to change their state. In the case of a manual switch, someone must change the position of the switch. A switch is considered to be in its normal state when it has not been acted upon.

Switch symbols, like the ones shown in the following illustration, are also used to indicate an open or closed path of current flow. Variations of these symbols are used to represent a number of different switch types.



Normally Open Switch Example:

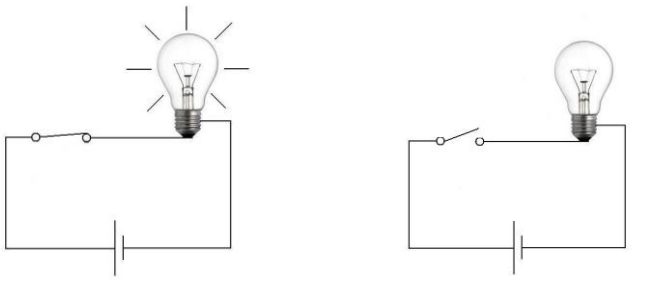
In the following illustration, a battery is connected to one side of a normally open switch, and a light is connected to the other side. When the switch is open, current cannot flow through the light. When someone closes the switch, it completes the path for current flow, and the light illuminates.



Switch is shown opposite its normal state

Normally Closed Switch Example:

In the following illustration, a battery is connected to one side of a normally closed switch and a light is connected to the other side. When the switch is closed, current flows through the light. When someone opens the switch, current flow is interrupted, and the light turns off.



Switch is shown opposite its normal state

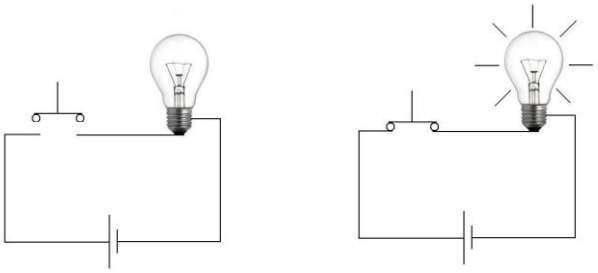
Pushbutton Symbols:

There are two general types of **pushbuttons**, **momentary** and **maintained**. The contacts of a momentary pushbutton change state open to closed or vice versa, when the pushbutton is pressed. They return to their normal state as soon as the button is released. In contrast, a maintained pushbutton latches in place when pressed. It must be unlatched to allow it to return to its normal state.

NO NC

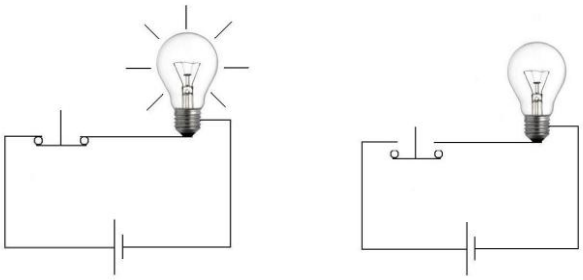
Normally Open Pushbutton Example:

In the following illustration, a battery is connected to one side of a normally open pushbutton, and a light is connected to the other side. When the pushbutton is pressed, current flows through the pushbutton, and the light turns on.



Normally Closed Pushbutton Example:

In the following example, current flows to the light as long as the pushbutton is not pressed. When the pushbutton is pressed, current flow is interrupted, and the light turns off.

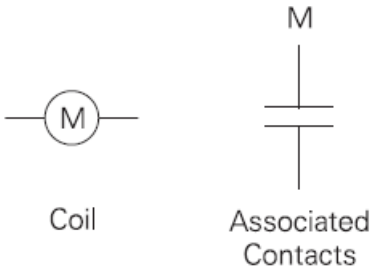


Coil Symbols:

Motor starters, contactors and relays are examples of devices that open and close contacts electromagnetically. The electromagnet in these devices is called a **coil.**

A coil is commonly symbolized as a circle with one or more letters and possibly a number inside. The letters often represent the type of device, such as M for motor starter or CR for control relay. A number is often added to the letter to differentiate one device from another. The contacts controlled by a coil are labeled with the same letter (and number) as the coil so that it is easy to tell which contacts are controlled by each coil.

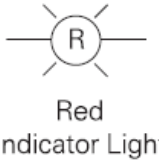
A coil often controls multiple contacts and a combination of normally open and normally closed contacts may be used.



Indicator Light Symbol:

An **indicator light**, often referred to as a **pilot light**, is a small electric light used to indicate a specific condition of a circuit. For example, a red light might be used to indicate that a motor is running.

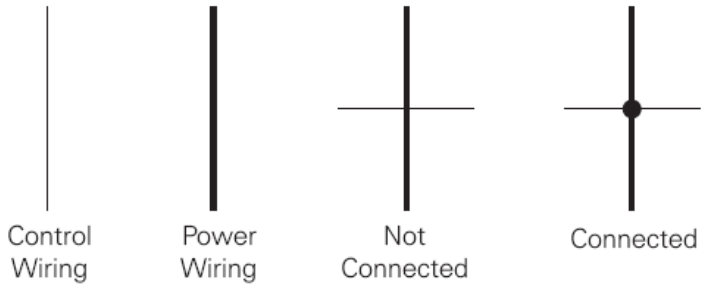
A letter in the center of the indicator light symbol is sometimes used to indicate the color of the light.



Line Diagrams

Control symbols are used in **line diagrams** also referred to as **ladder diagrams**. Line diagrams are made up of two types of circuits, control circuits and power circuits.

Within a line diagram, control circuit wiring is represented by a light line, and power circuit wiring is represented by a heavy line. A small dot or node at the intersection of two or more wires indicates an electrical connection.

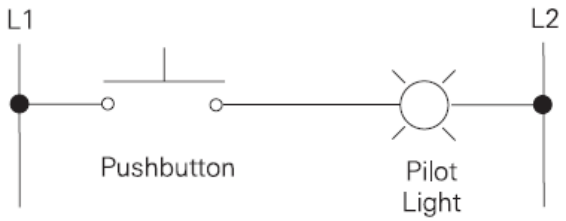


Line diagrams show the functional relationship of components in an electrical circuit, not the physical relationship.

For example, the following illustration shows the physical relationship of an indicator light and a pushbutton.

Reading a Line Diagram:

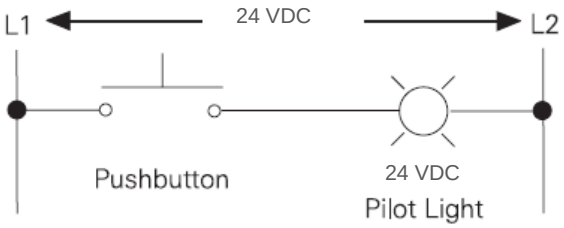
The following line diagram symbolically displays the functional relationship of these same components. In order to properly interpret this diagram, you must read it starting at L1 from left to right to L2. With that in mind, note that pressing the pushbutton allows current to flow from L1 to L2 through the pushbutton and the pilot light. Releasing the pushbutton stops current flow, turning the indicator light off.



A typical control circuit includes a control load and one or more components that determine when the control load will be energized. Some control loads, such as relays and contactors, activate other devices, but other control loads, such as indicator lights, do not.

For example, the following illustration shows the connection of an indicator light and a pushbutton. The power lines are drawn vertically and marked L1 and L2.

In this example, the voltage between L1 and L2 is 24 VDC. This means that the indicator light must be rated for 24 VDC, because, when the pushbutton is pressed, 24 VDC is applied to the indicator light.

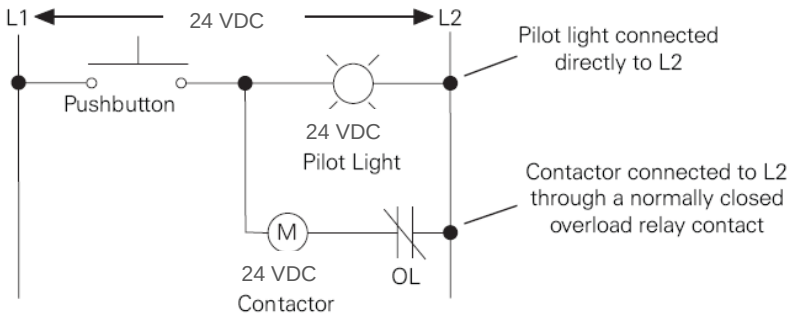


Connecting the Load to L2:

Only one control load can be placed in any one circuit line between L1 and L2. One side of the control load is either directly or indirectly connected to L2.

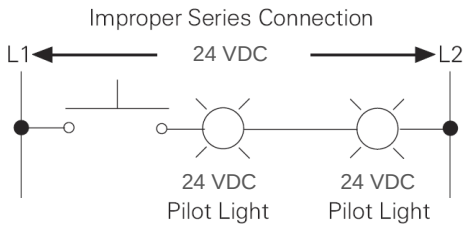
In the following example, an indicator light is directly connected to L2 on one circuit line. A contactor coil is indirectly connected through a set of overload contacts (OL) to L2 on a second, parallel circuit line.

Pressing the pushbutton applies 24 VDC to the indicator light and to the "M" contactor.

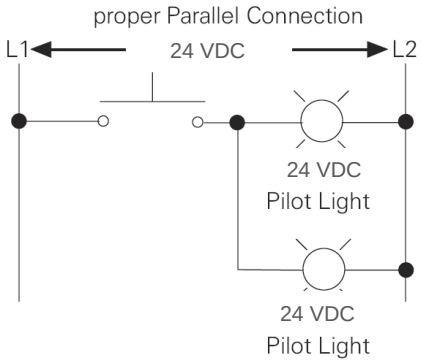


Control loads are generally not connected in series. The following illustration shows why.

In the following circuit, the control loads are improperly connected in series. When the pushbutton is pressed, the voltage across L1 and L2 is divided across both loads with neither load receiving the full 24 VDC necessary for proper operation.



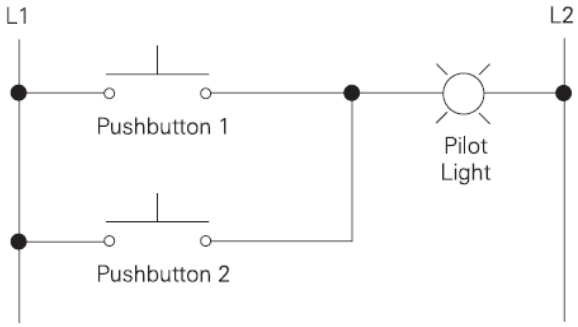
In the circuit below, the loads are properly connected in parallel, and, when the pushbutton is pressed, the full 24 VDC is applied to both loads. In addition, if one load fails in this configuration, the other load will continue to operate normally.



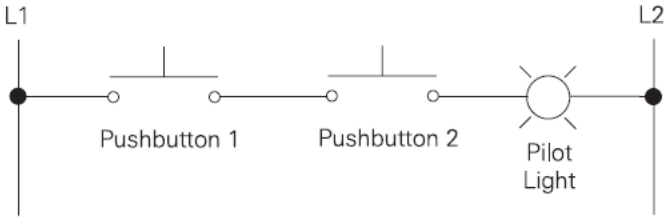
Connecting Control Devices:

In the previous example, only one control device is used to control the load. Usually more than one control device is needed. These control devices may be connected in series, parallel, or in a combination series-parallel circuit, depending on the logic required to control the load.

For example, in the following illustration, the pushbuttons are connected in parallel. Pressing either pushbutton, or both pushbuttons, allows current to flow from L1, through the indicator light, to L2.



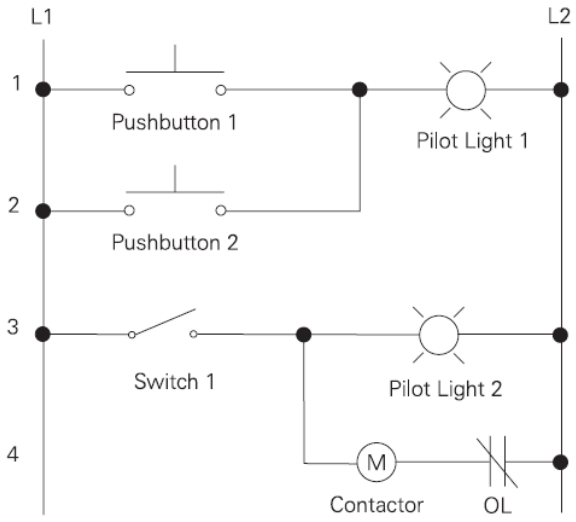
The next illustration shows two pushbuttons connected in series. Both pushbuttons must be pressed at the same time to allow current to flow from L1 through the load to L2.



Line Numbering:

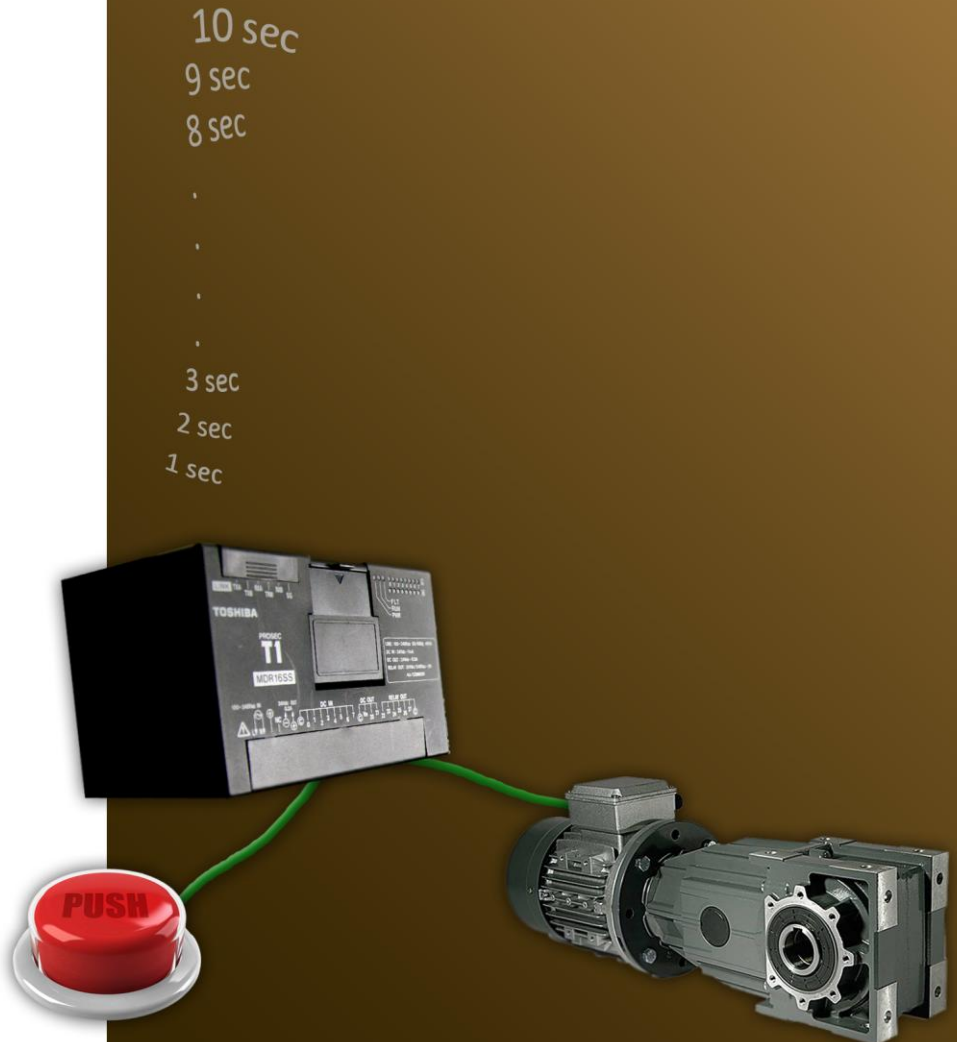
Because line diagrams often have multiple lines, the lines are often numbered to simplify describing the logic.

For example, in the following illustration, line 1 connects pushbutton 1 to pilot light 1, line 2 connects pushbutton 2 to pilot light 1, and line 3 connects switch 1 to pilot light 2 and to the "M" contactor on line 4.



Notes:

4. PLC Overview



The History of PLCs

In the 1960's Programmable Logic Controllers were first developed to replace relays and relay control systems. Relays, while very useful in some applications, also have some problems. The main problem is the fact that they are mechanical. This means that they wear down and have to be replaced every so often. Also, relays take up quite a bit of space. These, along with other considerations, led to the development of PLCs.

More improvements to PLCs occurred in the 70's. In 1973 the ability to communicate between PLCs was added. This also made it possible to have the controlling circuit quite a ways away from the machine it was controlling. However, at this time the lack of standardization in PLCs created other problems. This was improved in the 1980's.

The size of PLCs was also reduced then, thus using space even more efficiently. The 90's increased the collection of ways in which a PLC could be programmed (block diagrams, instruction list, C, etc.). They also saw PLCs being replaced by PC's in some cases. However, PLCs are still very much in use in all sorts of industries, and it's likely that they will remain there for quite some time.

What is a PLC?

A programmable Logic Controller, PLC, is a computer-type device used to control equipment in an industrial facility. In a traditional industrial control system, all control devices are wired directly to each other according to how the system is supposed to operate. In a PLC system, however, the PLC replaces the wiring between the devices. All equipment is wired to the PLC. Then, the control program inside the PLC provides the "wiring" connection between the devices.

The **control program "Software"** is the computer program stored in the PLC's memory that tells the PLC what's supposed to be going on in the system.

The Central Processing Unit "CPU" is the "Brains" of a **PLC** that retrieves, decodes, stores, and processes information. It also executes the program stored in the PLC's memory. It functions the same way of the CPU computer, except that it uses special instructions and coding. And has three parts:

- The Processor is the section of the CPU that codes, decodes, and computes data.
- The Memory system is the section of the CPU that stores both the control program and data from the equipment connected to the PLC.
- The Power Supply is the section that provides the PLC with the voltage and current it needs to operate.

Input and Output Devices

Input Device:

The PLC receives signals from various switches and sensors in the controlled machine or related equipment. Many of these signals are on/off type conditions, also called **discrete** or digital signals.

In some cases, the signals come from manual devices such as pushbuttons and selector switches. However, many discrete PLC inputs come from devices, such as limit switches or proximity switches, that are turned on and off by machine operations.

Pushbutton | **Selector Switch** | **Limit Switch** | **Proximity Switch**

In addition to discrete inputs, the PLC may also receive **analog** inputs from sensors that vary voltage or current as conditions in the machine or related equipment vary.

Inputs, as well as the current condition of PLC outputs and internal data values, are analyzed by the PLC as it executes its stored program.

The PLC uses this process to determine the signals it sends to output devices that control the operation of the machine or indicate machine conditions.

Output Device:

Just as some inputs are analog type inputs that vary in current or voltage, some outputs are analog type as well. Many outputs, however, are discrete (on/off) signals.

In some cases, these signals control equipment directly. In other cases, an intermediate device, such as a control relay or motor starter, is used.

The following illustration shows a few examples of the types of devices that may be controlled.

