

PROGRAMMABLE LOGIC CONTROLLERS MENG 3500



ControlLogix Controllers

- Automation companies have introduced a new class of industrial controllers known as programmable automation controllers (PACs).
- PAC combines PLC ruggedness with PC functionality.
- These devices resemble PLCs in physical appearance but integrate advanced features such as communication, data logging, signal processing, motion control, process control, and machine vision into a unified programming environment.
- A PLC mixes scan-based and event-driven program execution, whereas PAC software is typically event-driven.



ControlLogix Controllers

- PLC (Scan-Based + Event-Driven) Example: A conveyor system in a factory moves products and stops when a sensor detects an object. The PLC continuously scans input sensors, checks conditions, and controls the motor.
- PAC (Event-Driven) Example: A smart manufacturing system adjusts machine parameters based on real-time data from multiple sensors. Instead of scanning all logic repeatedly, the PAC processes events like temperature changes or remote commands to optimize performance dynamically.
- In short: PLCs are ideal for simple, repetitive control, while PACs handle complex, data-driven automation.



ControlLogix Controllers

PAC vs PLC

- PLCs are traditionally designed for discrete control applications, excelling in tasks that involve on/off control logic.
 - They are well-suited for applications in manufacturing, where precise timing and sequencing are crucial.
- On the other hand, PACs offer a more comprehensive approach by integrating both discrete and process control functionalities.
 - PACs are equipped with a broader range of capabilities, supporting not only sequential and logic control but also complex data handling and communication in continuous and batch processes.
 - Batch processes, such as pharmaceutical, food production, and chemical processing are examples of precise control systems that play a pivotal role in ensuring quality and consistency in the production of discrete units or batches.
 - Batch processes are involved in mixing specific quantities of raw materials, chemical reactions, and precise control over temperature and pressure to ensure the quality and consistency of the final product.

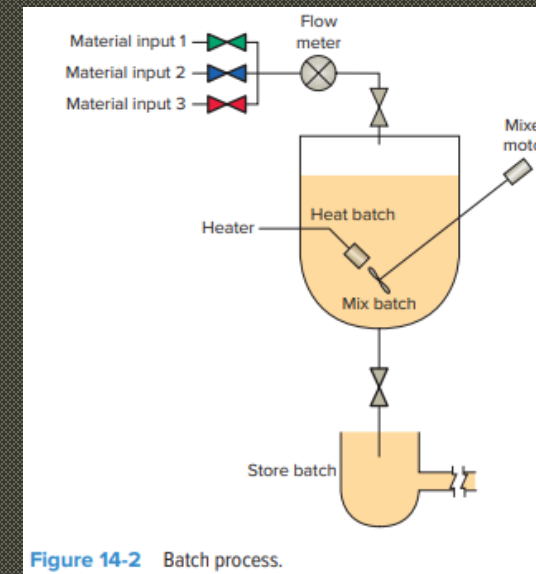


Figure 14-2 Batch process.

ControlLogix Controllers

Size and Application

- When classifying PLCs, we consider what they can do, how many inputs and outputs they have, their cost, and size, with the number of inputs and outputs being the most crucial factor.
- There are three major types of PLC application: single-ended, multitask, and control management.
 - A single-ended or stand-alone PLC application means one PLC controls one process, operating independently without communication with other computers or PLCs.
 - Multitask PLC application means one PLC handles multiple processes, needing enough I/O capacity. If it's part of a bigger system and needs to talk to a central PLC or computer, a data communications network is essential.
 - In a control management PLC application, one big PLC oversees others, needing a sizable processor for communication with other PLCs and maybe a computer. It supervises by sending programs to instruct other PLCs, requiring the ability to connect and communicate with any of them through proper addressing.

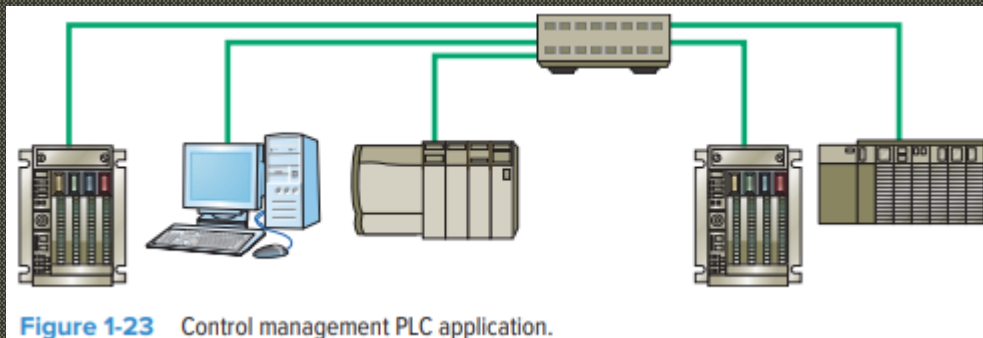


Figure 1-23 Control management PLC application.

ControlLogix Controllers

Memory

- PLC memory stores data, instructions, and the control program.
- Memory size is usually expressed in K or M values: 1 K, 16 K, 8 M, 16 M, etc.
 - 1 K means 1024, measured in the binary number system ($2^{10} = 1024$).
- A PLC's instructions set presents the various type of supported instructions.
- Memory needs in a PLC vary based on the application. Factors influencing the required memory size include:
 - Number of I/O points
 - Size of control program
 - Data-collecting requirements
 - Supervisory functions required
 - Future expansion

Table 1-1 Typical PLC Instructions

Instruction	Operation
XIC (Examine ON).....	Examine a bit for an ON or 1 condition
XIO (Examine OFF).....	Examine a bit for an OFF or 0 condition
OTE (Output Energize).....	Turn ON a bit (nonretentive)
OTL (Output Latch).....	Latch a bit (retentive)
OTU (Output Unlatch).....	Unlatch a bit (retentive)
TOF (Timer Off-Delay).....	Turn an output ON or OFF after its rung has been OFF for a preset time interval
TON (Timer On-Delay).....	Turn an output ON or OFF after its rung has been ON for a preset time interval
CTD (Count Down).....	Use a software counter to count down from a specified value
CTU (Count Up).....	Use a software counter to count up to a specified value

ControlLogix Controllers Configuration

- In a ControlLogix controller, there is no set memory for specific data or I/O.
- Users configure the internal memory layout using the programming software during project creation.
- The software needs hardware details like the processor and I/O modules.
- The communication software is used to set up a communications link between the programming software and the PLC hardware.



Figure 15-2 RSLogix 5000 screen.
Source: Image Courtesy of Rockwell Automation, Inc.

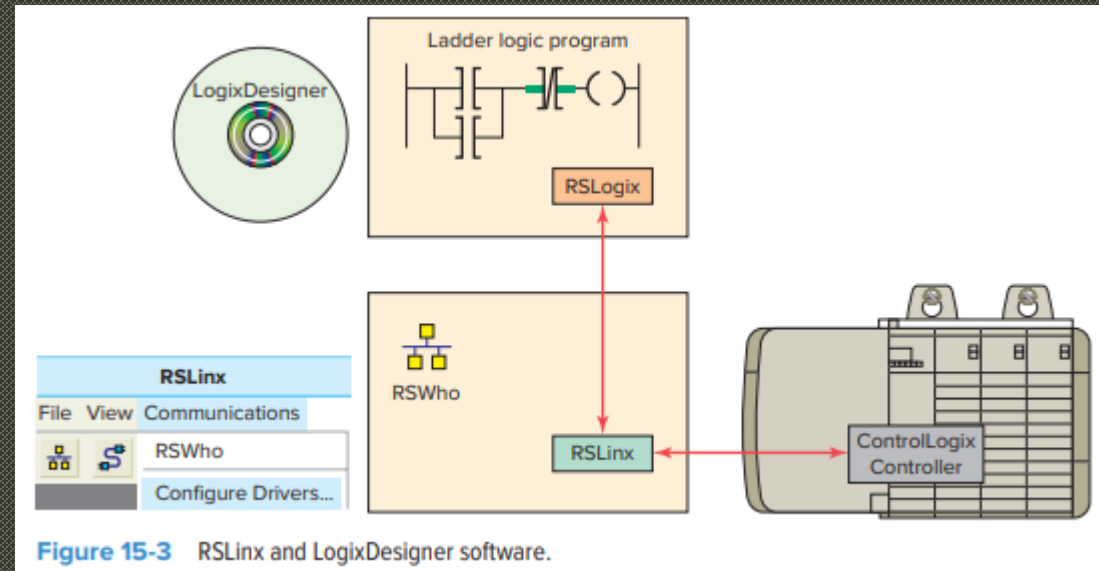


Figure 15-3 RSLinx and LogixDesigner software.

ControlLogix Controllers Project

- RSLogix software stores a controller's programming and configuration information in a file called a project.
- The main components of the project file are tasks, programs, and routines.
- A controller can hold and execute only one project at a time.
- Global vs Local Data, System Shared Data (Produced & Consumed tags)

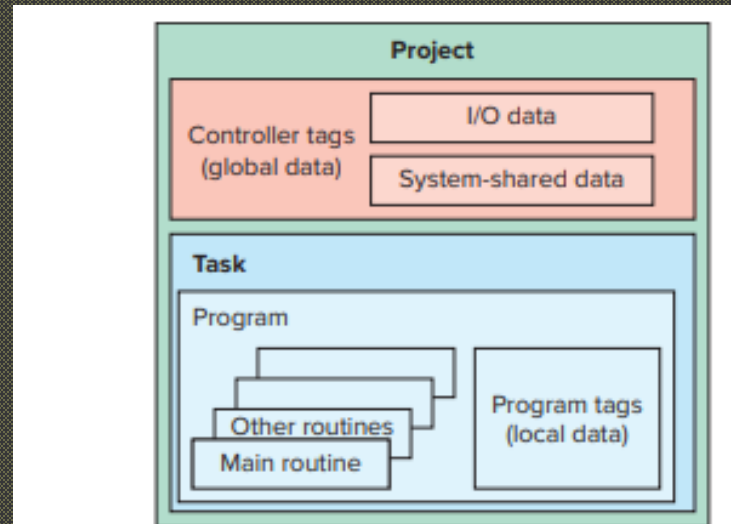


Figure 15-5 ControlLogix processor program file.

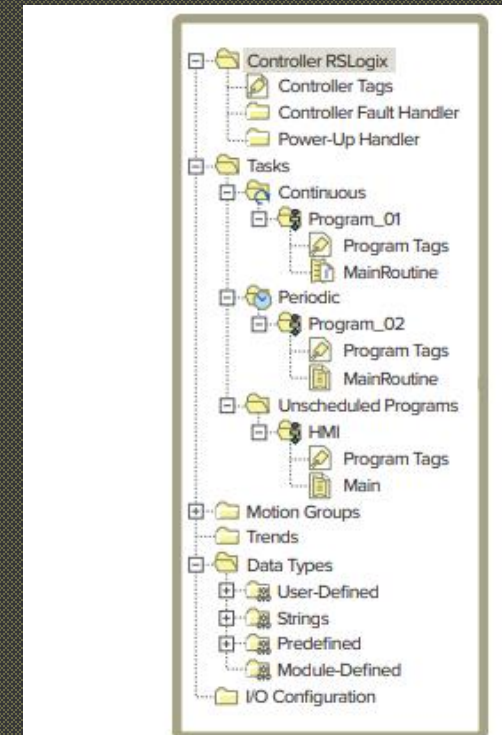


Figure 15-6 Controller organizer tree.