Indicator Light | **Solid State Relay** | **Motor Starter** | **Control Relay**

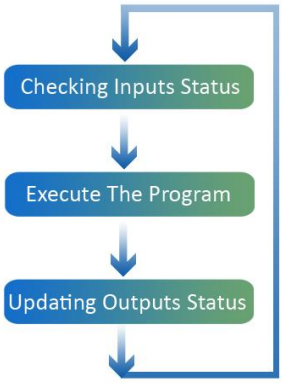
How the PLC works

Step 1, Check Input Status:

First the PLC takes a look at each input to determine if it is on or off. It records this data into its memory to be used during the next step.

Step 2, Execute Program:

One instruction at a time. Maybe your program said that if the first input was on then it should turn on the first output. Since it already knows which inputs are on/off from the previous step it will be able to decide whether the first output should be turned on based on the state of the first input. It will store the execution results for use later during the next step.



Step 3, Update Output Status:

Finally the PLC updates the status of the outputs. It updates the outputs based on which inputs were on during the first step and the results of executing your program during the second step. Based on the example in step 2 it would now turn on the first output because the first input was on and your program said to turn on the first output when this condition is true.

After the third step the PLC goes back to step one and repeats the steps continuously. One ***scan time*** is defined as the time it takes to execute the 3 steps listed above.

Scanning Time

Each PLC operational cycle is made up of three separate parts:

1. Input scan: Input terminals are read and the input status table is updated accordingly.
2. Program scan: Data in the input status table is applied to the program, the program is executed and the output status table is updated
3. Output scan: Data with the output status table is transferred to output terminals.

The sum of the three above times is called **Scanning Time.**

Why using PLC?

Many features and benefits obtained by using a programmable controller as follows:

Inherent Features	Benefits
Solid-state components	- High Reliability
Programmable memory	- Simplifies changes - Flexible control
Small Size	- Minimal space requirements
Microprocessor-based	- communication capability - Higher level of performance - Higher quality products - Multifunctional capability
Software timers/counters	- Eliminate hardware - Easily changed presets
Software control relays	- Reduce hardware/wiring cost - Reduce Space requirements
Modular architecture	- Installation flexibility - Easily installed - Reduces hardware costs - Expandability
Variety of I/O interfaces	- Controls a variety of devices - Eliminates customized control
Remote I/O Stations	- Eliminate long wire runs
Diagnostics indicators	- Reduce troubleshooting time
Modular I/O interface	- Neat appearance of panel - Easily maintained - Easily wired
Quick I/O disconnect	- Service w/o disturbing wiring

Without question, the “programmable” feature provides the single greatest benefit for the use and installation of programmable controllers. Eliminating hardwired control in favor of programmable control is the first step towards achieving a flexible control system.

Once installed, the control plan can be manually or automatically altered to meet day-to-day control requirements without changing the field wiring. This easy alteration is possible since there are no physical connections between the field input devices and output devices, as in hardwired systems. The only connection is through the control program, which can be altered.

I/O Modules

The **input/output (I/O) system** is the section of a PLC to which all of the field devices are connected. The I/O system is what actually physically carries out the control commands from the program stored in the PLC's memory.

Inputs are devices that supply a signal/data to a PLC. Typical examples of inputs are push buttons, switches, and measurement devices.

Outputs are devices that await a signal/data from the PLC to perform their control functions. Lights, horns, motors, and valves

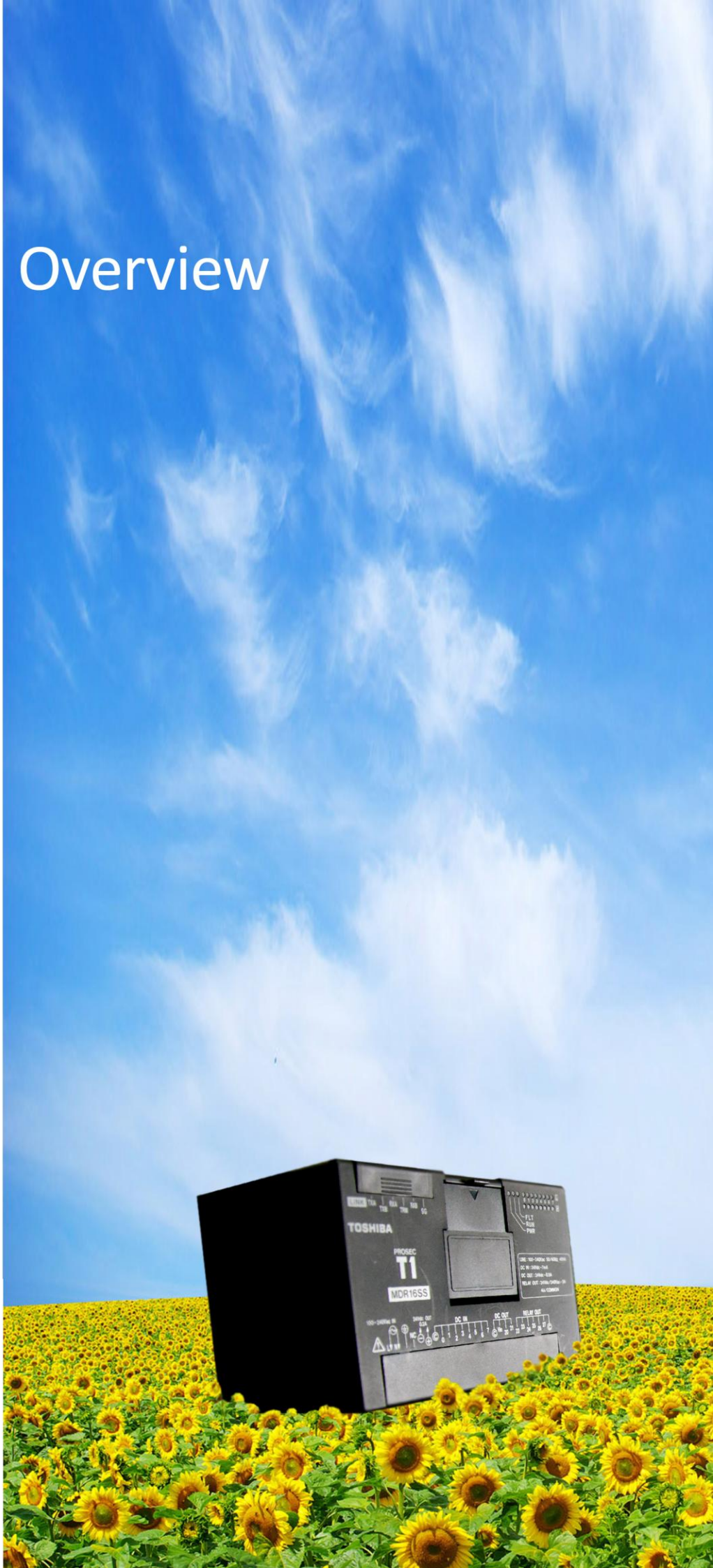
I/O module types:

- Digital
- Analogue
- Thermocouple
- Network
- High speed counter
- Position control
- Pulse input

How to select a PLC?

- Determine the type of control (Distributed, Centralized or Individual control).
- Determine the number of digital and analog inputs and outputs.
- Determine input/output specifications.
- Determine remote I/O requirements.
- Determine special I/O requirements.
- Future expansion plans.

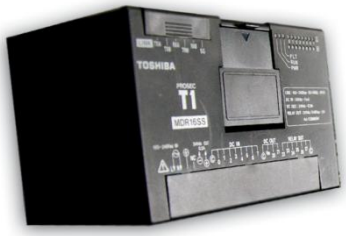
5. Hardware Overview



TOSHIBA T1-16S H/W

T1-16S PLC

T1-16S is a micro programmable controller with optional add-on expansion I/O modules. Its communications capability, advanced instruction set, and large memory (program and data register) make it ideally suited for applications that requiring large more expensive programmable Controllers.



T1-16S Main Unit

Power supply : 100 – 240 Vac or 24 Vdc
Input : 8 points, 24 Vdc
Output : 8 points, 6 relay & 2 transistor

T1-16S I/O Modules

- TDI116: 16 Input Points
- TDD116: 8 Input + 8 Output
- TRO108: 8 Output Relay points
- TDO116: 16 Output Transistor points
- TAD121: 1 Channel Analog Input, 0 to 5V/0 to 20mA
- TAD131: 1 Channel Analog Input, -10 to 10V
- TDA121: 1 Channel Analog Output, 0 to 5V/0 to 20mA
- TDA131: 1 Channel Analog Output, -10 to 10V
- TTC111: 1 Channel Thermocouple Input (K,J,E)
- TFR112: Remote station, 1 words input & 1 word output

Features

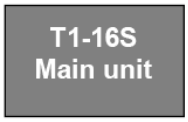
Large Program Memory

The T1-16S is a micro PLC, but it has a large program and data memory:

- 8192 Steps of Program Memory
- 4096 Words of Data register
- 256 Timers
- 256 Counters
- 144 Digital Points I/O
- 8 Analog Channels

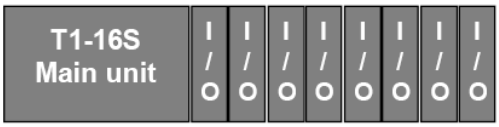
Expansion Capacity

< Basic configuration >



Main Unit
(Discrete = 8 inputs & 8 outputs)

< Expansion configuration >



Main Unit + 8 I/O Modules
(Max Discrete I/O = 16 points + 8 x 16 points = 144 points)
(Max Analog I/O = 8 channels)
Any mix of discrete & analog modules maybe used.

On-Line Program Changes

It is possible to make program changes and write to the internal EEPROM while the Unit is executing the user application program. This feature is very important in process control applications where it is not always possible to stop the process.

Real Time Clock/ Calendar

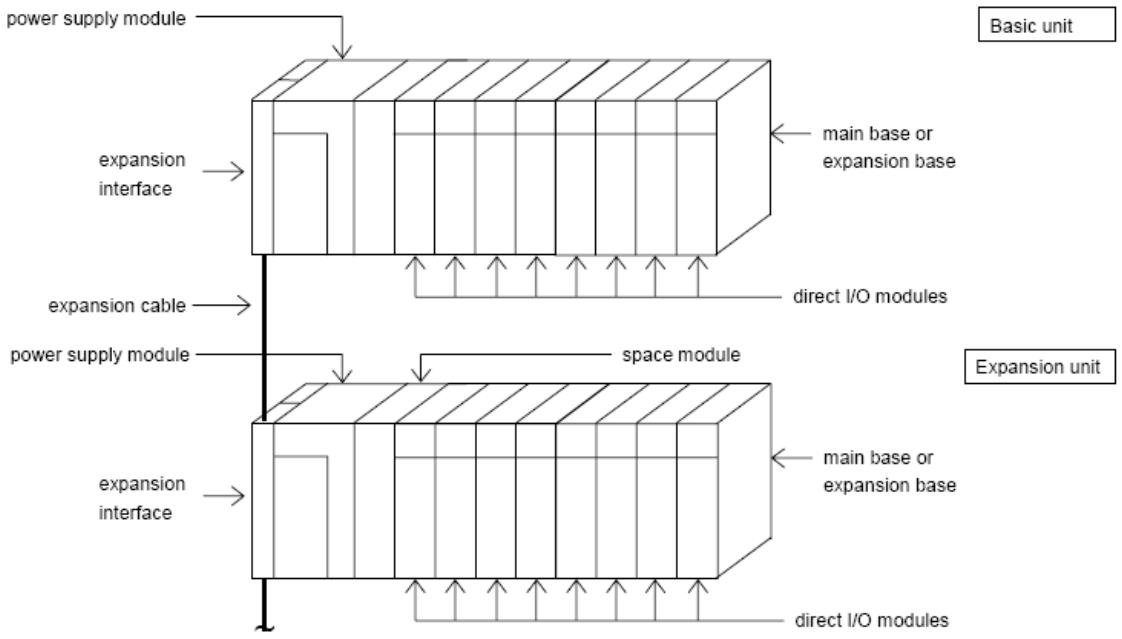
It has real time clock/calendar for year, month, day, hour, minute and second. It can be used for performing scheduled operations, it's backed up by built in capacitor.

TOSHIBA S2E H/W

Basic configuration

According to variety of industrial applications, a basic unit or with expansion units are selected. The basic unit is composed of a main base, a power supply module, direct I/O modules, and an expansion interface is added using expansion units.

The expansion unit is composed of an expansion base, a power supply module, direct I/O modules and an expansion interface. Each expansion interface modules are connected with expansion cables. Up to three expansion units can be connected in the S2E. The S2E accesses direct I/O modules via the G2-bus in the model 2000.



S2E Modules

S2E CPU Module

Type	Description
PU612E	Program memory: 32Ksteps I/O points: 2048 points

Power Supply Module

Type	Description
PS693	100 to 240Vac input
PS692	24Vdc input
PS652	100 to 110Vdc input

Base (main/expansion units common)

Type	Description
BU668	8 I/Os at both main and expansion units
BU666	5 I/Os at main / 6 I/Os at expansion
BU664	3 I/Os at main / 4 I/Os at expansion

Expansion I/O Interface

Type	Description
IF661	For main and expansion units common

Digital Input Module

Type	Description
DI632D	8 points (Isolated), 12 to 24Vdc/ac input
DI633	16 points, 12 to 24Vdc/ac input
DI634	32 points, 24Vdc input
DI635	64 points, 24Vdc input
DI635H	64 points, 24Vdc input (High Speed)
DI653	16 points, 100 to 110Vdc input
IN653	16 points, 100 to 120Vac input
IN663	16 points, 200 to 240Vac input

Digital Output Module

Type	Description
DO633	16 points, 24Vdc output
DO633P	16 points, 24Vdc output (current source)
DO634	32 points, 24Vdc output
DO635	64 points, 24Vdc output
AC663	12 points, 100 to 240Vac
RO663	16 points, relay output, max. 240Vac/24Vdc
RO662S	8 points (isolated), relay, max. 240Vac/24Vdc

Network Module

Type	Description
FL612	FL-net controller station (Ver. 2.0)
SN621	TOSLINE-S20, co-axial cable type
SN622	TOSLINE-S20, optical cable type (bus)
DN611A	DeviceNet scanner module
UN611	TOSLINE-F10 master station
UN612	TOSLINE-F10 remote station

Misc

Type	Description
SP600	Empty slot dummy module
BT611	Spare lithium battery

Analog input module

Type	Description
AD624L	4 channel, 1-5/4-20mA, 8-bit
AD634L	4 channel, 0-10V, 8-bit
AD624	4 channel, 1-5/4-20mA, 12-bit
AD674	4 channel, ±10V, 12-bit
AD668	8 channel, ±10V/1-5/4-20mA, 16-bit
AD628S	8 channel (isolated), 0-5V/0-20mA, 12-bit
AD638S	8 channel (isolated), 10V, 12-bit
RT614	4 channel, Pt100 input, 12-bit
TC618	8 channel, TC (K/J/E) input, 12-bit, hold

Analog output module

Type	Description
DA622L	2 channel, 1-5/4-20mA, 8-bit
DA622	2 channel, 1-5/4-20mA, 12-bit
DA672	2 channel, ±10V , 12-bit
DA664	4 channel, ±10V/1-5/4-20mA, 16-bit
DA624S	4 channel (isolated), 0-20mA, 16-bit, hold

Special I/O module

Type	Description
PI632	2 channel pulse input, 100kpps, 5/12/24Vdc
PI672	2 channel pulse input, 100kpps, RS422
CD633	16 points, change detect, 12 to 24Vdc input
MC612	2 axis positioning, pulse output, 200kpps
MC614	4 axis positioning, pulse output, 1.3Mpps
CF611	Communication interface, RS232C, 1 port

I/O Expansion Cable

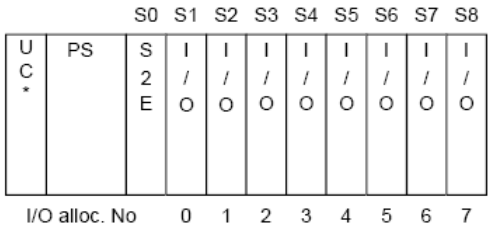
Type	Description
CS6R3	0.3m length
CS6R5	0.5m length
CS6R7	0.7m length
CS6*1	1.2m length

Unit configuration

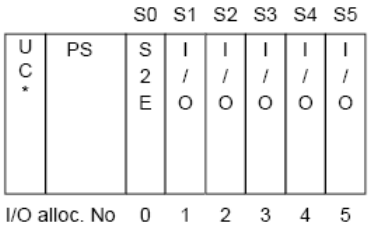
Some examples of minimum/maximum configuration are shown as below. The control module S2E should be mounted on S0 that is the left end slot of the basic unit.

Minimum Configuration:

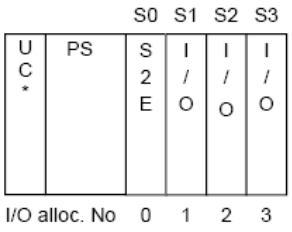
BU668:



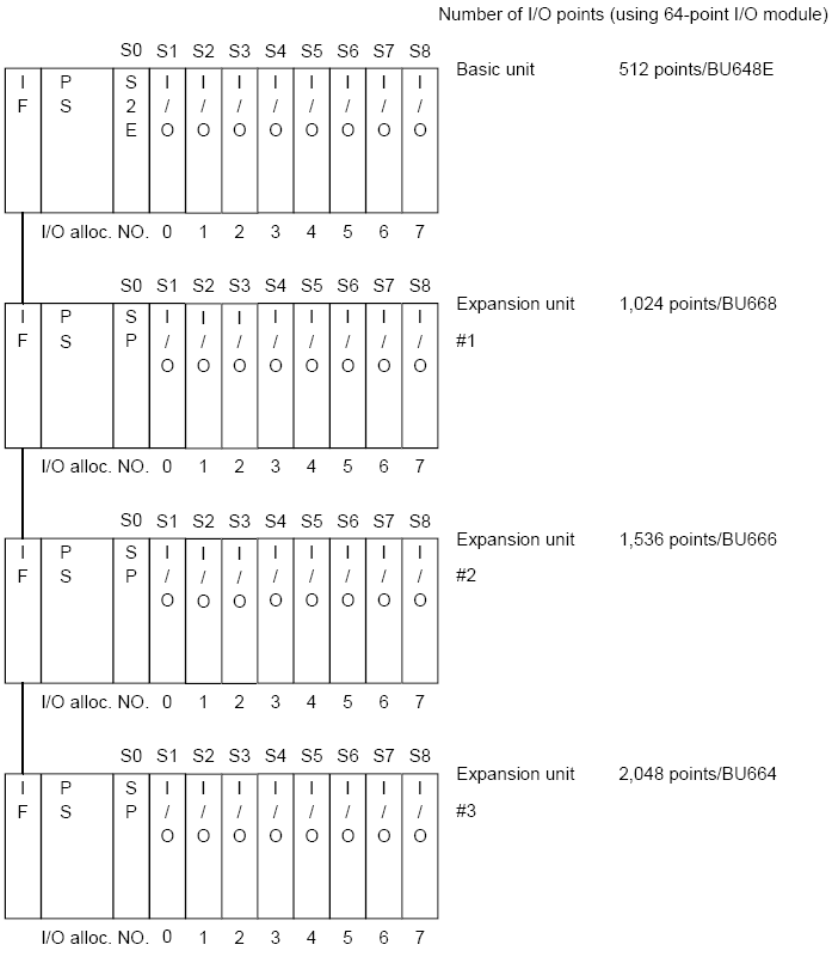
BU666:



BU664:



Maximum Configuration:

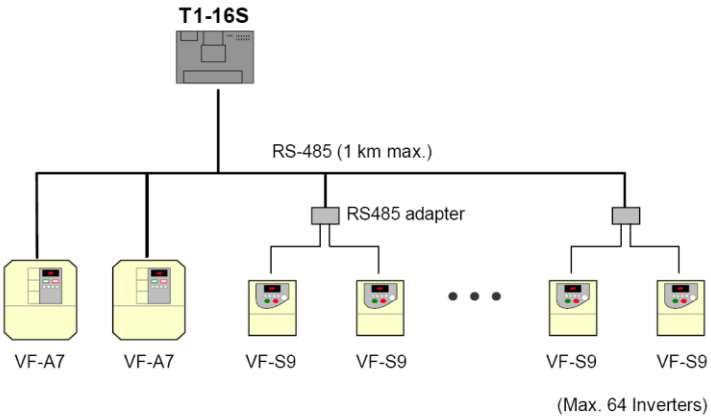


Communication Features and Application

Easy Connection Inverters

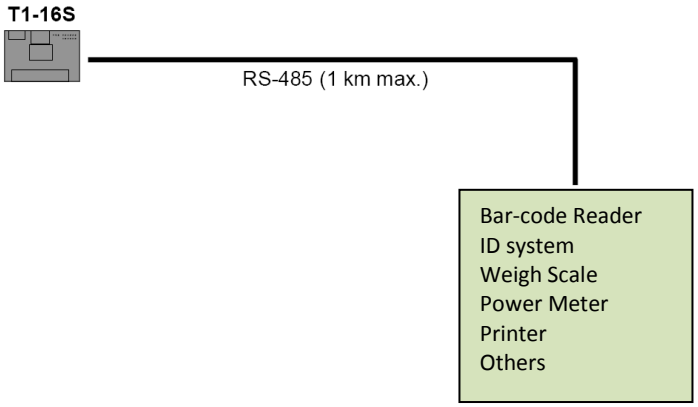
The RS485 port supports a special mode to exchange data with TOSHIBA Inverter:

Monitoring: Operation frequency and terminal status
Controlling: Run, Stop, Jog, Forward, Reverse, ..etc.