ControlLogix Controllers

Tasks

- Tasks, the **first level** of scheduling in a project, are groups of scheduled programs.
- When a task runs, its programs execute in order.
- Only one task runs at a time. The number of tasks depends on the specific controller.
- The main types of tasks include:
 - Continuous tasks execute nonstop but are always interrupted by a periodic task.
 - Lowest priority.
 - Named Main Task.
 - Periodic tasks act like timed interrupts.
 - They interrupt the continuous task, running for a fixed duration at specific intervals.
 - Event tasks also act as interrupts.
 - It differs from timed interrupts as it's triggered by an occurrence or failure to occur.

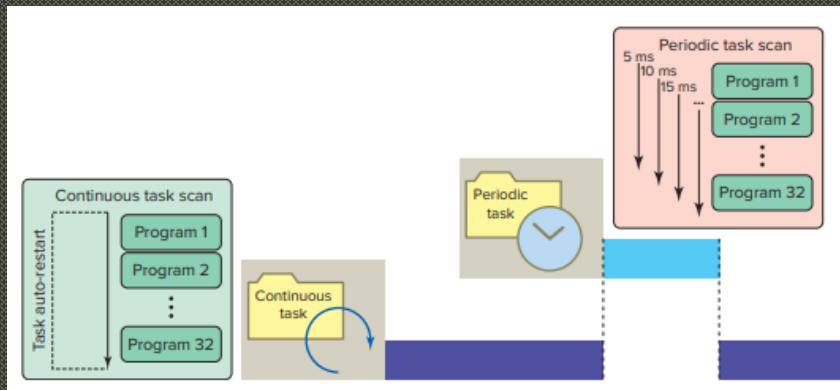


Figure 15-7 Continuous and periodic tasks.

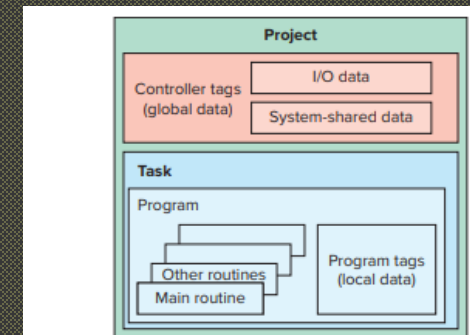


Figure 15-5 ControlLogix processor program file.

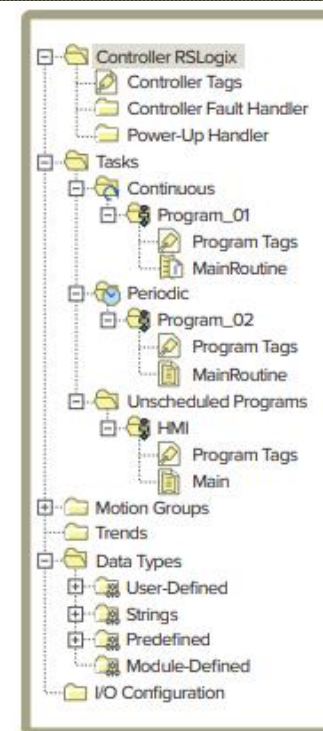
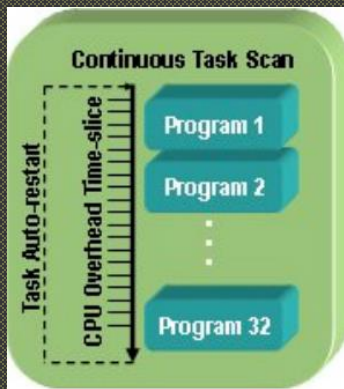


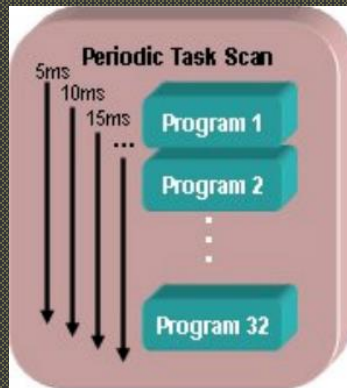
Figure 15-6 Controller organizer tree.

ControlLogix Controllers Tasks

In a continuous task, the controller scans the code top to bottom and then repeats the process.



A periodic task runs top to bottom like a continuous task but waits a set time before restarting.



An event task runs only when triggered, interrupting lower-priority tasks, executing once, and then resuming the previous task.

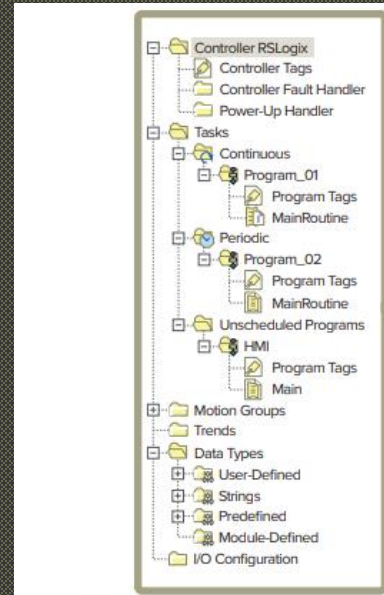
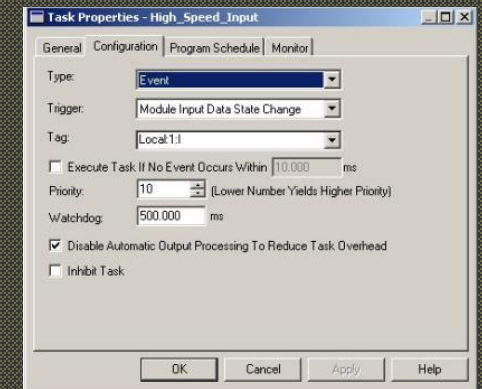
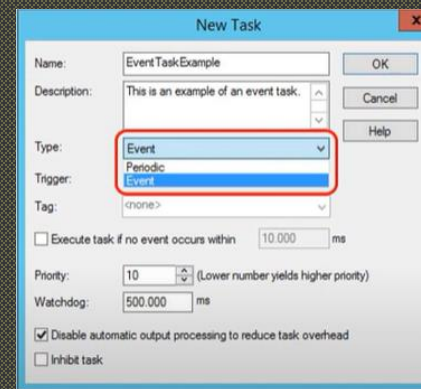
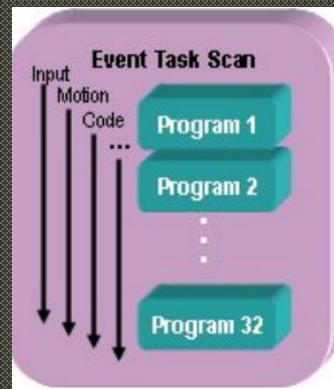


Figure 15-6 Controller organizer tree.

ControlLogix Controllers Programs - Routines

- Programs are the **second level** of scheduling within a project.
- The folders under Main Task specify the order in which programs should execute.
- Programs that are not assigned to a task are unscheduled.
 - Unscheduled programs, once downloaded to the controller, stay inactive until they are needed.
- Routines, at the **third scheduling level**, provide the project's executable code.
- Each routine contains a set of logic elements for a specific programming language.
 - Main routine and subroutines.

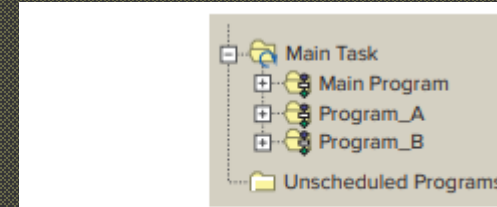


Figure 15-8 Order of execution of programs.

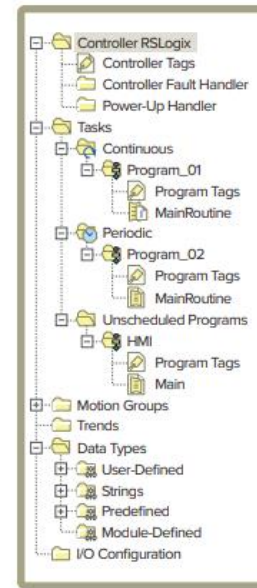


Figure 15-6 Controller organizer tree.

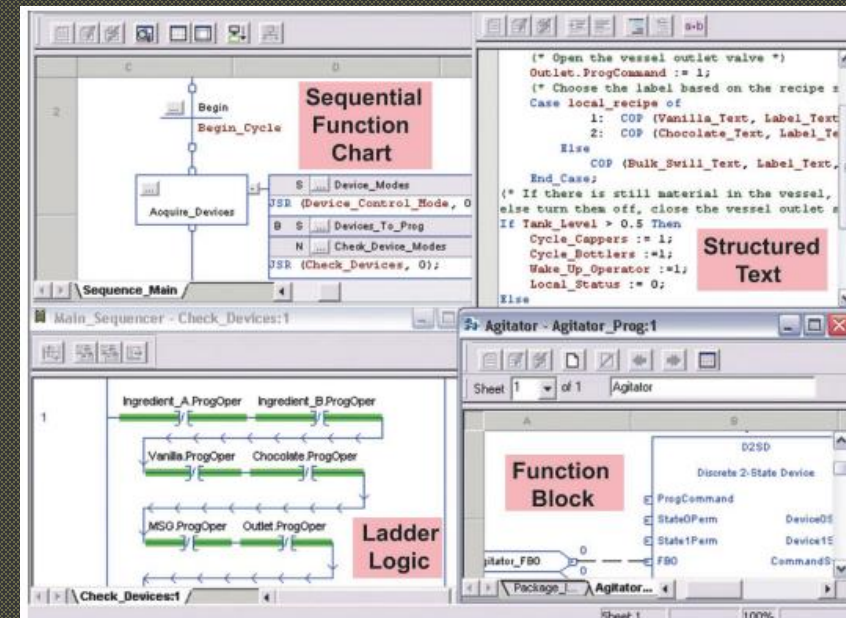
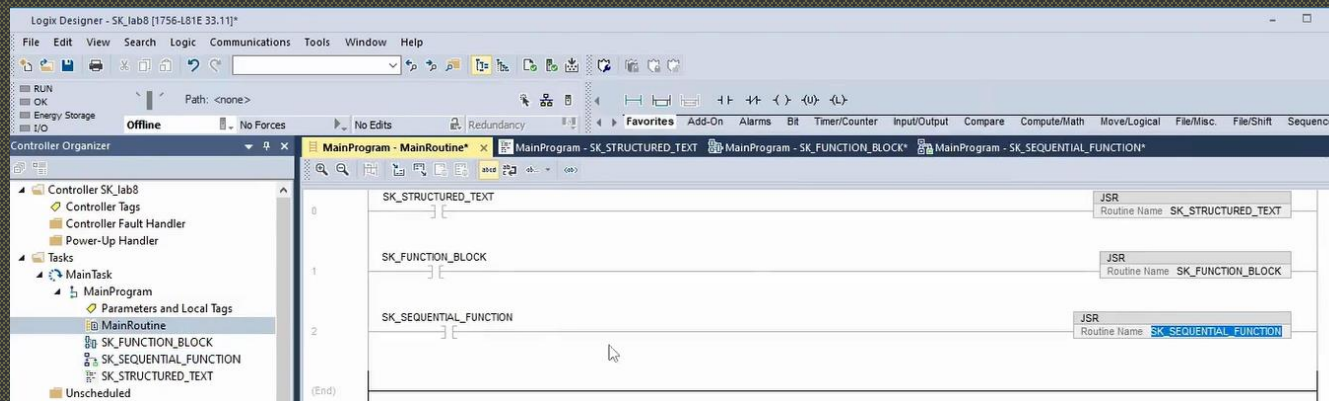


Figure 15-9 Each routine contains a set of logic elements for a specific programming language.
Source: Image Courtesy of Rockwell Automation, Inc.