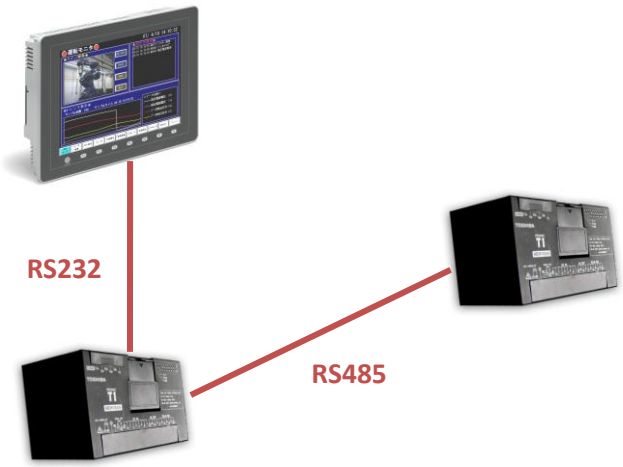
Flexible Comm. Interface

The RS485 port supports a flexible communicating function, Free ASCII mode. This free ASCII mode lets the T1-16S/S2E act as a master to communicate with other field devices that have a serial ASCII communication function.



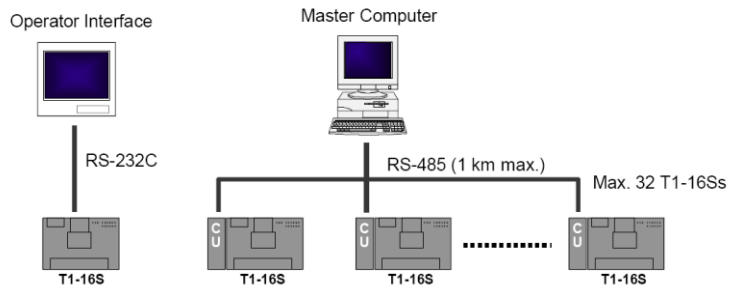
Easy Data Linkage between two T1-16S/S2E and Touch Screens

The RS485 port supports the data link mode to exchange data between 2 units without using special program. Also the RS232 could be used to support Touch Screens.



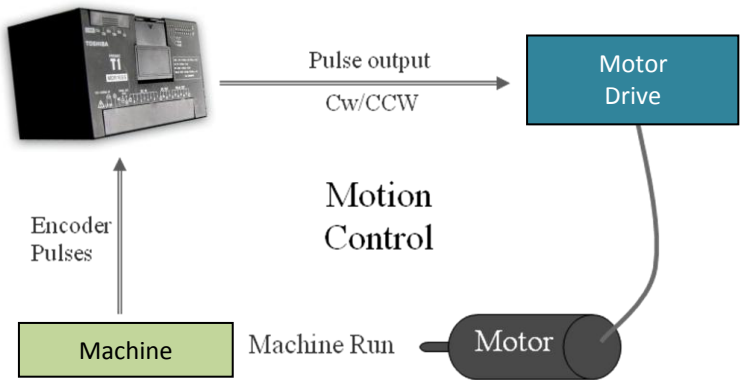
SCADA System/Remote Programming

By using the RS485 port up to 32 T1-16S can communicate with a computer or other higher level controller. Most major SCADA software supports the T-series PLC communication protocol.



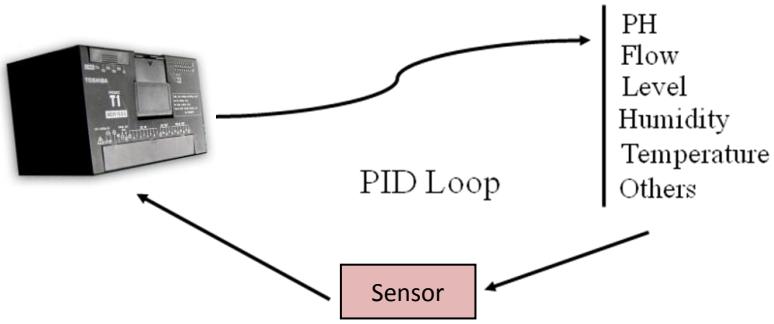
Motion Control

The T1-16S has two high speed counter inputs. The 5K pulses per second used independently for the applications needs high speed compare, reset, and strobe. The counters can be used together as a quadrature counter to count a 2-phases pulse encoder mode; in this case the countering speed is max. 20K cps (counter per second). The 5K pulses per second also can generate variable frequency pulse trains CW and CCW; this output can be used to drive a stepper motor.



Process Control

The T1-16S has a PID (Proportional, Integral and Derivative) function, this PID and analog I/O modules enable the T1-16S to be applied to many process control applications. Multiple PID loops can be used.



6. Getting Started with T-PDS

Installing the T. PDS Windows version 2.0:

For Microsoft Windows 95 or higher:

1. Insert the CD in the CD Rom drive.
2. Setup program will run automatically.
3. Follow the instructions that appear on the screen.
4. The setup program will creates a group program items in the program manager.
5. The name of the program group is T-PDS for windows V2.0.
6. Run the T-PDS program and Enjoy with it.

T-PDS Windows Menu Structure

Top Menu

Item	File	Edit	View	Search	PLC	Debug	Option	Window	Help
Level	1	2	3	4	5	6	7	8	9

Level	Item	Selection Result
1	File	Sub Menu: New Project, Open Project, Save Project, Save Project As, Close Project, Compare Project, Multiple Project, Print, Print Setup, Transfer Program, Exit.
1-1	New Project	Opens the PLC Type Dialog Box.
1-2	Open Project	Opens the File Search Dialog Box.
1-3	Save Project	Opens the Execute Dialog Box.
1-4	Save Project As	Opens the Save As File Location Dialog Box.
1-5	Close	Closes the Current Project.
1-6	Compare Project	Opens the File Selection Box to Locate the Second Project.
1-7	Multiple Project	Opens a 2nd T-PDS Windows Program.
1-8	Print	Opens the Print Dialog Box.
1-9	Print Setup	Opens the Print Setup Dialog Box.

Level	Item	Selection Result
1-10	Transfer	Program Sub Menu: PLC -> File, File -> PLC.
1-11	Exit	Exits the T-PDS Windows Program.
2	Edit	Sub Menu: Edit Mode, Cut, Copy, Paste, Insert, Delete, Direct, Edge/Digit/Index, Change Language, Block, Function, Comment, Check Program, Write.
2-1	Edit Mode	Changes to the Edit Mode for Creating or Modifying Ladder logic.
2-2	Cut	Deletes the Current Selection and Copies it to the Clipboard.
2-3	Copy	Copies the Current Selection to the Clipboard.
2-4	Paste	Pastes the Contents of the Clipboard to the Current Cursor Position.
2-5	Insert	Sub Menu: Line, Rung, Column.
2-6	Delete	Sub Menu: Line, Rung, Column.
2-7	Edge/Digit/Index	Sub Menu: Edge, Digit, Index.
2-8	Change Language	Changes between Ladder Logic (LL) and Sequential Function Chart (SFC).
2-9	Block	Sub Menu: Block Edit, Block Merge, & Block Divide.
2-10	Function	Sub Menu: Change Device & Replace Address
2-11	Comment	Sub Menu: Block Comment, Rung Comment, & Reg/Dev Comment.
2-12	Check Program	Checks the syntax of the program.
2-13	Write	Saves the Edited Block to Disk in the Off-line Mode and to the PLC in the On-Mode.

Level	Item	Selection Result
3	View	Sub Menu: Tool Bar, Status Box, Data Box, Auxiliary Monitor, Data Monitor, Data Format, Comment Format, Trace Format, Zoom In, & Zoom Out.
3-1	Tool Bar	Toggles the Tool Bar Off and On.
3-2	Status Box	Toggles the Status Box Off and On.
3-3	Data Box	Displays the Data Box.
3-4	Auxiliary Monitor	Displays the Auxiliary Box in the On-Line Mode.
3-5	Data Monitor	Opens the Data Monitor Screen.
3-6	Data Format	Opens the Data Format Dialog Box.
3-7	Comment Format	Displays the view comment selection box. Select Block comment, Rung comment, Data register/device comment, or Tag name.
3-8	Trace Format	Sub Menu: Line, Rung, Column.
3-9	Zoom In	Zooms In on the Current Block.
3-10	Zoom-Out	Zooms Out on the Current Block.
4	Search	Sub Menu: Find, Rung, Block, & GoTo.
4-1	Find	Opens the Find Dialog Box.
4-2	Rung	Defines the First Rung and the Last Rung of the Search Range.
4-3	Block	Sub Menu: Start of Program, End of Program, Previous Block, & Next Block.
4-4	Goto	Opens the GoTo Dialog Box.

Level	Item	Selection Result
5	PLC	Sub Menu: System Parameters, I/O Allocation, Event History, Scan Time, Power Interruption, Memory Management, Password, PLC Control, Online/Offline.
5-1	System Parameters	Opens the System Parameters Dialog Box (ID, Retentive Memory, .Error Status, Diagnostic Messages, etc).
5-2	I/O Allocation	Sub Menu: I/O Allocation, Interrupt Assignment, Network Assignment.
5-3	Event History	Displays the Event History Dialog box, a running record with date and time (if controller has RTC) of operation and error status.
5-4	Scan Time	Displays the Scan Time for the current program in the Online Mode.
5-5	Power Interruption	Displays a time and date listing of power interruption for T2 and T3 CPUs.
5-6	Memory Manag.	Sub Menu: Clear Event, Clear Memory, Clear I/C Card, Read EEPROM/IC Card, Write EEPROM/IC Card.
5-7	Password	Sub Menu: Change Protect Level, Set Password.
5-8	PLC Control	Sub Menu: Halt, Run, Force Run, Error Reset, Hold, Hold Cancel, Float Box.
5-9	Online/Offline	Toggles between Online and Offline Mode.
6	Debug	Sub Menu: Force, Set On/Off, Change Value, Sample Trace, Status Latch, Data Validity Check.
6-1	Force	Forces/Releases the device selected by the cursor.
6-2	Set On/Off	Allows a Forced Device to be set ON and OFF.
6-3	Change Value	Displays the Data Box for the register. Set register value, size, and type.
6-4	Sample Trace	Displays the sample trace setup box.

Level	Item	Selection Result
7	Option	Sub Menu: Cross Reference, Usage Map, Forced List, Select Comment File, Instruction Box, Communication.
7-1	Cross Reference	Opens the Cross Reference setup box which generates the Cross Reference List.
7-2	Usage Map	Opens the Program Range setup box that generates the Usage List.
7-3	Forced List	Opens the Program Range setup box that generates the Forced Devices List.
7-4	Select Comment File	Opens the select comment box; program comment file, data comment file, or browse.
7-5	Instruction Box	Opens the instruction box; select tool bar or float box, and box line size.
7-6	Communication	Opens the type of communication selection box; Direct, Computer Link, or Network.
8	Window	Sub Menu: New Window, Cascade, Tile, Arrange Icons, Close All.
8-1	New Window	Opens the New Window selection box.
8-2	Cascade	Cascades all open windows from the top left.
8-3	Tile	Arranges all open windows from top to bottom starting with the first window opened.
8-4	Arrange Icon	
8-5	Close All	Closes all open widows.

Level	Item	Selection Result
9	Help	Contents, Search on Help, Index, Technical Support, About T-PDS for Windows.
9-1	Contents	Table of Contents for T-PDS Help.
9-2	Search	Opens the windows search box.
9-3	Index	Displays the Help Index.
9-4	Technical Support	Explains available technical support on T-Series PLCs.
9-5	About TPDS	Opens the T-PDS & Computer Statistics Box.

Notes:

7. Ladder Instructions

```
                                                    Y0022 |
 |X0000 X0001                                      -( )--|
1|-| |-+-|/|-------------------------------------------|
 |     |                                              |
 |Y0022|                                              |
 |-| |-+                                              |
 |                                                    |
 |                                                    |
2|[END ]-------------------------------------------|
```

I/O assignment

Devices and registers

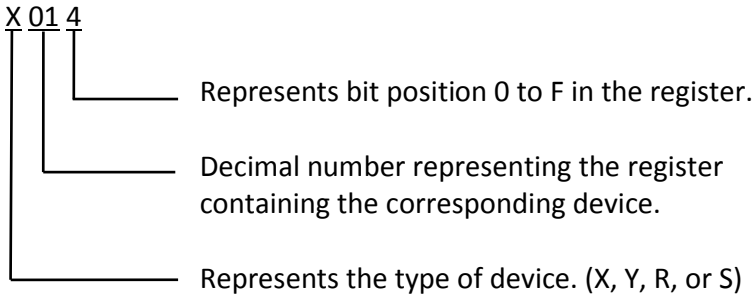
The T series program consists of bit-based instructions that handle ON/OFF information, such as contact and coil instructions, and register-based (16-bit) instructions, such as those for data transfer and arithmetic operations.

Devices are used to store the ON/OFF information of contacts and coils, Registers are used to store 16-bit data.

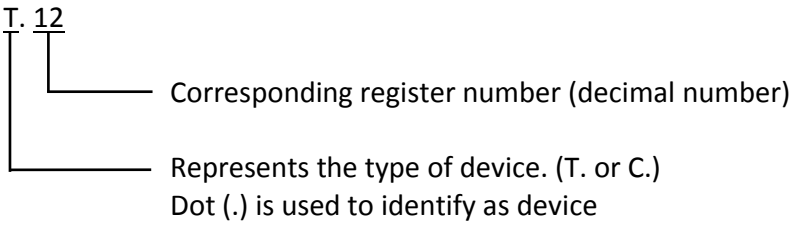
Devices	Registers
• X External input • Y External output • R Auxiliary relay • S Special devices • T. Timer devices • C. Counter devices	• XW External input registers • YW External output registers • RW Auxiliary relay registers • SW Special registers • T Timer registers • C Counter registers • D Data registers • I, J, K Index registers

Addressing devices

A device number of X, Y, R and S devices consists of a register number and bit position as follows.

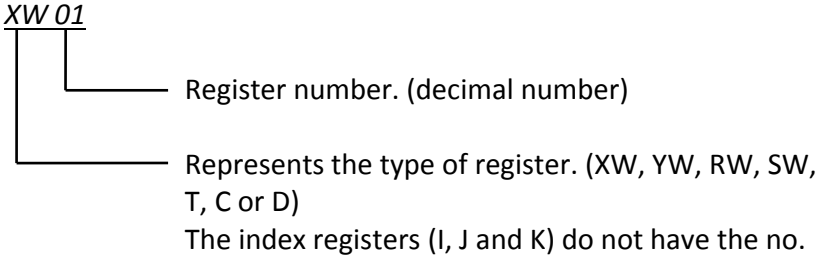


As for the timer (T.) and the counter (C.) devices, a device number is expressed as follows.



Addressing register

A register number except the index registers is expressed as follows.