ControlLogix Controllers

Tags

- ControlLogix uses a tag-based addressing structure.
- Tags are names that describe a specific application.
- A tag represents and identifies data locations in the controller's memory.
- To group data, create an array – a collection of similar tags.
- Scope determines which programs can access a tag.
 - A program tag contains data accessible only by routines within its specific program (local data).
 - Routines in other programs can't access program-scoped tags from another program.
 - A controller tag contains data accessible by all routines within a controller (global data). I/O tags are automatically created as controller scoped tags

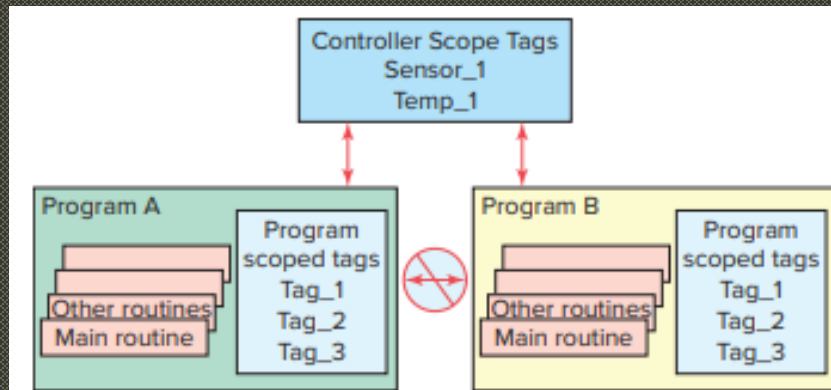


Figure 15-11 Program scoped and controller scoped tags.

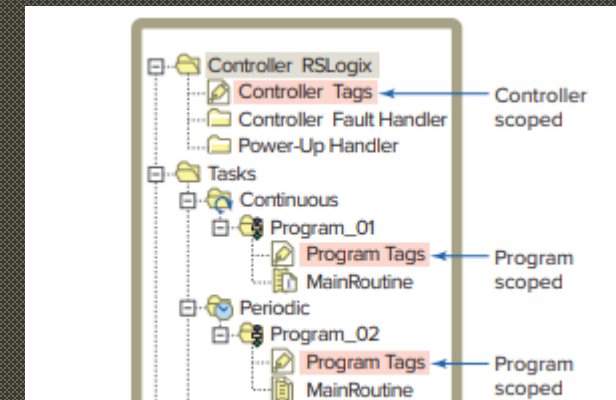
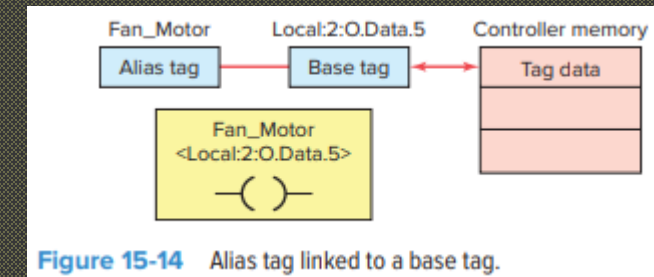
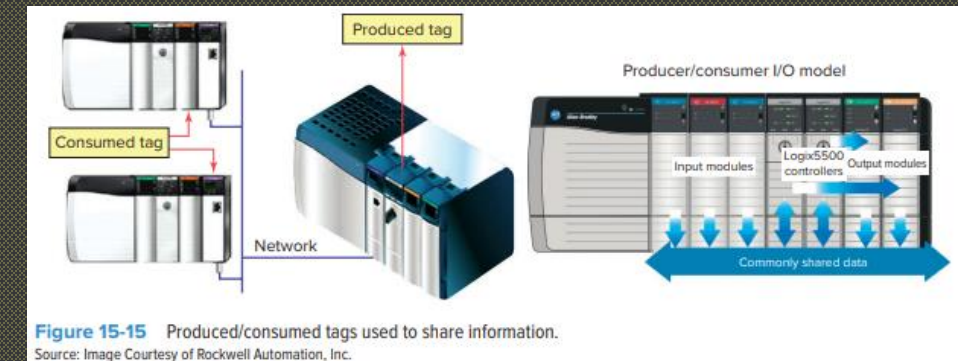
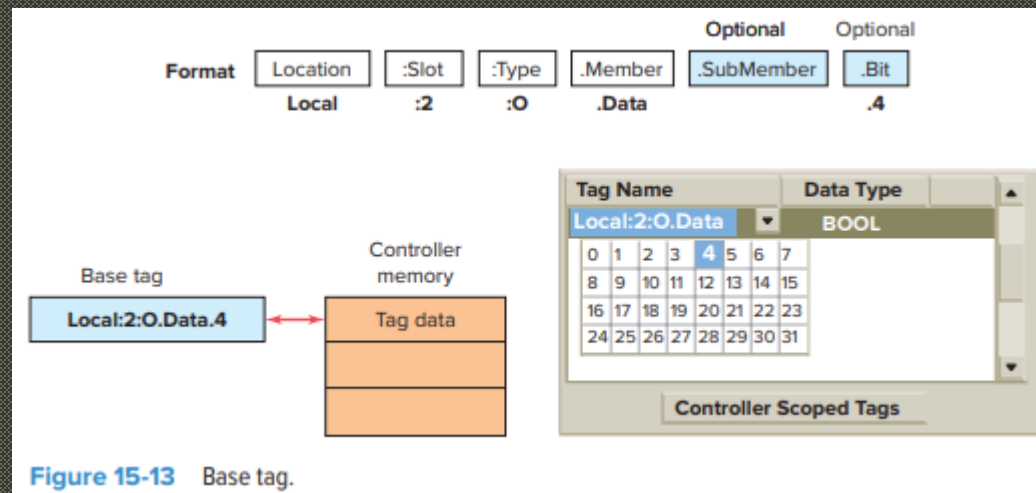
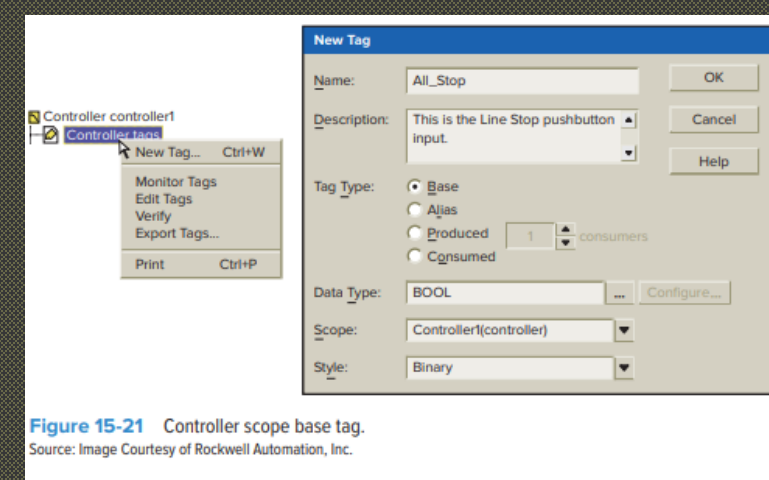


Figure 15-12 Listing of program and controller scoped tags.

ControlLogix Controllers Tags

- There are four different tag types: base, alias, produced, and consumed tags.
 - A base tag stores different data types for project logic use.
 - An alias tag provides an alternate name for a tag.
 - The alias tag is used to create a tag name for a physical input or output.
 - Produced/consumed tags share information between devices on a network.
 - A produced tag sends data while a consumed tag receives data.
 - Produced tags are always controller scoped.



ControlLogix Controllers

Tags

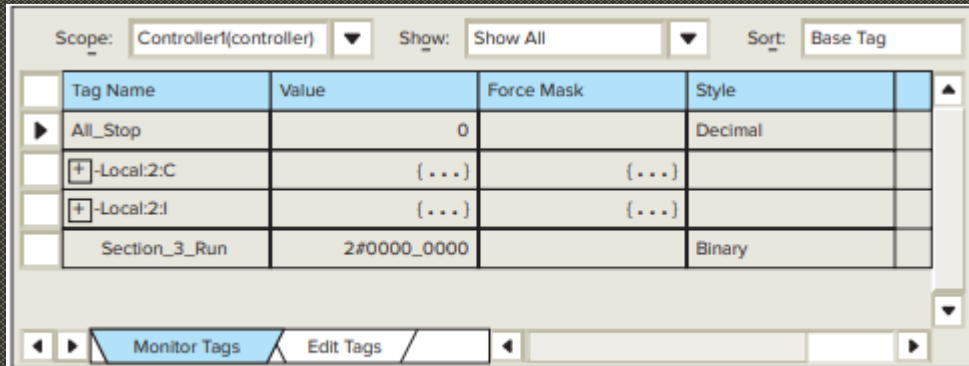


Figure 15-22 Monitor Tags window.

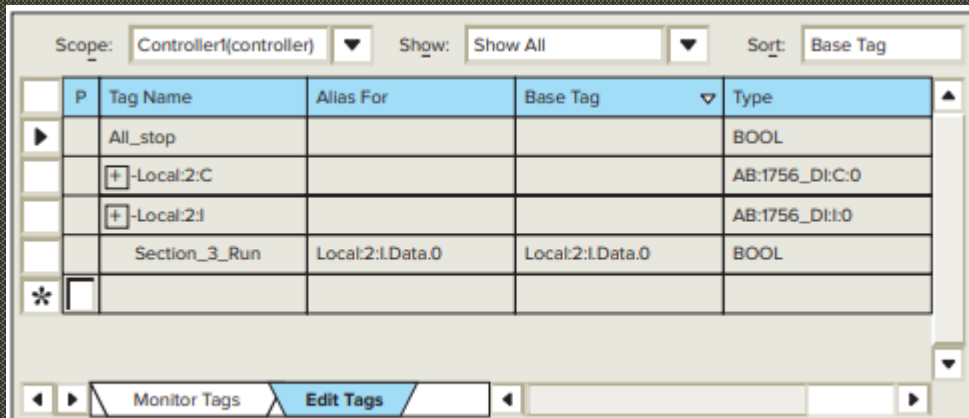


Figure 15-23 Edit Tags window.

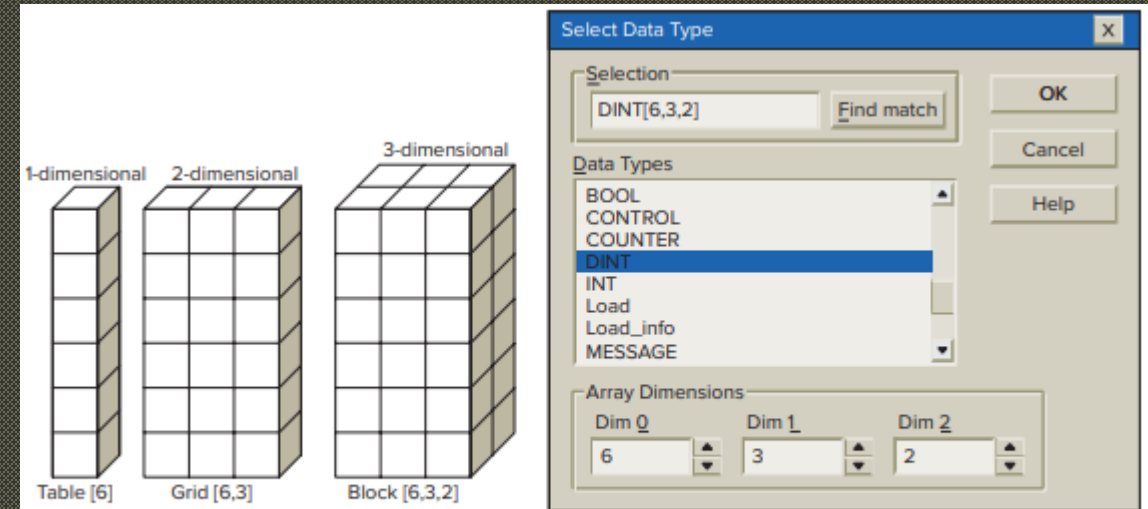


Figure 15-24 Types of arrays.