Special Function Devices and registers

Register/Device	Name	Function	
S000	T1/T1S operation mode	0: Initialization 1: HALT mode 2: RUN mode 3: RUN-F mode 4: HOLD mode 6: ERROR mode	
S001			
S002			
S003			
S004	CPU error	ON at error state	
S005	I/O error	ON at error state	
S006	Program error	ON at error state	
S007	EEPROM Alarm	ON when EEPROM write exceeds 100,000 times	
S008	Fixed-time scan time-over	ON when actual scan time is longer than the setting time as fixed-time scan	
S00A	Clock/calendar error	ON when clock/calendar data is illegal	
S00D	TL-F10 error	ON when TOSLINE-F10 transmission error occurs	
S00F	Retentive data invalid	ON when retentive data in RAM are invalid	
S010	System ROM error	ON at error state	
S011	System ROM error	ON at error state	
S012	Program memory error	ON at error state	
S013	EEPROM error	ON at error state	
S021	I/O mismatch	ON at error state	
S030	Program error	ON at error state	
S031	Scan time over	ON when the scan time exceeds 200 ms	
S040	Timing relay 0.1 s	OFF 0.05 s / ON 0.05 s (0.1 s interval)	All OFF at the Beginning of Run Mode
S041	Timing relay 0.2 s	OFF 0.1 s / ON 0.1 s (0.2 s interval)	
S042	Timing relay 0.4 s	OFF 0.2 s / ON 0.2 s (0.4 s interval)	
S043	Timing relay 0.8 s	OFF 0.4 s / ON 0.4 s (0.8 s interval)	
S044	Timing relay 1.0 s	OFF 0.5 s / ON 0.5 s (1.0 s interval)	
S045	Timing relay 2.0 s	OFF 1.0 s / ON 1.0 s (2.0 s interval)	
S046	Timing relay 4.0 s	OFF 2.0 s / ON 2.0 s (4.0 s interval)	
S047	Timing relay 8.0 s	OFF 4.0 s / ON 4.0 s (8.0 s interval)	
S04E	Always OFF	Always OFF	
S04F	Always ON	Always ON	

Register/Device	Name	Function
SW07	Clock/calendar (Year)	Lower 2 digits of the calendar year (01, 02, ...)
SW08	Clock/calendar (Month)	Month (01, 02, ... 12)
SW09	Clock/calendar (Day)	Day (01, 02, ... 31)
SW10	Clock/calendar (Hour)	Hour (00, 01, ... 59)
SW11	Clock/calendar (Minute)	Minute (00, 01, ... 59)
SW12	Clock/calendar (Second)	Second (00, 01, ... 59)
SW13	Clock/calendar (Week)	Day of the week (Sun = 00, Mon = 01, ... Sat = 06)
SW16	Mode of special input functions	Used to select the special input functions
SW17	Input filter constant	Used to set the input filter constant
SW18	Preset values for high speed counter	Used to set the preset values for high speed counters
SW19		
SW20		
SW21		
SW22	Count values for high speed counter	Preset count values of the high speed counters are stored
SW23		
SW26	Mode of special output functions	Used to select the special output functions
SW28	Special output frequency setting	Output frequency setting for the pulse/PWM output
SW30	Analog setting value 1	Input value of the analog setting adjuster V0
SW31	Analog setting value 2	Input value of the analog setting adjuster V1
SW34	TL-F10 send data	TOSLINE-F10 transmission data (send to master)
SW35	TL-F10 receive data	TOSLINE-F10 transmission data (receive from master)
SW54	Basic unit I/O LED display mode	Used to display the selected I/O module status (0 = Basic unit, 1 to 8 = I/O module slot 0 to 7, 9 and 10 = TOSLINE-F10)
SW55	Number of EEPROM write data	Used to set the number of data registers to be saved in the EEPROM (0 to 2048, initial value is 2048)
SW56	RS-485 port operation mode	Used to set the RS-485 port operation mode (0 = Computer link, 1 = Data link, 2 = Free ASCII, 3 = Inverter connection)
SW57	RS-485 port response delay	Used to set the RS-485 port response delay time (0 to 30: 0 to 300ms)

Basic Instructions

Function: **Normally Open Contact**

Expression:

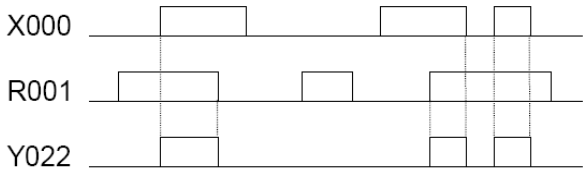
```
        A
Input --I  I-- Output
```

Function Description: When the input is ON and the device *A* is ON, the output is turned ON.

Example:

```
      X0000     R0001                                                  Y0022
|-------I  I---------I  I-----------------------------------------------(   )--------|
```

Coil Y0022 comes ON when the devices X0000 and R0001 are both ON.



Function: **Normally Closed Contact**

Expression:

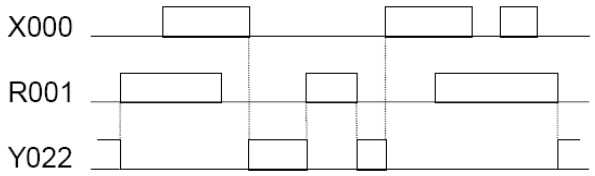
```
          A
Input --I / I-- Output
```

Function Description: When the input is ON and the device *A* is OFF, the output is turned OFF.

Example:

```
      X0000      R0001                                                Y0022
|-------I / I---------I / I-----------------------------------------------(    )--------|
```

Coil Y0022 comes ON when the devices X0000 and R0001 are both OFF.



Function: **Coil**

Expression:

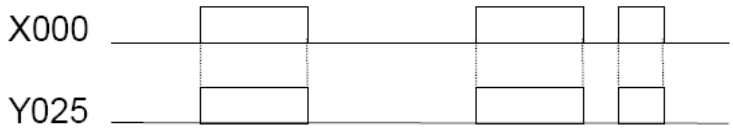
```
          A
Input –(  )--|
```

Function Description: Relay coil of device *A.* When the input is ON, the device *A* is set to ON.

Example:

```
     X0000                                                                    Y0022
|-------I  I-----------------------------------------------------------------------(    )------|
```

Coil Y0022 comes ON when the devices X0000 is ON



Function: **Invert Coil**

Expression:

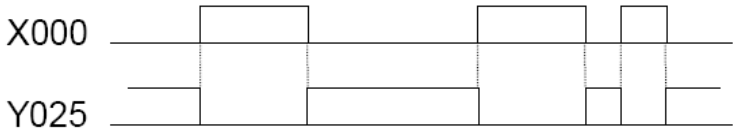
```
          A
Input --( | )--|
```

Function Description: When the input is OFF, the device A is set to ON, and when the input is ON, the device A is set to OFF. This instruction inverts the input state and store it in the device A.

Example:

```
      X0000                                                                  Y0025
|-------I  I-----------------------------------------------------------------( I )--------|
```

Y0025 comes ON when X0000 is OFF & Y0025 comes OFF when X0000 is ON.



Function: **Transitional contact (Rising edge)**

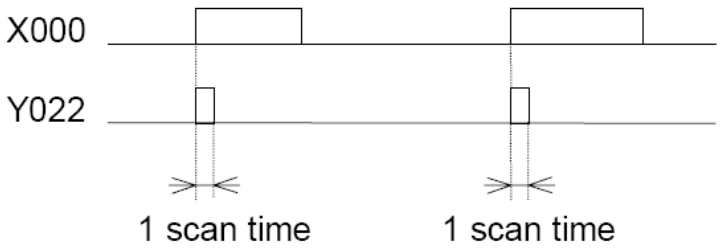
Expression: Input --I↑I-- Output

Function Description: When the input at last scan is OFF and the input at this scan is ON, the output is turned ON. This instruction is used to detect the input changing from OFF to ON.

Example:

```
      X0000                                                    Y0022
|-------I  I---------I↑I-----------------------------------------------------(    )--------|
```

Coil Y022 comes ON for only 1 scan when the device X000 comes ON.



Note: The maximum usable number in a program is 2048.
(-I↑I- -I↓I- -IPI- -INI- -(P)-| -(N)-| total)

Function: **Transitional contact (falling edge)**

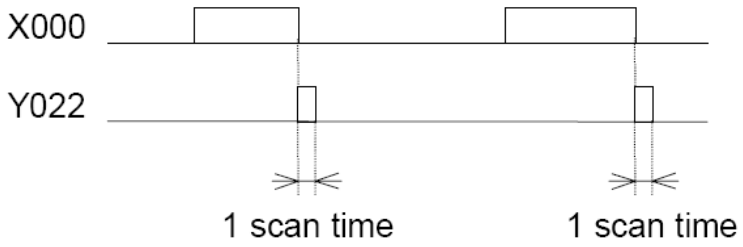
Expression: Input --I↓I-- Output

Function Description: When the input at last scan is ON and the input at this scan is OFF, the output is turned ON. This instruction is used to detect the input changing from ON to OFF.

Example:

```
      X0000                                                    Y0022
|-------I  I---------I↓I-----------------------------------------------------(    )--------|
```

Coil Y022 comes ON for only 1 scan when the device X000 comes OFF.



Note: The maximum usable number in a program is 2048.
(-I↑I- -I↓I- -IPI- -INI- -(P)-| -(N)-| total)

Function: **Forced Coil**

Expression:

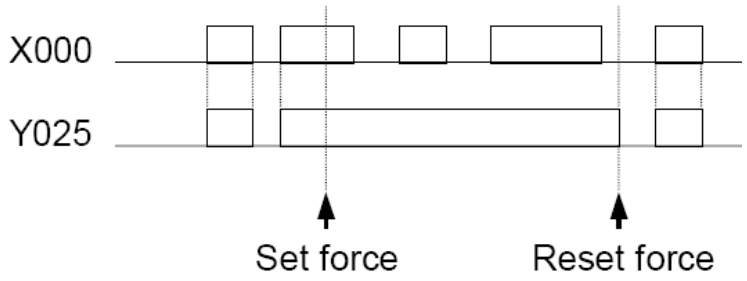
```
          A
Input –X-(  )--|
```

Function Description: Regardless of the input sate the state of device *A* is retained.

Example:

```
     X0000                                                          Y0025
|-------I  I-----------------------------------------------------------X--(   )--------|
```

Device Y025 retains the preceding state regardless of the devices X000 state.



Note: The forced coil is a debugging function. The state of a forced coil device can be set ON or OFF by the programming tool.

Function: **Inverter**

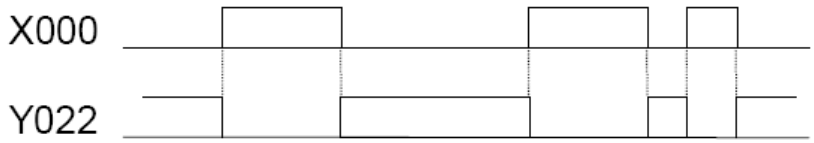
Expression: Input –I ❙ I-- Output

Function Description: When the input is OFF, the output is turned ON, and when the input is ON, the output is turned OFF. This instruction inverts the link state.

Example:

```
      X0000                                                    Y0022
|-------I  I---------I ❙ I-----------------------------------------------------(    )--------|
```

Y022 comes ON when X000 is OFF, and Y022 comes OFF when X000 is ON.



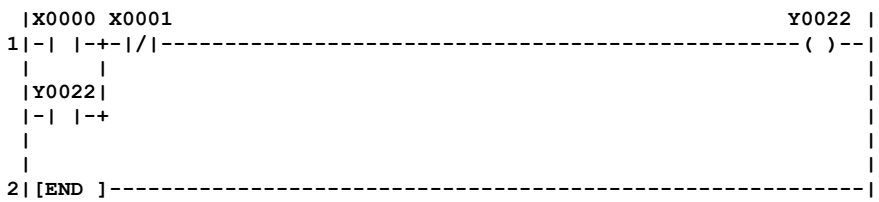
Sample Program: **Motor Start / Stop**

Description: A simple circuit with two push buttons as contacts and one output as a coil. As the first push button is pressed, the Output goes on and the motor starts to run. When the second push button is pressed, the output goes off and the motor stops.

Assignment:

Signal	PLC address	State
Start	X0	I/P
Stop	X1	I/P
Motor	Y22	O/P

Ladder:



Function: **ON delay timer**

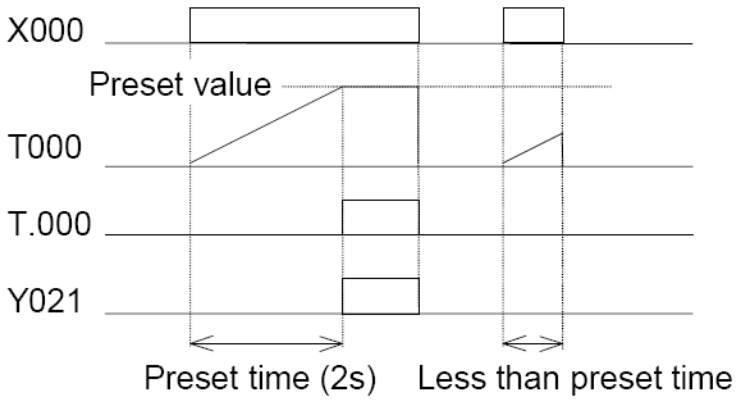
Expression: Input-[A TON B]-Output

Function Description: When Input OFF to ON, timer register *B* is started. The elapsed time is stored in *B*. When the specified time by *A* has elapsed, the output and the timer device corresponding to *B* are turned ON. When Input ON to OFF, *B* is cleared to 0.

Example:

```
     X0000                                                          Y0021
|-------I  I-------[ 00200  TON   T0000 ]---------------------------------(     )-------|
```

Y0021 (and the timer device T.0000) is turned ON 2 seconds after X0000 came ON.



Note:
- Time is set in 10 ms units for: T000 to T063 (0 to 327.67 s)
- Time is set in 100 ms units for: T064 to T255 (0 to 3276.7 s)
- Multiple timer instructions (TON, TOF or SS) with the same timer register are not allowed.

Function: **OFF delay timer**

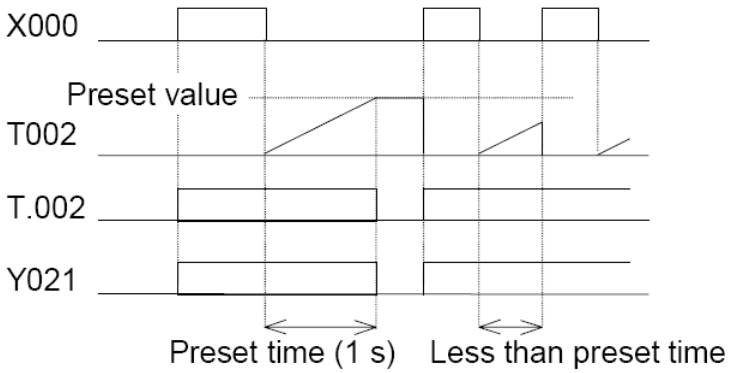
Expression: Input-[A TOF B]-Output

Function Description: When Input OFF to ON, the output and the timer *B* are set to ON. When Input ON to OFF, timer updating for *B* is started. The elapsed time is stored in *B*. When *A* has elapsed, the output and the timer turned OFF.

Example:

```
     X0000                                                          Y0021
|-------I  I-------[ 00100  TOF   T0002 ]---------------------------------(    )--------|
```

Y0021 (and the timer device T.0002) is turned OFF 1 second after X0000 came OFF.



Note:
- Time is set in 10 ms units for: T000 to T063 (0 to 327.67 s)
- Time is set in 100 ms units for: T064 to T255 (0 to 3276.7 s)
- Multiple timer instructions (TON, TOF or SS) with the same timer register are not allowed.

Function: **Single Shot timer**

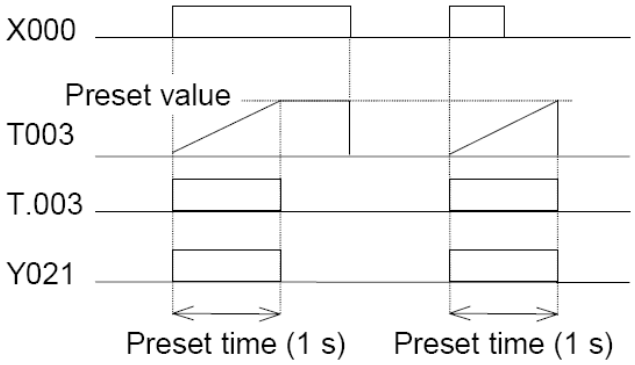
Expression: Input-[A SS B]-Output

Function Description: When Input OFF to ON, the output and the timer *B* are set to ON, and timer updating for *B* is started. When the specified time by *A* has elapsed after the input came ON, the output and the timer device are turned OFF.

Example:

```
     X0000                                                                Y0021
|-------I I---------[ 00100  SS   T0003 ]-----------------------------------(    )-------|
```

Y0021 (and the timer device T.0003) is turned OFF 1 second after X0000 came ON.



Note:
- Time is set in 10 ms units for: T000 to T063 (0 to 327.67 s)
- Time is set in 100 ms units for: T064 to T255 (0 to 3276.7 s)
- Multiple timer instructions (TON, TOF or SS) with the same timer register are not allowed.

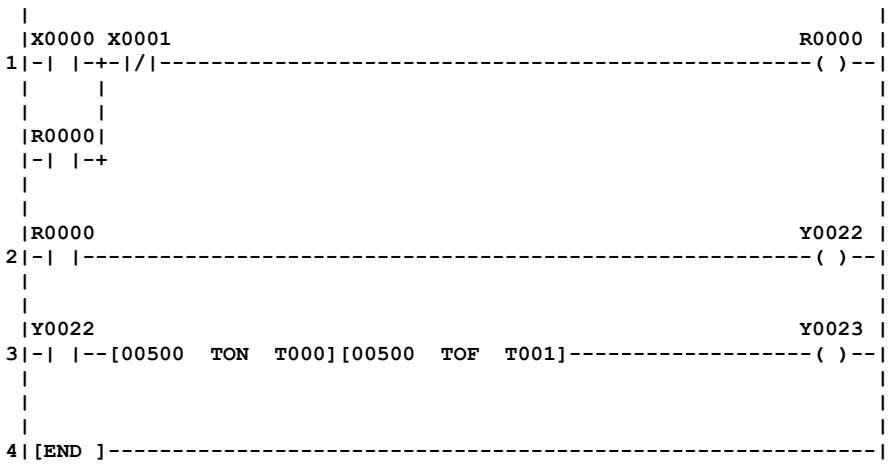
Sample Program: Start /Stop Pumps Sequentially

Description: 2 pumps need to be started one by one when the high level is reached in a certain tank and then stopped one by one when the Low level is reached in the tank.

Assignment:

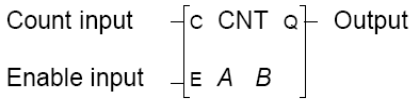
Signal	PLC address	State
High Level	X0	I/P
Low Level	X1	I/P
Pump 1	Y22	O/P
Pump 2	Y23	O/P

Ladder:



Function: **Counter**

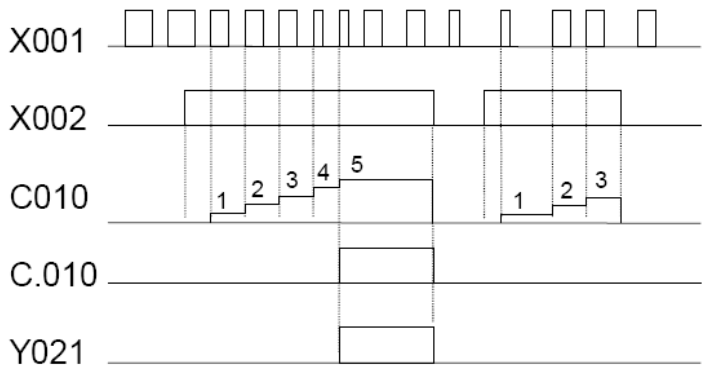
Expression:



Function Description: While enable input ON, it counts the number of the count input changes from OFF to ON. The count value is stored in B. When the count value reaches the set value A, the output and *B* are turned ON. When the enable input comes OFF, *B* is cleared to 0

Example:

```
|X0001                                                    Y0020 |
|-| |--C CNT      Q-----------------------------------------( )--|
|      |          |                                              |
|      |          |                                              |
|X0002 |          |                                              |
|-| |--E00005 C010|                                              |
```



Note:

- No transitional contact is required for the count input. The count input rising edge is detected by this instruction.
- For the count input, direct linking to a connecting point is not allowed. In this case, insert a dummy contact (always ON= S04F, etc.) just before the input. Refer to Note of Shift register FUN 074.
- Multiple counter instructions (CNT) with

Function: **Master control set/reset**

Expression:

```
-----[ MCS ]----|
|----[ MCR ]----|
```

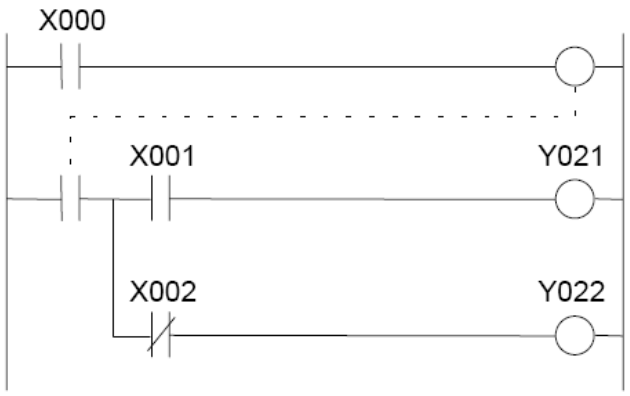
Function Description: When the MCS input is ON, ordinary operation is performed. When the MCS input is OFF, the state of left power rail between MCS and MCR is turned OFF.

Example:

```
  X0000                                                          
------I  I-------------------------------------------------------[ MCS ]-------
  X0001                                                          Y0021
------I  I-------------------------------------------------------(    )---------

  X0002                                                          Y0022
------I/I- -------------------------------------------------------(    )---------
 ---------------------------------------------------------------------[ MCR ]------
```

Equivalent Circuit:



Note: MCS and MCR must be used as a pair.
Nesting is not allowed.

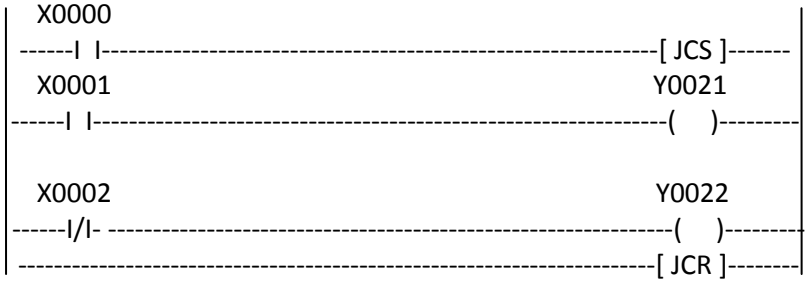
Function: **Jump control set/reset**

Expression:

```
-----[ JCS ]----|
|----[ JCR ]----|
```

Function Description: When the JCS input is ON, instructions between JCS and JCR are skipped (not executed). When the JCS input is OFF, ordinary operation is performed.

Example:



When X000 is ON, the rung 2 circuit is skipped, therefore Y021 is not changed its state regardless of the X001 state. When X000 is OFF, Y021 is controlled by the X001 state.

Note: JCS and JCR must be used as a pair.
Nesting is not allowed.

Function:	End
Expression:	|----[END]----|
Function Description:	Indicates the end of main program or subprogram. Instructions after the END instruction are not executed.
Example:	**|**----[END]----------**|**
Note:	Instructions after END instruction are not executed. Those steps are, however, counted as used steps.

Data Transfer Instructions

Function: Move

Expression: |---[A MOV B]---|

Function Description: When the input is ON, the data of *A* is stored in B.

Example:

```
    R0010
 |----I  I-------[12345    MOV    D0100 ]-----------------------------------------------|
```

When R0010 is ON, a constant data (12345) is stored in D0100 and the output is turned ON.

```
    X0005
 |----I  I-------[SW030   MOV   RW045 ]-----------------------------------------------|
```

When X0005 is ON, the data of SW030 is stored in RW045 and the output is turned ON. If SW030 is 500, the data 500 is stored in RW45**.**

Function: DMove

Expression: --[A+1.A DMOV B+B.1]--

Function Description: When the input is ON, the double-word (32-bit) data of (*A+*1.*A*) is stored in double-word register (*B+*1.B).
The data range is -2147483648 to 2147483647.

Example:

```
    R0011
|----I  I-------[D0101.D0100  DMOV  RW017.RW016 ]-------------------------|
```

When R0011 is ON, a double-word data of D0101.D0100 is stored in RW017.RW016 and the output is turned ON. If D0101.D0100 is 1234567, the data 1234567 is stored in RW017.RW016.

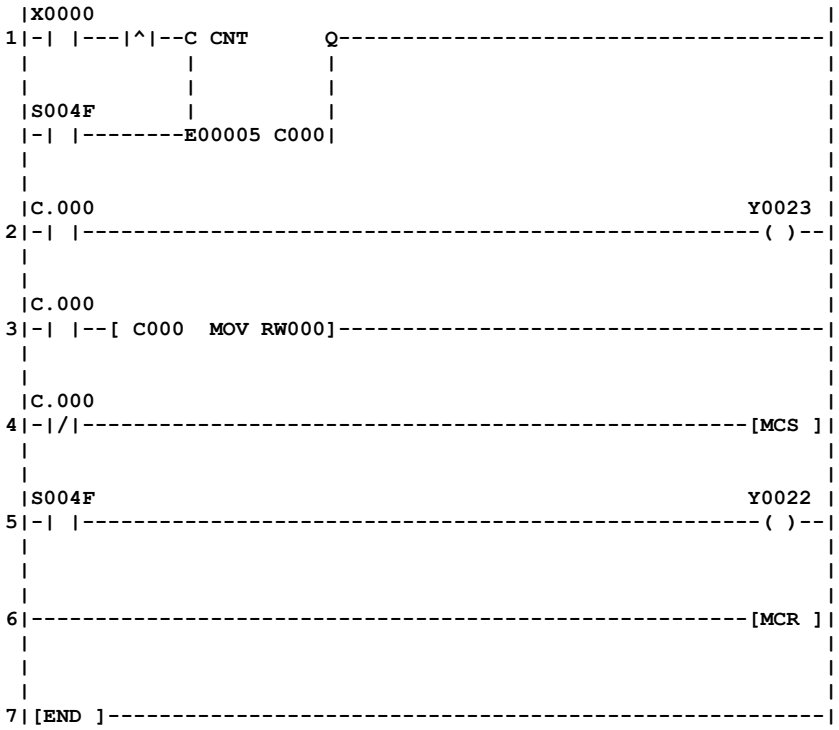
Sample Program: Parking Lot (Garage) Application

Description: Simple application of counting the cars entering the garage and When it reaches full capacity it gives indication of full capacity. Also there is an indication of empty space inside the garage.

Assignment:

Signal	PLC address	State
Car entered	X0	I/P
Empty space	Y22	O/P
Full capacity	Y23	O/P

Ladder:



Arithmetic Instructions

Function: Addition(+)

Expression: In –[A + B → C]--Out

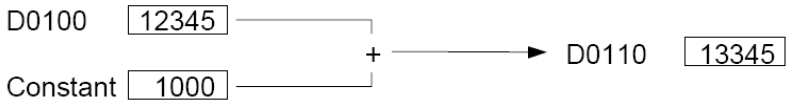
Function Description: When the input is ON, the data of *A* and the data of *B* are added, and the result is stored in C.

Example:

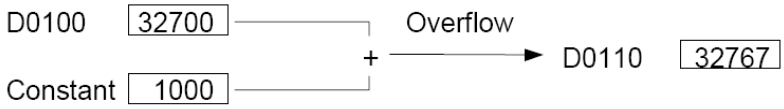
```
   R0005                                                                R0010
|----I  I---------[ D0100 + 01000  →  D0110 ]-------------------------(    )----|
```

When R0005 is ON, the data of D0100 and the constant data 1000 is added, and the result is stored in D0110.

If the data of D0100 is 12345, the result 13345 is stored in D0110, and R010 is turned OFF.



If the data of D0100 is 32700, the result exceeds the limit value, therefore 32767 is stored inD0110, and R010 is turned ON.



Function: **Subtraction(-)**

Expression: In –[A - B → C]--Out

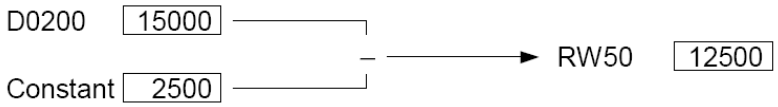
Function Description: When the input is ON, the data of *B* is subtracted from the data of A, and the result is stored in C.

Example:

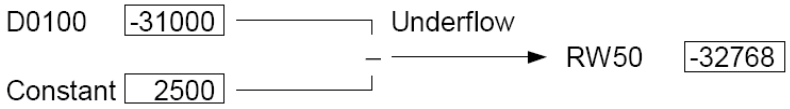
```
   R0005                                                                R0010
|----I  I---------[ D0200 - 02500  →  RW050 ]----------------------------(    )------|
```

When R0005 is ON, the constant data 2500 is subtracted from the data of D0200, and the result is stored in RW050.

If the data of D0200 is 15000, the result 12500 is stored in RW50, and R010 is turned OFF.



If the data of D0100 is 32700, the result exceeds the limit value, therefore 32767 is stored in D0110, and R010 is turned ON.



Function: **Multiplication (X)**

Expression: In -[A*B → C+1.C]-Out

Function Description: When the input is ON, the data of A is multiplied by the data of B, and the result is stored in double-length register C+1.C

Example:

```
   R0005
|----I  I--------[ D0050 * RW005  →  D0101.D100 ]-------------------------------|
```

When R0005 is ON, the data of D0050 is multiplied by the data of RW005, and the result is stored in double-length register D0101.D0100 (upper 16-bit in D0101 and lower 16-bit in D0100).

If the data of D0050 is 1500 and the data of RW05 is 20, the result 30000 is stored in D0101· D0100.

Function: **Division (/)**

Expression: In -[A / B → C]-Out

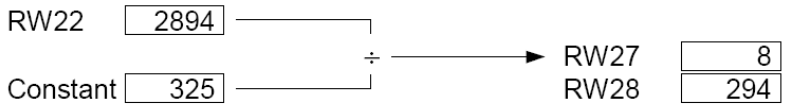
Function Description: When the input is ON, the data of *A* is divided by the data of B, and the quotient is stored in *C* and the remainder in *C+1.*

Example:

```
   R0005
|----I  I--------[ RW022  /  00325  →  RW027 ]-----------------------------------|
```

When R0005 is ON, the data of RW022 is divided by the constant data 00325, and the quotient is stored in RW027 and the remainder is stored in RW028.

If the data of RW22 is 2894, the quotient 8 is stored in RW27 and the remainder 294 is stored in RW28.



Note:
- If divisor (operand B) is 0, ERF (instruction error flag = S051) is set to ON. The ERF (S051) can be reset to OFF by user program, e.g. [RST S051].
- If the index register K is used as operand C, the remainder is ignored.
- If operand A is -32768 and operand B is -1, the data -32768 is stored in C and 0 is stored in C+1.

Function: **Double-Word Addition(D+)**

Expression: In –[A+1.A D+ B+1.B → C+1.C]--Out

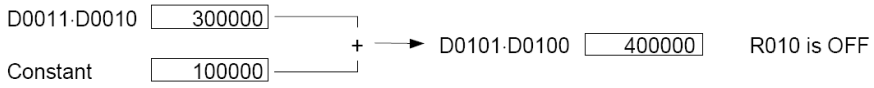
Function Description: When the input is ON, the double-word data of *A+1·A* and *B+1·B* are added, and the result is stored in *C+1·C*. The data range is -2147483648 to 2147483647.
If the result is greater than 2147483647, the upper limit value 2147483647 is stored in *C+1·C*, and
the output is turned ON.
If the result is smaller than -2147483648, the lower limit value
-2147483648 is stored in *C+1·C*, and the output is turned ON.

Example:

```
   R0005                                                          R0010
|----I  I-----[ D0011.D0010 + 0000100000 → D0101.D0100 ]------(    )----|
```

When R005 is ON, the data of D0011.D0010 and the constant data 100000 is added, and the result is stored in D0101·D0100.

If the data of D0011·D0010 is 300000, the result 400000 is stored in D0101·D0100, and R010 is turned OFF. (No overflow/underflow)



Function: **Double-Word Subtraction(D-)**

Expression: In –[A+1.A D- B+1.B → C+1.C]--Out

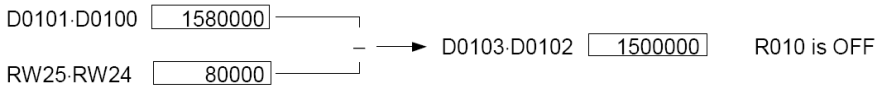
Function Description: When the input is ON, the double-word data of *B+1·B* is subtracted from *A+1·A*, and the result is stored in *C+1·C*. The data range is -2147483648 to 2147483647.
If the result is greater than 2147483647, the upper limit value 2147483647 is stored in *C+1·C*, and the output is turned ON.
If the result is smaller than -2147483648, the lower limit value -2147483648 is stored in *C+1·C*, and the output is turned ON.

Example:

```
   R0005                                                          R0010
|----I  I-----[ D0101.D0100 + RW025.RW024 → D0103.D0102 ]----(    )----|
```

When R005 is ON, the double-word data of RW25·RW24 is subtracted from the double-word data of D0101·D0100, and the result is stored in D0103·D0102.

If the data of D0101·D0100 is 1580000 and the data of RW25·RW24 is 80000, the result 1500000 is stored in D0103·D0102, and R010 is turned OFF. (No overflow/underflow)



Function: **Unsigned Multiplication**

Expression: In –[A U* B → C+1.C]--Out

Function Description: When the input is ON, the unsigned data of *A* and *B* are multiplied, and the result is stored in double-length register *C+1·C*. The data range of *A* and *B* is 0 to 65535 (unsigned 16-bit data)

Example:

```
   R0010                                                                R0010
|----I  I-----[ D0050  U*  RW005  →  D0101.D0100 ]--------------------(    )----|
```

When R010 is ON, the data of D0050 is multiplied by the data of RW05, and the result is stored in double-length register D0101·D0100 (upper 16-bit in D0101 and lower 16-bit in D0100).

If the data of D0050 is 52500 and the data of RW05 is 30, the result 1575000 is stored in D0101·D0100.

Note: This instruction handles the register data as unsigned integer.

Function: **Unsigned Multiplication**

Expression: In –[A U/ B → C+1.C]--Out

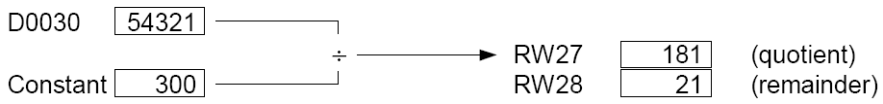
Function Description: When the input is ON, the unsigned data of *A* is divided by the unsigned data of *B*, and the quotient is stored in *C* and the remainder in *C+1.*
The data range of *A* and *B* is 0 to 65535 (unsigned 16-bit data)

Example:

```
   R0010                                                                R0010
|----I I-----[ D0030 U/ 00300 → D0050 ]--------------------------------(    )----|
```

When R010 is ON, the data of D0030 is divided by the constant data 300, and the quotient is stored in D0050 and the remainder is stored in D0051.

If the data of D0030 is 54321, the quotient 181 is stored in D0050 and the remainder 21 is stored in D0051.



Note:
- If divisor (operand *B*) is 0, ERF (instruction error flag = S051) is set to ON. The ERF (S051) can be reset to OFF by user program, e.g. –[RST S051]–.
- If the index register K is used as operand C, the remainder is ignored.
- This instruction handles the register data as unsigned integer.

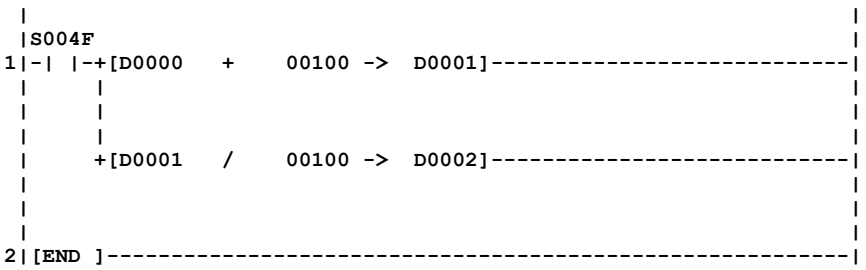
Sample Program: Calculation and Conversion Application

Description: Any calculation can be done easily inside the PLC using addition, Subtraction, multiplication and division as follows:
Creating a linear equation for the following points (0,1),(100, 2), The equation will be: Y = (X+100)/100.

Assignment:

Eq. Unknowns	PLC address	State
X	D0	Data Register
Y	D2	Data Register

Ladder:



Function: Increment +1

Expression: In --[+ 1 A]--Out

Function Description: When the input is ON, the data of A is increased by 1 and stored in A.

Example:

```
   X0004
|----I  I------- I↑I ----[ +1  D0050 ]----------------------------------------------------|
```

At the rising edge of X0004 changes from OFF to ON, the data of D0050 is increased by 1 and stored in D0050.

If the data of D0050 is 750 before the execution, it will be 751 after the execution.

Function: Decrement -1

Expression: In --[-1 A]--Out

Function Description: When the input is ON, the data of A is decreased by 1 and stored in A.

Example:

```
   X0005
|----I  I------- I↑I ----[ -1  D0050 ]----------------------------------------------------|
```

At the rising edge of X0004 changes from OFF to ON, the data of D0050 is decreased by 1 and stored in D0050.

If the data of D0050 is 1022 before the execution, it will be 1021 after the execution.

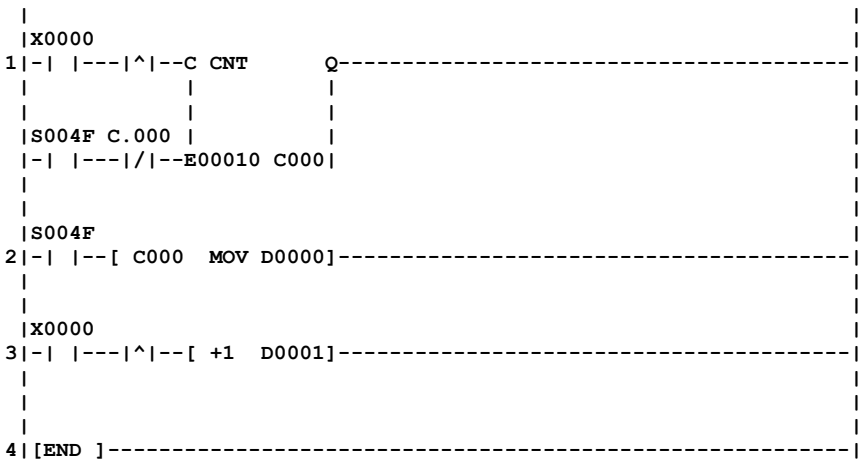
Sample Program: Counter and Totalizer Application

Description: Application to show how to count a product produced by a machine by using a counter that can be reset after reaching the set value and also counting this product using totalizer to count the whole number of Products produced since the startup of this machine without being reset.