Source: Image Courtesy of Rockwell Automation, Inc.

Array - Temp
Data Type - INT[5]

Temp[0]	297
Temp[1]	200
Temp[2]	180
Temp[3]	120
Temp[4]	100

Figure 15-25 Memory layout for a one-dimensional array.

ControlLogix Controllers

Structures – Additional Data Type

- Logix controllers are based on 32-bit operations. The types of data that can be a base tag are BOOL, SINT, INT, DINT, and REAL.
- In a ControlLogix controller, there are three structure types: **predefined**, **module-defined**, and **user-defined**.
 - The controller automatically creates predefined structures, including timers, counters, messages, and PID instructions.
 - Module-defined structures are automatically created when configuring I/O modules for the system.
 - A user-defined structure complements predefined structures, allowing the creation of custom structures to store and handle data as a group.

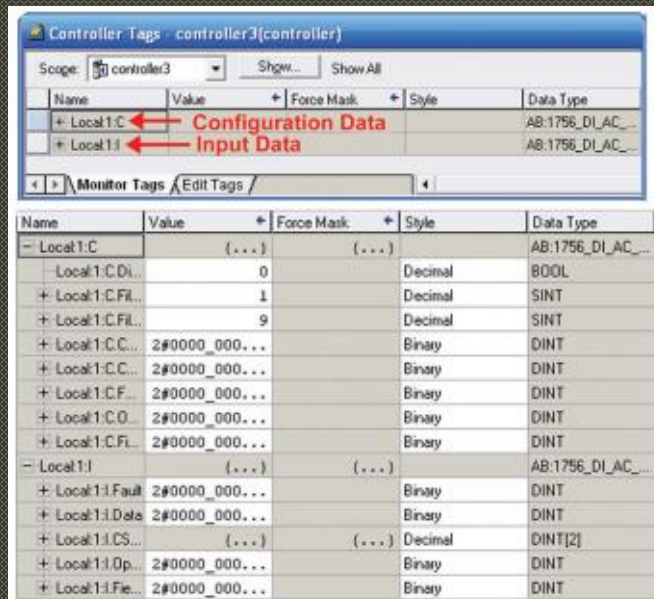
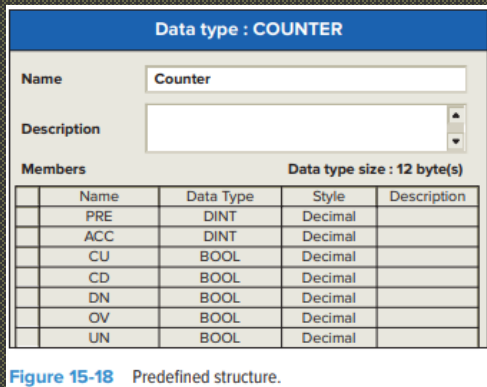
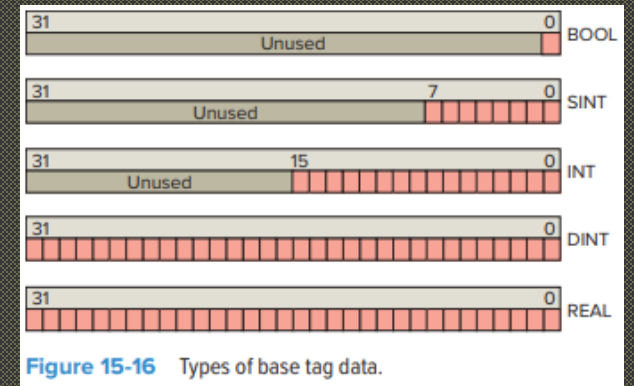
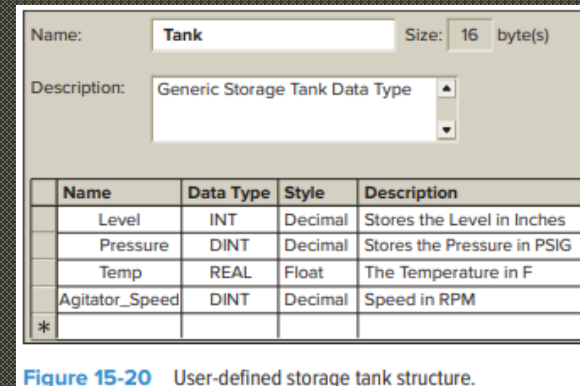


Figure 15-19 Module-defined structure for a digital input module.
Source: Image Courtesy of Rockwell Automation, Inc.