Assignment:

Eq. Unknowns	PLC address	State
Counting sensor	X0	I/P
Counter value	D0	Data Register
Totalizer Value	D1	Data Register

Ladder:



Logical Operation Instructions

Function: And

Expression: In --[A AND B → C]--Out

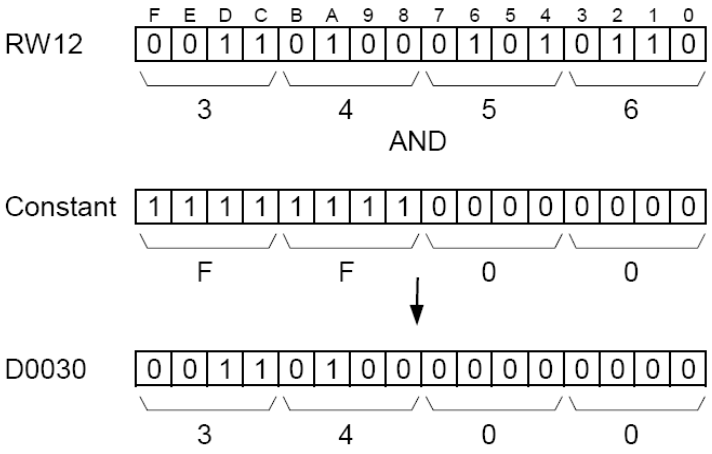
Function Description: When the input is ON, this instruction finds logical AND of A and B, and stores the result in C.

Example:

```
    R0012
|----I  I----------[ RW012  AND  HFF00  → D0030 ]---------------------------------|
```

When R0012 is ON, logical AND operation is executed for the data of RW012 and the constant data HFF00, and the result is stored in D0030.

If the data of RW12 is H3456, the result H3400 is stored in D0030.



Function: Or

Expression: In --[A OR B → C]-- Out

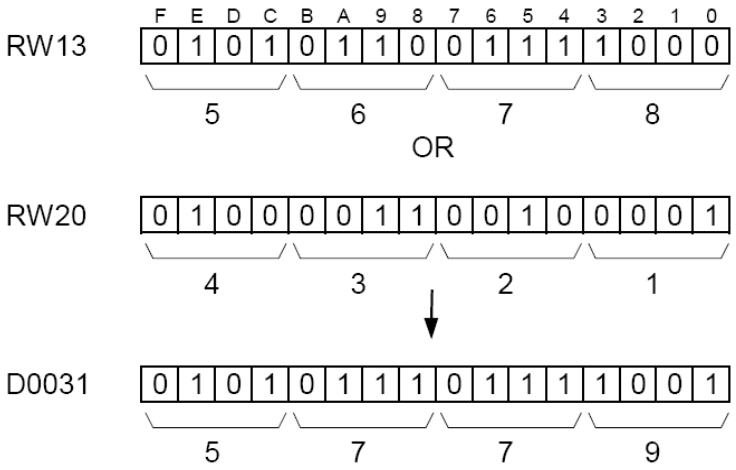
Function Description: When the input is ON, this instruction finds logical OR of A and B, and stores the result in C.

Example:

```
   R0012
|----I  I----------[ RW013  OR  RW020  → D0031 ]---------------------------------|
```

When R0012 is ON, logical OR operation is executed for the data of RW013 and RW020, and the result is stored in D0031.

If the data of RW13 is H5678 and RW20 is H4321, the result H5779 is stored in D0031.



Function: Exclusive OR

Expression: In --[A EOR B → C]-- Out

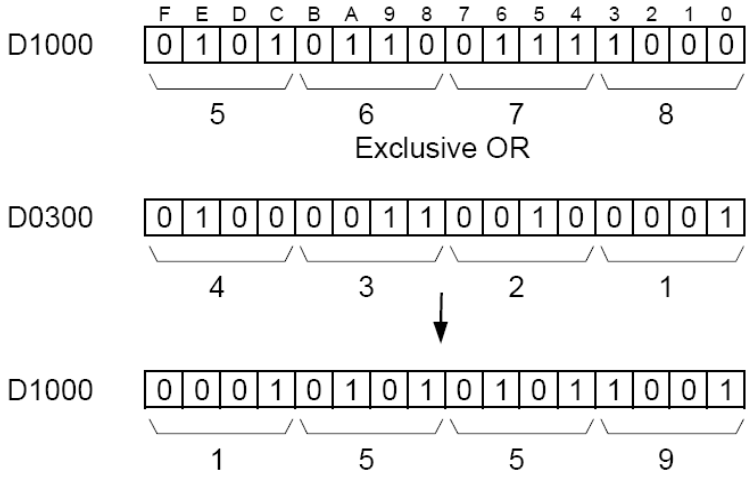
Function Description: When the input is ON, this instruction finds exclusive OR of *A* and B, and stores the result in C.

Example:

```
    R0012
|----I  I----------[ D1000  EOR  D0300  → D1000 ]---------------------------------|
```

When R0012 is ON, exclusive OR operation is executed for the data of D1000 and D0300, and the result is stored in D1000.

If the data of D1000 is H5678 and D0300 is H4321, the result H1559 is stored in D1000.



Special Processing Instructions

Function: Set

Expression: In --[SET A]-- Out

Function Description: When the input is ON, the device A is set to ON if A is a device, or the data HFFFF is stored in the register A if A is a register.

Example:

```
   R0010
|----I  I----------[ SET  R0025 ]-----------------------------------------------------------|
```

When R0010 is ON, R0025 is set to ON. The state of R0025 is remained even if R0010 comes OFF.

Function: Reset

Expression: In --[RST A]--Out

Function Description: When the input is ON, the device *A* is reset to OFF if *A* is a device, or the data 0 is stored in the register *A* if *A* is a register.

Example:

```
   R0011
|----I  I----------[ RST  R0005 ]-----------------------------------------------------|
```

When R0011 is ON, R0005 is reset to OFF. The state of R0005 is remained even if R0011 comes OFF.

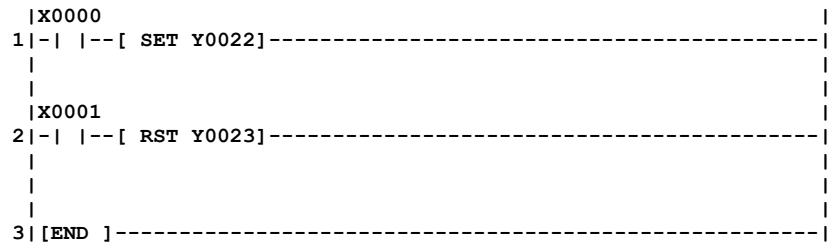
Sample Program: Motor Start / Stop

Description: A simple circuit with two push buttons as contacts and one output as a coil. As the first push button is pressed, the Output goes on and the motor starts to run. When the second push button is pressed, the output goes off and the motor stops, only this time Using set and reset instructions.

Assignment:

Signal	PLC address	State
Start	X0	I/P
Stop	X1	I/P
Motor	Y22	O/P

Ladder:



Function: Up-down counter

Expression:

Direction input —[U U/D Q]— Output
Count input —[C
Enable input —[E A]

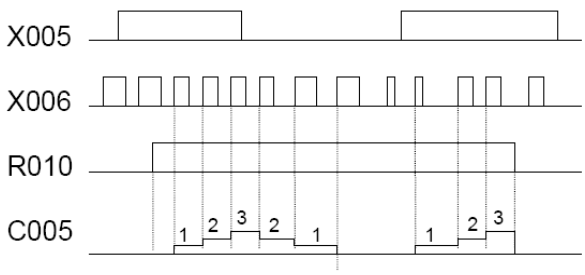
Function Description: While the enable input is ON, this instruction counts the number of the count input changes from OFF to ON. The count direction (up count or down count) is selected by the state of the direction input. The count value is stored in the counter register *A*.

- Up count when the direction input is ON
- Down count when the direction input is OFF

When the enable input is OFF, the counter register *A* is cleared to 0.

Example:

```
|X0005                                                          |
|-| |--------U U/D      Q-----------------------------------------|
|              |          |                                       |
|X0006         |          |                                       |
|-| |---|^|--C           |                                       |
|              |          |                                       |
|R0010         |          |                                       |
|-| |--------E      C005|                                       |
```



Compare Instructions

Function: Greater than

Expression: In --[A > B]--Out

Function Description: When the input is ON, the data of *A* and the data of *B* are compared, and if *A* is greater than *B*, the output is turned ON.

Example:

```
   R000C                                                                R0020
|----I  I----------[ D0125  >  02500 ]-----------------------------------------(    )------|
```

When R000C is ON, the data of D0125 is compared with the constant data 2500, and if the data of D0125 is greater than 2500, R0020 is turned ON.

If the data of D0125 is 3000, the comparison result is true. Consequently, R020 is turned ON.

Other compare instructions:

Expression	Function Description
I/P –[A >= B]-- O/P	Greater than or Equal
I/P –[A = B]-- O/P	Equal
I/P –[A <> B]-- O/P	Not Equal
I/P –[A < B]-- O/P	Less than
I/P –[A <= B]-- O/P	Less than or Equal

Function: Double-word greater than

Expression: I/P –[A+1.A D> B+1.B]-- O/P

Function Description: When the input is ON, the double-word data of *A+1·A* and *B+1·B* are compared, and if *A+1·A* is greater than *B+1·B*, the output is turned ON.

Example:

```
   R0010                                                              R0014
|----I  I----------[ D0101.D0100  >  0000200000 ]----------------------(    )------|
```

When R010 is ON, the data of D0101·D0100 is compared with the constant data 200000, and if the data of D0101·D0100 is greater than 200000, R014 is turned ON.

If the data of D0101·D0100 is 250000, the comparison result is true. Consequently, R014 is turned ON.

D0101·D0100 | 250000 | > | Constant | 200000

R014 is ON

Note: This instruction deals with the data as double-word integer (-2147483648 to 2147483647).

Other compare instructions:

Expression	Function Description
I/P–[A D>= B]—O/P	D. Word Greater than or equal
I/P –[A D= B]-- O/P	Double word Equal
I/P –[A D<> B]-- O/P	Double Not Equal
I/P –[A D< B]-- O/P	Double Word Less than
I/P –[A D<= B]-- O/P	D. Word Less than or Equal

Function: Unsigned greater than

Expression: Input –[*A* U> *B*]– Output

Function Description: When the input is ON, the data of *A* and the data of *B* are compared, and if *A* is greater than *B*, the output is turned ON.

Example:

```
   R000C                                                                R0020
|----I  I----------[ D125   U>  40000 ]-------------------------------------(    )------|
```

When R00C is ON, the data of D0125 is compared with the constant data 40000, and if the data of D0125 is greater than 40000, R020 is turned ON.

If the data of D0125 is 52000, the comparison result is true. Consequently, R020 is turned ON.

D0125 [52000] > Constant [40000] ⟶ R020 is ON

Note: This instruction deals with the data as unsigned integer (0 to 65535).

Expression	Function Description
I/P–[A U>= B]—O/P	Unsigned Greater than or Equal
I/P –[A U= B]-- O/P	Unsigned Equal
I/P –[A U<> B]-- O/P	Unsigned Not Equal
I/P –[A U< B]-- O/P	Unsigned Less than
I/P –[A U<= B]-- O/P	Unsigned Less than or Equal

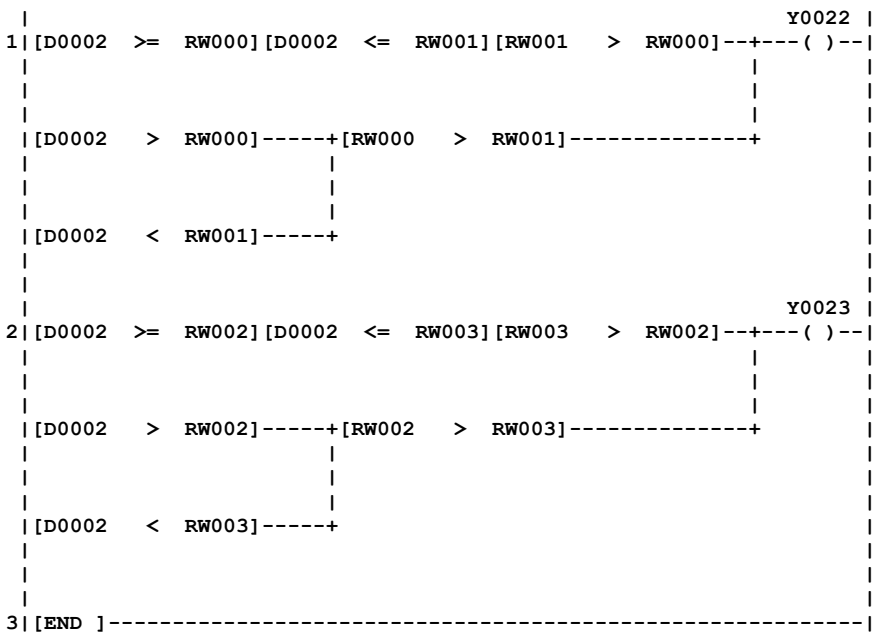
Sample Program: Ice Cream machine using encoder

Description: A simple Application of an ice cream machine using an encoder where we read the encoder pulses and by comparing this reading to the start and end angle of every station, if the reading is within range the station is turned on.

Assignment:

Signal	PLC address	State
Encoder Reading	D2	Data Register
Start of Station 1	RW0	Auxiliary Register
End of Station 1	RW1	Auxiliary Register
Start of Station 2	RW2	Auxiliary Register
End of Station 2	RW3	Auxiliary Register
Station 1	Y22	O/P
Station 2	Y23	O/P

Ladder:



Special Purpose Functions

Function: Function Generator (Analog Signals Processing)

Expression: In --[A FG(n) B → C]-- Out

Function Description: When the input is ON, this instruction finds the function value *f(x)* for *A* as *x*, and stores it in *C*. The function *f(x)* is defined by the parameters stored in 2 × *n* registers starting with *B*.

Example:

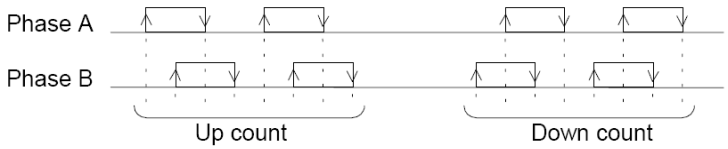
```
 1|-|^|-+[ 00000 MOV D0100][ 02000 MOV D0101]----------------------|
  |     |                                                          |
  |     |                                                          |
  |     |                                                          |
  |     +[ 00000 MOV D0102][ 00100 MOV D0103]----------------------|
  |                                                                |
  |                                                                |
  |R0010                                                           |
 2|-| |--[XW004   FG  (02)  D0100  ->  D0104]----------------------|
```

When R010 is ON, the FG instruction finds the function value *f(x)* for *x* = XW004, and stores the result in D0104. The function *f(x)* is defined by 2 × 2 = 4 parameters stored in D0100 to D0103. In this example, these parameters are set at the first scan.

Note: This function can be used to read various types of physical quantities (such as: Temperature, Pressure, Humidity,….,etc.) by taking the analog signal (Voltage or Current) measured by special purpose sensors and convert it into an actual reading to be used in the sequence of the program or even to be displayed on HMI or SCADA systems.

Function: High Speed Counter (Encoder Pulses- Quadrature bi pulse mode)

Function Description: This function counts up or down the quadrature bi-pulse (2-phase pulses whose phases are shifted 90° each other). Counts up when phase A precedes, and counts down when phase B precedes. Both rising and falling edges of each phase are counted. Consequently, 4 times count value against the pulse frequency is obtained.



When the count value reaches the comparison value 1 (or 2), the I/O interrupt program #1 (or #2) is activated (if the interrupt enable flag for each is ON). This counter operation is enabled while the soft-gate is ON. The count value is reset to 0 when the soft-gate is changed from ON to OFF.

Related Register :

Function	Register/device
Phase A	X000
Phase B	X001
Reset input	X002
Comparison value 1	SW19.SW18
Comparison value 2	SW21.SW20
Count value	SW23.SW22
Soft gate	S240
Interrupt enable 1	S241
Interrupt enable 2	S249

Example:

```
1|-|^|--[ 02051 MOV SW016]-----------------------------------------|
 |                                                                  |
 |R0010                                                             |
2|-| |---|^|-+[ 0000000400 DMOV SW019·SW018]-------------------- --|
 |           |                                                      |
 |           +[ 0000000500 DMOV SW021·SW020]-------------------- --|
 |           |                                                      |
 |           |                                                S0240 |
 |           +-------------------------------------------------( )--|
 |           |                                                      |
 |           |                                                S0241 |
 |           +-------------------------------------------------( )--|
 |           |                                                      |
 |           |                                                S0249 |
 |           +-------------------------------------------------( )--|
 |                                                                  |
 |                                                                  |
3|-|^|--[SW022   /    00004 ->  D0100]-----------------------------|
```

In this example, 2051 (H0803) is set in SW16. As a result, the quadrature bi-pulse counter is selected.

When R010 comes ON, the data 150000 is set into the comparison value 1 register(SW19·SW18), and 200000 is set into the comparison value 2 register (SW21·SW20). While R010 is ON, the soft-gate (S240), the interrupt enable flag 1 (S241) and the interrupt enable flag 2 (S249) are set to ON to enable the counter operation.

The count value is stored in SW23·SW22.

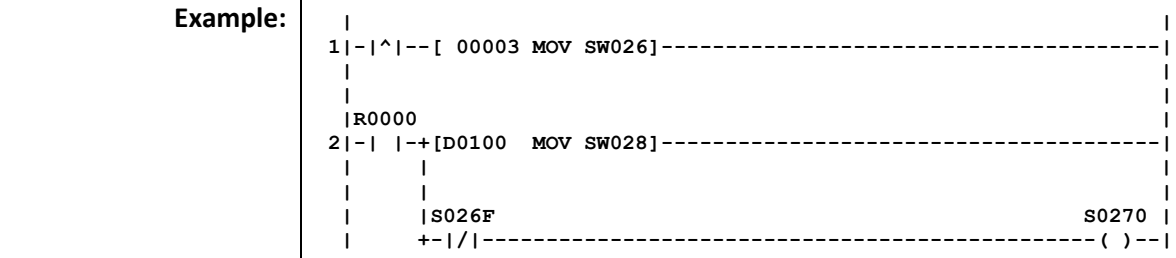
Function: Pulse Output Function

Function Description: This function is used to output a variable frequency pulse train. The controllable pulse frequency is 50 to 5000 Hz (1 Hz units). The output mode can be selected to be CW/CCW. CW pulse is output when the frequency setting is positive (50 to 5000), and CCW pulse is output when it is negative (-50 to -5000).

Related Registers:

Function	Register/Device
CW Pulse	Y20
CCW Pulse	Y21
Pulse Enable Flag	S270
Frequency Setting Register	SW28
Frequency Setting Error Flag	S26F

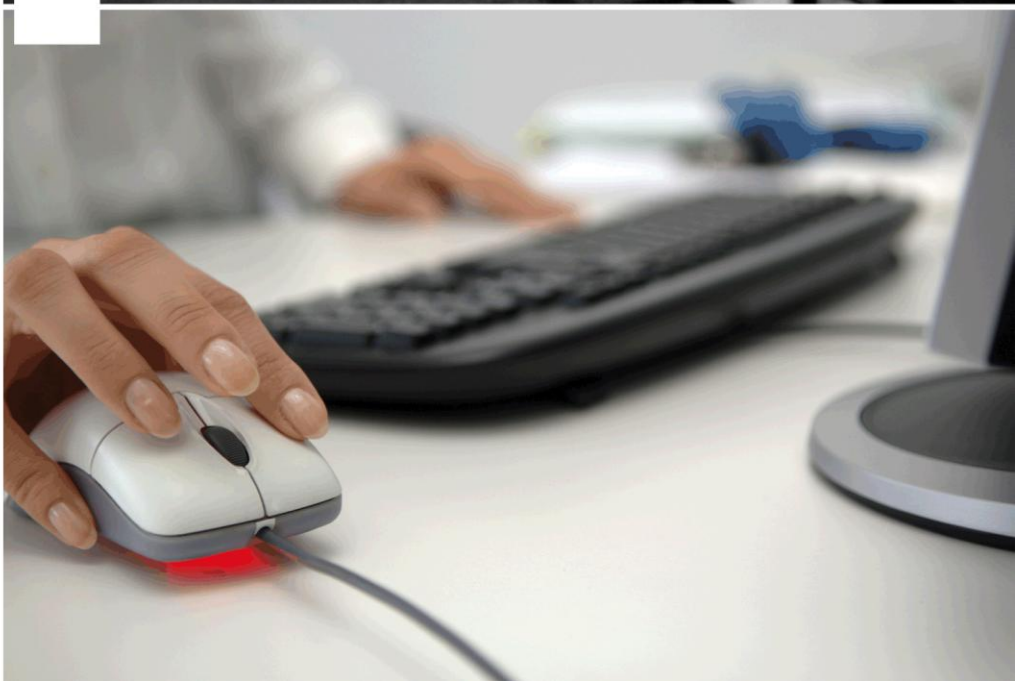
Example:



In this example, 3 (H0003) is set in SW26. As a result, the CW/CCW mode pulse output function is selected. When R000 is ON, the pulse output is started with the frequency designated by D0100.
If an invalid frequency is designated, the frequency setting error flag (S26F) comes ON and the pulse enable flag (S270) is turned OFF. Then the pulse output is stopped.

Notes:

8. Applications and Exercises



Open a new file under T-PDS software:

1. Run the T-PDS program.
2. Go to the (File) menu at top of the screen and choose (new Project).
3. Select T1S from the list and press OK.
4. The Ladder edit page will appear.
5. Go to the (PLC) menu at the top of the screen.
6. Choose (I/O Allocation).
7. From the allocation list in front of you choose (00-00) and press OK.
8. At the (module type and description) choose (main slot 0 only).
9. At the (top Register no:) write (0).
10. Press OK.
11. Go to (Edit) menu and choose (Edit mode).
12. Choose (Ladder) and press OK.
13. Now you are able to write your program.
14. Enjoy writing your program and navigate the software.

Download your ladder file to the PLC:

1. Write your program.
2. Save the program.
3. Go to the (File) menu at the top of the screen and choose (Transfer program).
4. Choose (File -> PLC) from the (Transfer program) menu.
5. Choose your file and double click on it.
6. Make sure that, the button at the PLC is in HALT position not in the RUN position.
7. The file transferring to the PLC.
8. Go to (PLC) menu and choose (Memory Management).
9. Choose (Write EEPROM/IC Card).
10. Change the button at the PLC to RUN position.
11. Go to the (PLC) menu at the top of the screen and chose (Online/Offline)

APPLICATION (1): Alarm System

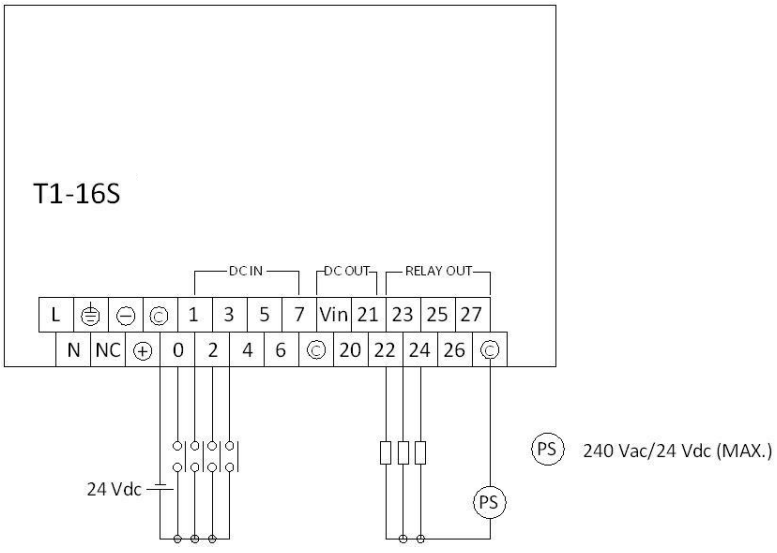
Description: There are four hazard inputs to the alarm system that go on when some operational malfunction occurs. We do not define what the hazards are, for PLC operation illustration, we only use the fact that there are four, the system operate as the following:

- If one input is on, nothing happens.
- If any two inputs are on, a red pilot light goes on.
- If three inputs are on, an alarm siren sounds.
- If all four are on, the fire department is notified

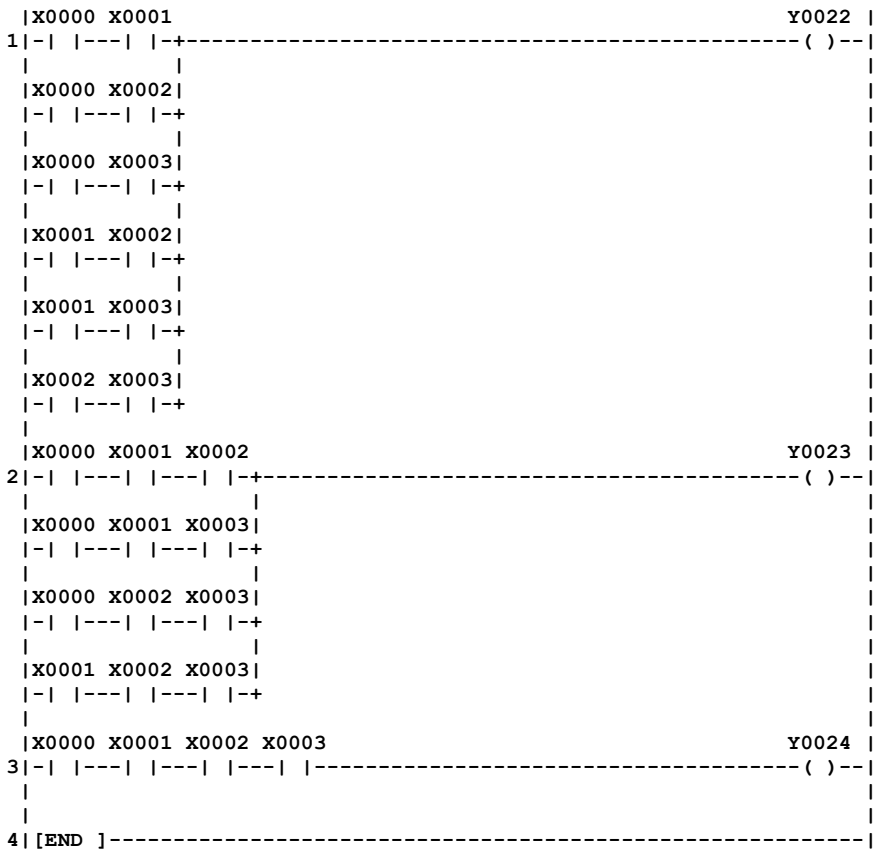
Assignment:

Signal	PLC address	State
Alarm 1	X0000	Input
Alarm 2	X0001	Input
Alarm 3	X0002	Input
Alarm 4	X0003	Input
Pilot light	Y0022	Output
Siren sounds	Y0023	Output
Fire dept.	Y0024	Output

Hardware Wiring:



Ladder Diagram:



Application (2): Motor run forward and reverse application

Description: This is a forward and reverse circuit. Each direction's coil has its own start button and one stop button to stop any coil.

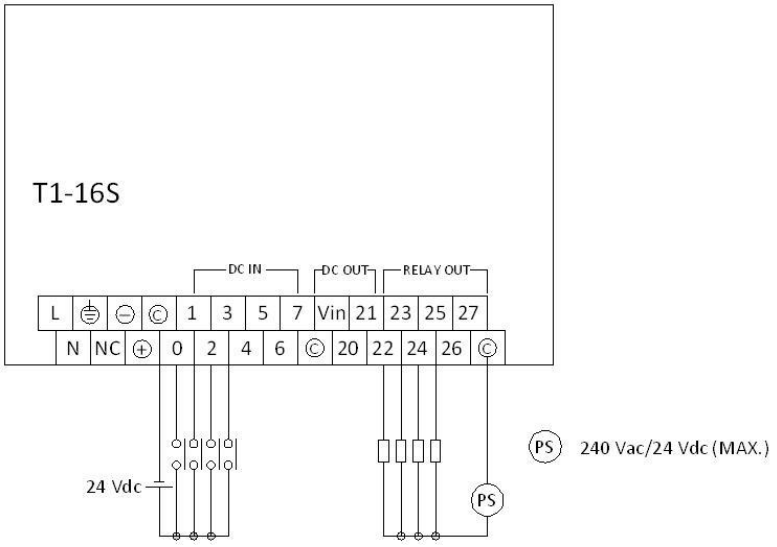
At this circuit, interlocks are provided so that both outputs cannot be energized at the same time.

Directional pilot light indicators are present; each direction has separate light to know the motor direction.

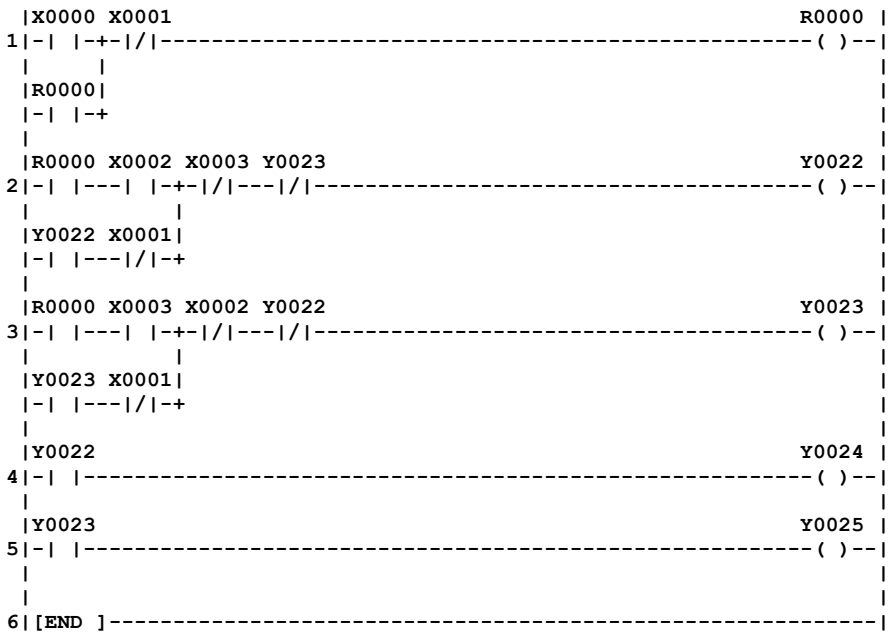
Assignment:

Signal	PLC address	State
Start switch	X0000	Input
Stop switch	X0001	Input
Forward switch	X0002	Input
Reverse switch	X0003	Input
Forward coil	Y0022	Output
Reverse coil	Y0023	Output
Red lamp	Y0024	Output
Green lamp	Y0025	Output
Run state	R0000	Marker

Hardware Wiring:



Ladder Diagram:



Application (3): Garage Door Controller

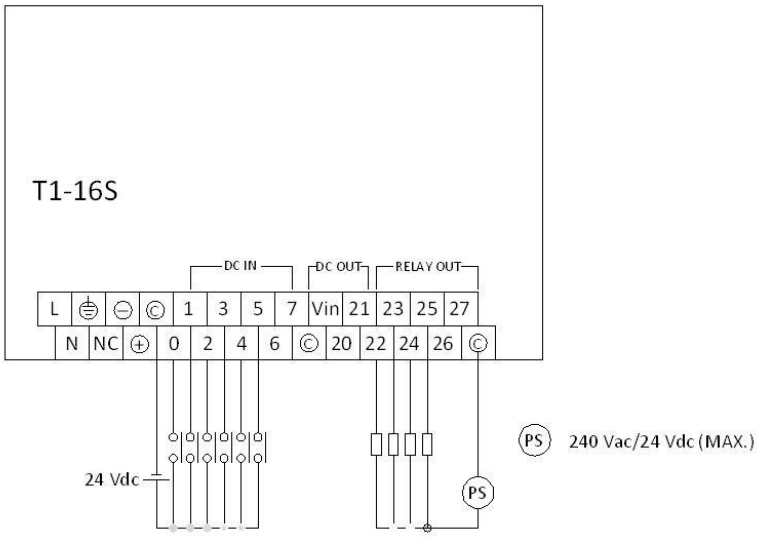
Description: The roller door of an underground garage for cars is to be opened electrically. When a push button is pressed, the PLC controls an electric motor which opens the roller door and closes it again after a specified time:

- The door should be opened from outside the garage using a key operated switch.
- The door also should be opened from inside using a push button.
- The door should close again automatically after a specified time.
- A warning lamp lights up during the closing process to prevent cars from driving into the closing door.
- A light barrier prevents the door from closing when a car is present in the entrance to the garage.
- Limit switches notify the PLC when the roller door is fully open or fully close.
- There is a switch to turn on the garage lighting independently of the roller door for a certain period of time

Assignment:

Signal	PLC address	State
Key switch	X0000	Input
Push Button	X0001	Input
L.S. (door open)	X0002	Input
L.S. (door close)	X0003	Input
Light Barrier	X0004	Input
P. Button (lighting)	X0005	Input
K1 (open door)	Y0022	Output
K2 (close door)	Y0023	Output
H1 (warning lamp)	Y0024	Output
H2 (garage light.)	Y0025	Output

Hardware Wiring:



Ladder Diagram:

```
  |X0000 X0002                                                    Y0022 |
1|-| |-+-|/|-----------------------------------------------------( )--|
  |     |                                                              |
  |     |                                                              |
  |X0001|                                                              |
  |-| |-+                                                            |
  |     |                                                              |
  |     |                                                              |
  |Y0022|                                                              |
  |-| |-+                                                            |
  |                                                                    |
  |                                                                    |
  |X0002                          X0003                          R0000 |
2|-| |--[12000  TON  T000]-----+-|/|-------------------------( )--|
  |                           |                                        |
  |                           |                                        |
  |R0000                      |                                        |
  |-| |-------------------------+                                      |
  |                                                                    |
  |                                                                    |
  |R0000 X0004                                                     Y0023 |
3|-| |---| |-+-----------------------------------------------( )--|
  |           |                                                        |
  |           |                                                        |
  |           |                                                Y0024 |
  |           +-----------------------------------------------( )--|
  |                                                                    |
  |                                                                    |
  |X0005                                                           Y0025 |
4|-| |---|^|--[30000  SS   T001]-----------------------------( )--|
  |                                                                    |
  |                                                                    |
  |                                                                    |
5|[END ]--------------------------------------------------------------|
```

Application (4): Belt Sequence Control

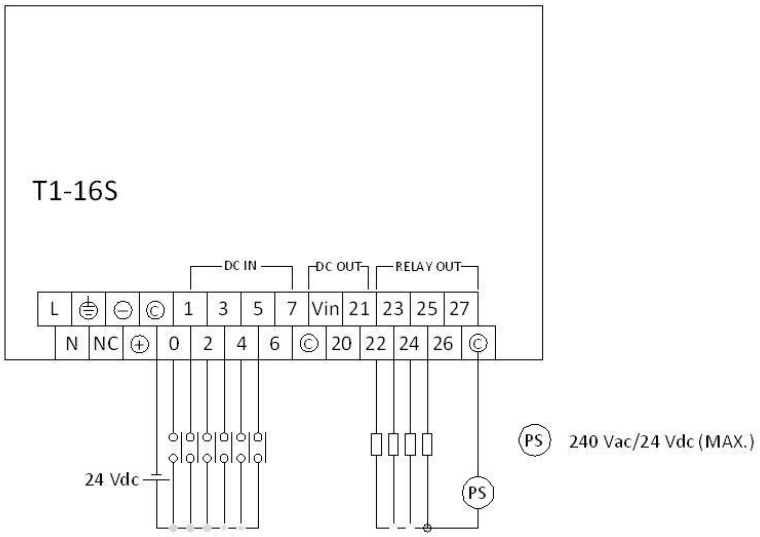
Description: It's required to start up and shut down three conveyor belts at diff. times. There are to be three operating modes "staggered start-up", "Staggered Shutdown" and "Fast Stop". The motor protective circuit breakers in the belt drives should be monitored; if a circuit breaker trips, the conveyor system should be stopped and the fault should be signaled by a flashing light.

- When start button is pressed, the belts start up at 5-second intervals. Belt 3 starts first
- When the stop button is pressed, the belts stop in reverse order, i.e. starting from belt 1. The belts also stops at 5 seconds intervals knowing that when the stop button is pressed, 5 seconds elapse before belt 1 stops.
- When fast stop button is pressed, all three belts stop without a time delay.
- If a motor fails, the fault is signaled via the flashing light and automatically stops all three belts

Assignment:

Signal	PLC address	State
Start	X0000	Input
Stop	X0001	Input
Fast Stop	X0002	Input
Protection Belt 1	X0003	Input
Protection Belt 2	X0004	Input
Protection Belt 3	X0005	Input
Belt 1	Y0022	Output
Belt 2	Y0023	Output
Belt 3	Y0024	Output
Indicating Light	Y0025	Output

Hardware Wiring:



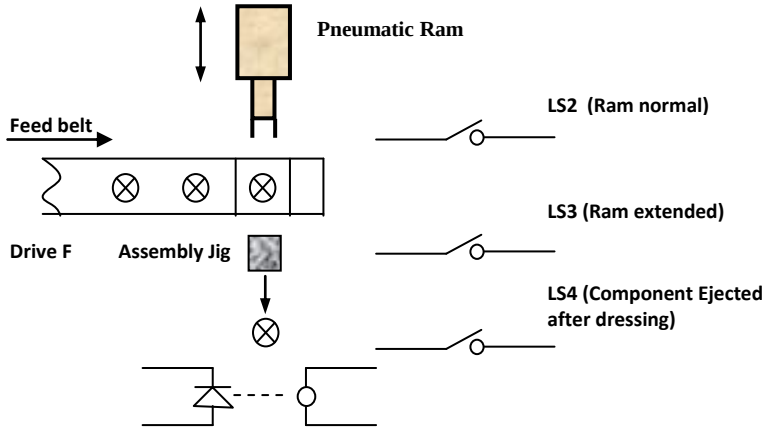
Ladder Diagram:

```
  |X0000 R0002 R0012                                                R0000 |
1|-| |-+-|/|---|/|-----------------------------------------------( )--|
  |     |                                                                |
  |R0000|                                                                |
  |-| |-+                                                               |
  |                                                                      |
  |                                                                      |
  |X0001 R0000 R0012                                                R0001 |
2|-| |-+-| |---|/|-----------------------------------------------( )--|
  |     |                                                                |
  |R0001|                                                                |
  |-| |-+                                                               |
  |                                                                      |
  |                                                                      |
  |R0000                    R0012                                   Y0024 |
3|-| |--[00500  TOF  T004]-|/|-------------------------------( )--|
  |                                                                      |
  |                                                                      |
  |Y0024 R0003                                        R0012      Y0023 |
4|-| |---|/|--[00500  TON  T000][00500  TOF  T003]-|/|-------( )--|
  |                                                                      |
  |                                                                      |
  |Y0023 R0001                                        R0012      Y0022 |
5|-| |---|/|--[00500  TON  T001][00500  TOF  T002]-|/|-------( )--|
  |                                                                      |
  |                                                                      |
  |Y0023                                                            R0002 |
6|-| |---|v|--[00500  SS   T006]-----------------------------( )--|
  |                                                                      |
  |Y0022                                                            R0003 |
7|-| |---|v|--[00500  SS   T005]-----------------------------( )--|
  |                                                                      |
  |                                                                      |
  |X0003                                                            R0012 |
8|-| |-+-----------------------------------------------------( )--|
  |     |                                                                |
  |X0004|                                                                |
  |-| |-+                                                               |
  |     |                                                                |
  |X0005|                                                                |
  |-| |-+                                                               |
  |                                                                      |
  |                                                                      |
  |R0012 S0044                                                      Y0025 |
9|-| |---| |-----------------------------------------------( )--|
  |                                                                      |
  |                                                                      |
  |                                                                      |
10|[END ]-------------------------------------------------------------|
```

Application (5): Sequence control of an assembly machine

Description: A feed belt moves components in succession towards the assembly machine, when a component is in position in front of the assembly Jig, limit switch LS1 operates to stop the feed belt. A pneumatic ram is then operated to push the component firmly into the assembly Jig. When LS2 is operated, this Stops the ram and moves it back into normal position.
The limit switch LS3 also signals that the ram is clear of the assembly Jig, and the machine now presses the raw component into the desired shape. The pressing operation is completed and the resulting artifact is ejected from the press, which activates a light sensor LS4, this indicates that the next component can be moved up, and then to the Jig. The process then continues.

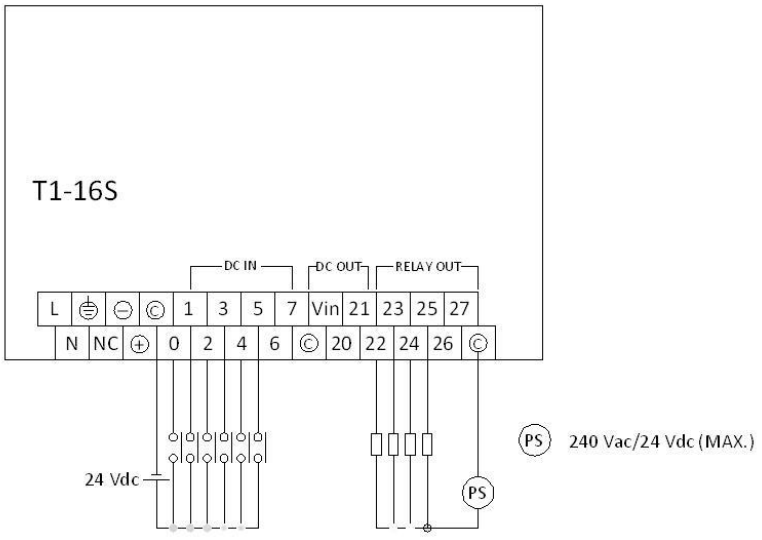
Drawing Overview:



Assignment:

Signal	PLC address	State
Start	X0000	Input
Stop	X0001	Input
LS1	X0002	Input
LS2	X0003	Input
LS3	X0004	Input
LS4	X0005	Input
-	R0000	Auxiliary Relay
-	R0001	Auxiliary Relay
-	R0002	Auxiliary Relay
-	R0003	Auxiliary Relay
-	R0004	Auxiliary Relay
Belt	Y0022	Output
Ram Out	Y0023	Output
Ram In	Y0024	Output
Press	Y0025	Output

Hardware Wiring:



Ladder Diagram:

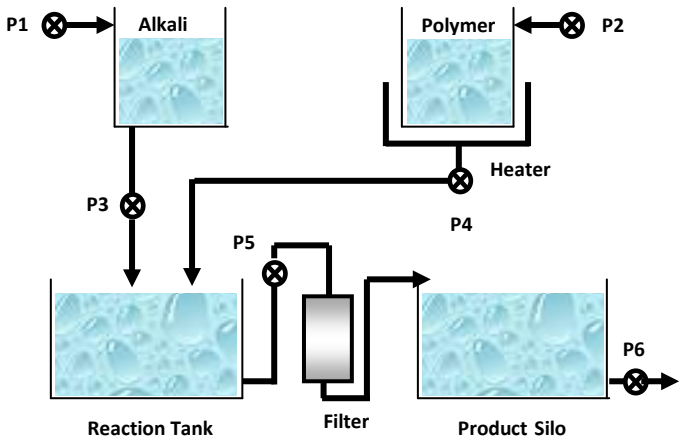
```
  |X0000 X0001 X0002                                          R0000 |
1|-| |-+-|/|---|/|-------------------------------------------( )--|
  |     |                                                          |
  |R0000|                                                          |
  |-| |-+                                                        |
  |     |                                                          |
  |X0005|                                                          |
  |-| |-+                                                        |
  |                                                              |
  |R0000                                                   Y0022 |
2|-| |-------------------------------------------------------( )--|
  |                                                              |
  |X0002 X0003                                             R0001 |
3|-| |-+-|/|-----------------------------------------------( )--|
  |     |                                                        |
  |     |                                                        |
  |R0001|                                                        |
  |-| |-+                                                      |
  |                                                            |
  |R0001                                                   Y0023 |
4|-| |-------------------------------------------------------( )--|
  |                                                              |
  |X0003 X0004                                             R0002 |
5|-| |-+-|/|-----------------------------------------------( )--|
  |     |                                                        |
  |R0002|                                                        |
  |-| |-+                                                      |
  |                                                            |
  |R0002                                                   Y0024 |
6|-| |-------------------------------------------------------( )--|
  |                                                              |
  |X0004 X0005                                             R0003 |
7|-| |-+-|/|-----------------------------------------------( )--|
  |     |                                                        |
  |R0003|                                                        |
  |-| |-+                                                      |
  |                                                            |
  |R0003                                                   Y0025 |
8|-| |-------------------------------------------------------( )--|
  |                                                              |
9|[END ]-------------------------------------------------------|
```

Application (6): Chemical process plant control

Description: The plant consists of four tanks with pumps to transfer the liquid contents through the system, each tank is fitted with sensors to detect "Empty" and "Full" states. Tank 2 has an integral heating element with associated temperature sensor. Tank 3 is equipped with a stirring arm to mix the two liquids when they are pumped from tank 1 and Tank 2, the lower tanks 3 and 4 have twice the capacity of Tanks 1 and 2, and will therefore be filled by the contents of tanks 1 and 2.

- Tanks 1 and 2 are filled from the supply by Alkali and Polymer respectively via pumps 1 and 2. Pumps 1 and 2 turns off when high level sensor indicates ON.
- The heating element in tank 2 is activated, increasing the polymer temperature up to 60 ^{O}C.
- Instead of using heating element we will heat Tank 2 for 120 seconds, after time is out the temperature will be about 60 ^{O}C, this should turn Off the heater and turn ON pumps 3 and 4 to transfer the liquids into the reaction vessel, Tank 3
- The stirring arm must also run when this tank is being used for 60 seconds. Once Tank 3 is full, Pumps 3 and 4 are stopped. If the stirring time is greater than 60 seconds, Pump 5 is working to transfer the mixture to Tank 4 (product silo) via a filter unit.
- Pump 5 is stopped once Tank 4 is full or Tank 3 is Empty. Finally the product is run off into storage using Pump 6.

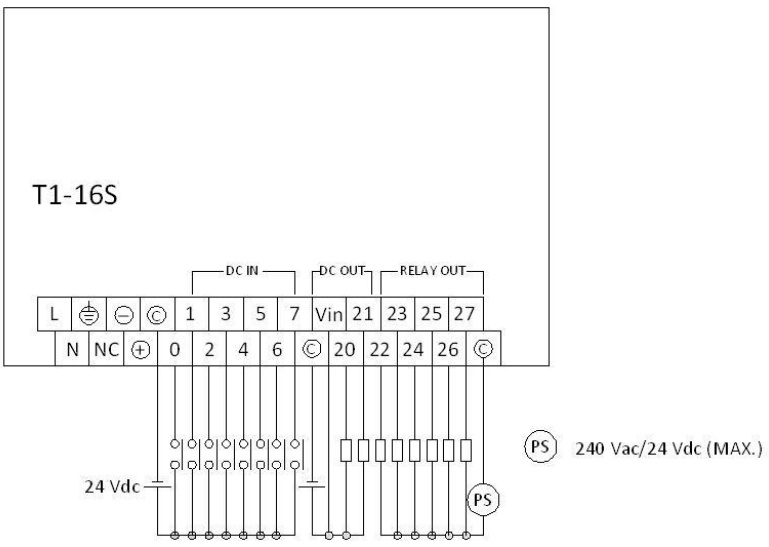
Drawing Overview:



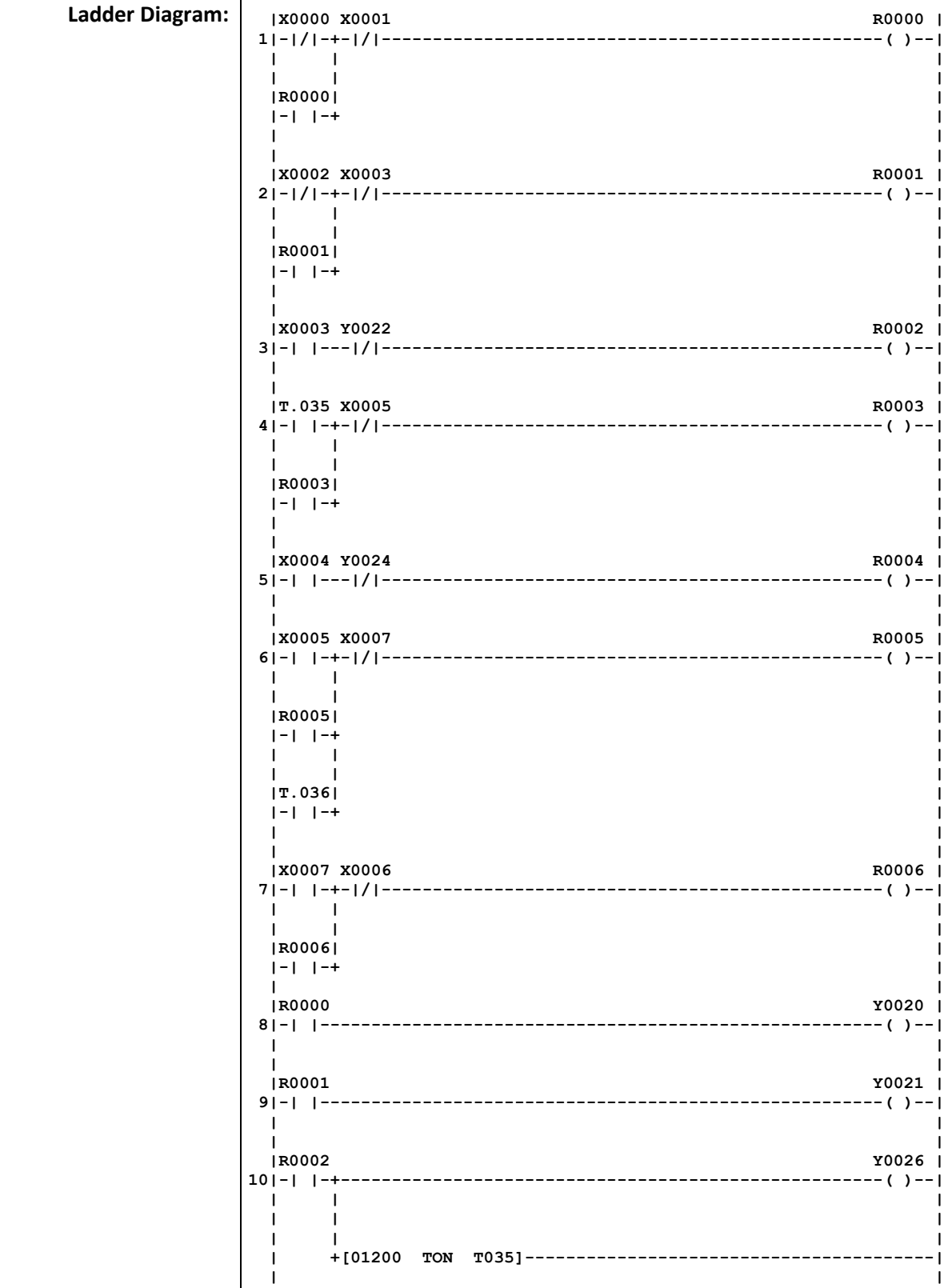
Assignment:

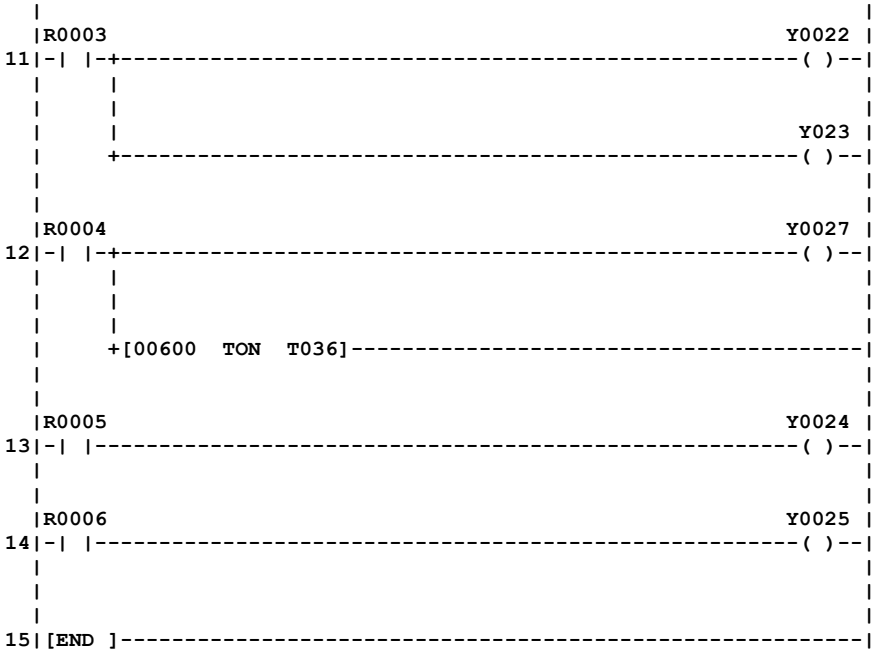
Signal	PLC address	State
Tank 1 Empty	X0000	Input
Tank 1 Full	X0001	Input
Tank 2 Empty	X0002	Input
Tank 2 Full	X0003	Input
Tank 3 Empty	X0004	Input
Tank 3 Full	X0005	Input
Tank 4 Empty	X0006	Input
Tank 4 Full	X0007	Input
-	T035	Internal timer
-	T036	Internal timer
-	R0001	Auxiliary Relay
-	R0002	Auxiliary Relay
-	R0003	Auxiliary Relay
-	R0004	Auxiliary Relay
-	R0005	Auxiliary Relay
-	R0006	Auxiliary Relay
-	R0007	Auxiliary Relay
-	R0008	Auxiliary Relay
Pump 1	Y0020	Output
Pump 2	Y0021	Output
Pump 3	Y0022	Output
Pump 4	Y0024	Output
Pump 5	Y0025	Output
Pump 6	Y0026	Output
Heat .element	Y0026	Output
Stirring Arm	Y0027	Output

Hardware Wiring:



Ladder Diagram:




R0003
Y0022
Y023
R0004
Y0027
[00600 TON T036]
R0005
Y0024
R0006
Y0025
[END]

Conclusion

Main steps in planning and constructing a large process Ladder diagram:

1. Define the process to be controlled.
2. Make a sketch of the process operation.
3. Create a written step sequence listing for the process.
4. Add sensors on the sketch as needed to carry out the control sequence.
5. Add manual controls as needed for process setup or operational checking.
6. Consider the safety of the operation personnel and make additions and adjustments as needed.
7. Add master stop switches as required for safe shutdown.
8. Create the ladder logic diagram that will be used as a basis for the PLC program.
9. Consider the “what if’s” where the process sequence may go astray.

Exercises

Construct PLC ladder diagrams for the problems listed. A sequence could be written first, if necessary. As an option, show the inputs and outputs modules along with the device to terminal electrical connections. In the laboratory, you may load the PLC CPU with your program, connect the PLC to the simulator and check the proper operation of the circuit by running the program sequence.

1. A fan is to be started and stopped form any one of three locations. Each location has a start and a stop button, note that the normally closed stops should be in series and normally open starts in parallel.
2. A two way hydraulic cylinder has two solenoids controlling it, energizing one solenoid causes the cylinder to extend and energizing the other solenoid causes it to retract. A limit switch at each end indicates full extension. Use two stop three wire controls, one for each direction. Construct a two directional control system, including interlocks to control the solenoid.
3. A stock control system consists of three shelves. The system gives you an indication that each shelf is full. Use counter function to count the parts that enter to its shelf. The capacity of the shelves is 100, 550 and 800 parts. When S1 is full a green light indicate, when S2 is full a yellow light is indicate, when S3 is full a red light is indicate, that means you could not add any parts to the shelves. In case of withdraw a parts from any shelf the light of the same shelf is turn off.
4. A part is placed on a conveyor, the part automatically moves down the conveyor. In the middle of the conveyor the part goes through a 2 meter long painting section. The sprayer paints for the time the part is under the booth, during which time the conveyor does not stop. When the part reaches the end of the conveyor, the conveyor stops and the part is removed. Assume that only one part can be on the conveyor at one time. (*Hint: use one limit switch at the front of the booth and another at the end*).
5. A wood saw, W, a fan, F, and a lubricant pump, P, all go ON when a start button is pushed. A stop button stops the saw only, the fan is to run an additional 5 seconds to blow the chips away, the lube pump is to run for 8 seconds after shutdown of W. Additionally, if the saw has run more than one minute, the fan should stay on indefinitely. The fan may then be turned off by pushing a separate fan reset button. If the saw has run less than one minute, the pump should go off when the saw is turned off. The 8 seconds time delay off does not take place for a running time of less than 1 minute.

ALMAWARED
ENGINEERING
&TRADING SAE

2 Ahmed Taysser St., Heliopolis, Cairo
Tel. : 24192480 | Fax : 24192483
almawared@link.net www.met-eg.com