ControlLogix Controllers Addresses

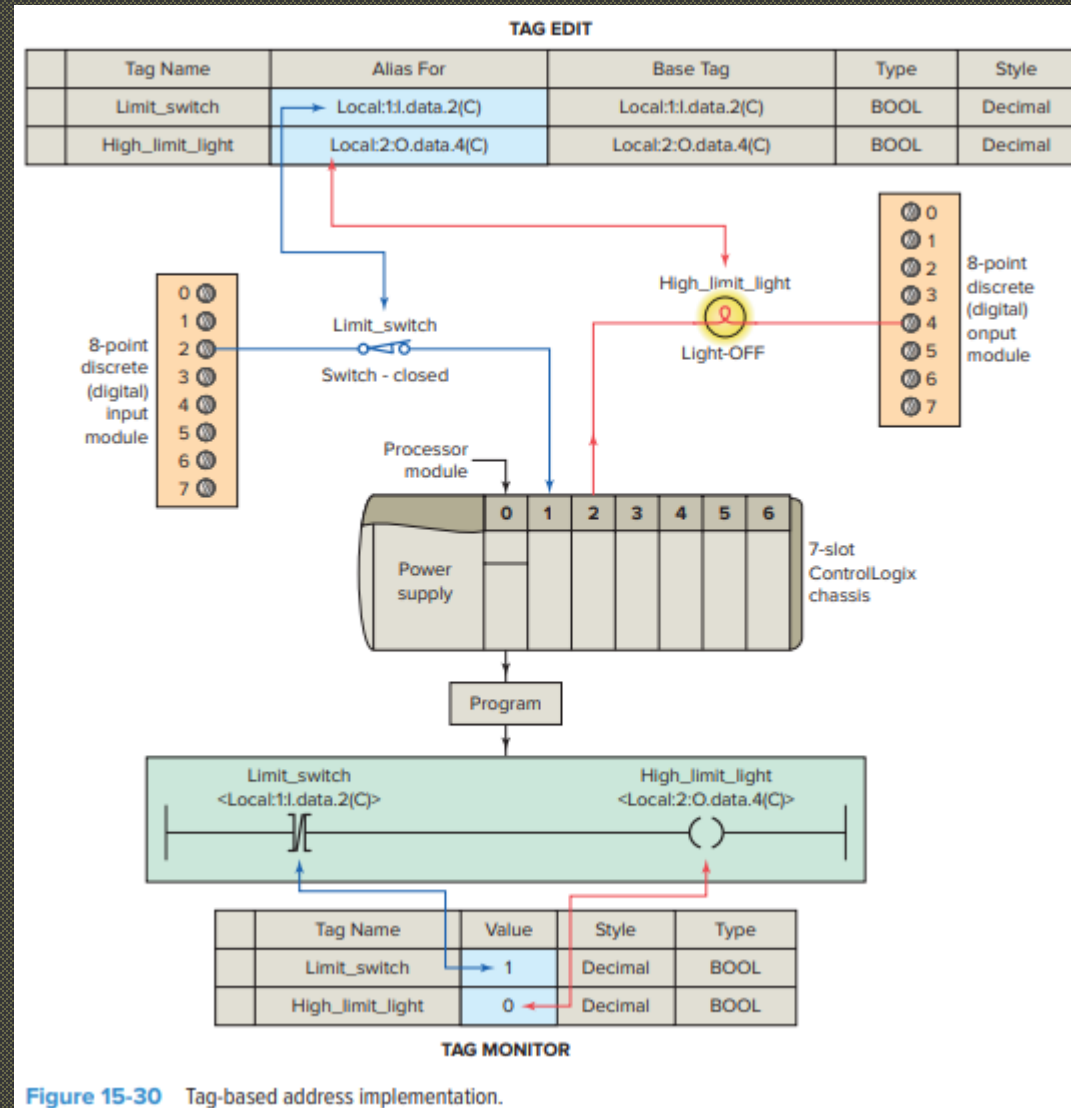
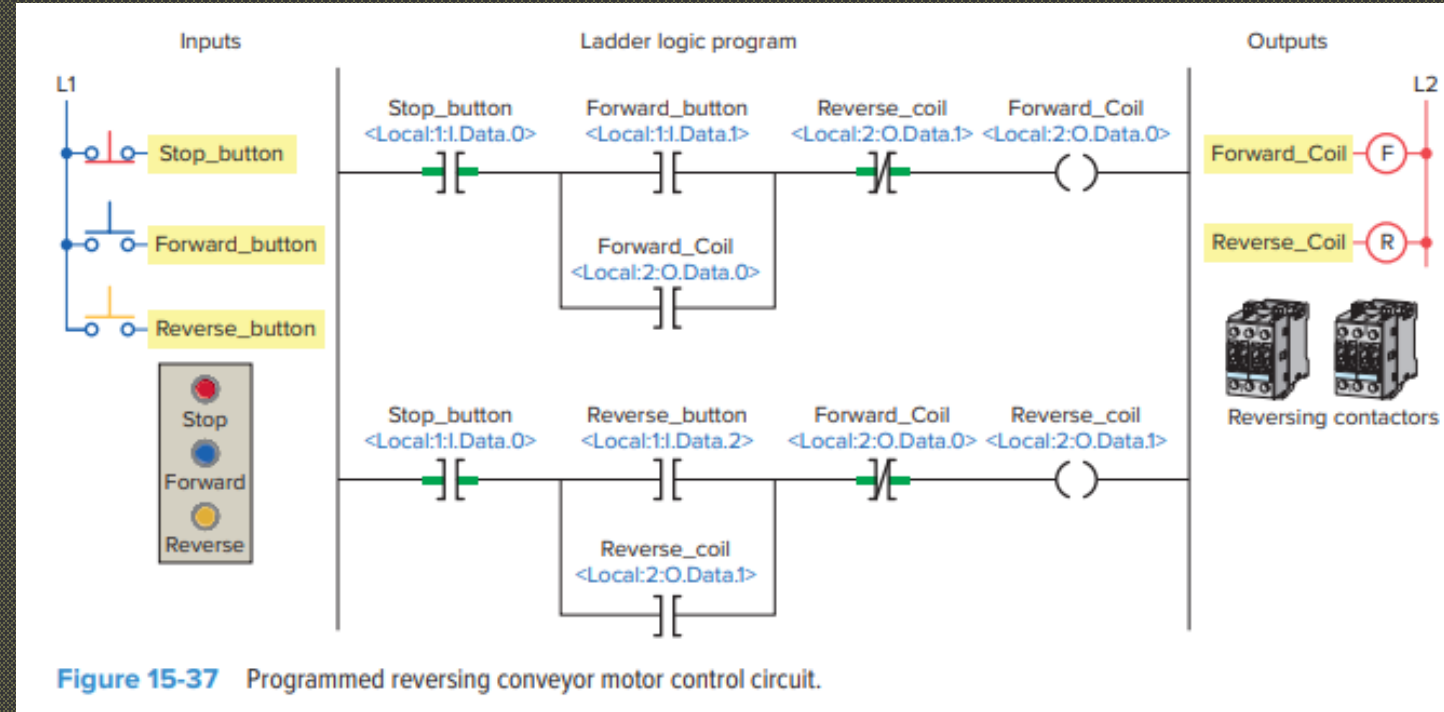
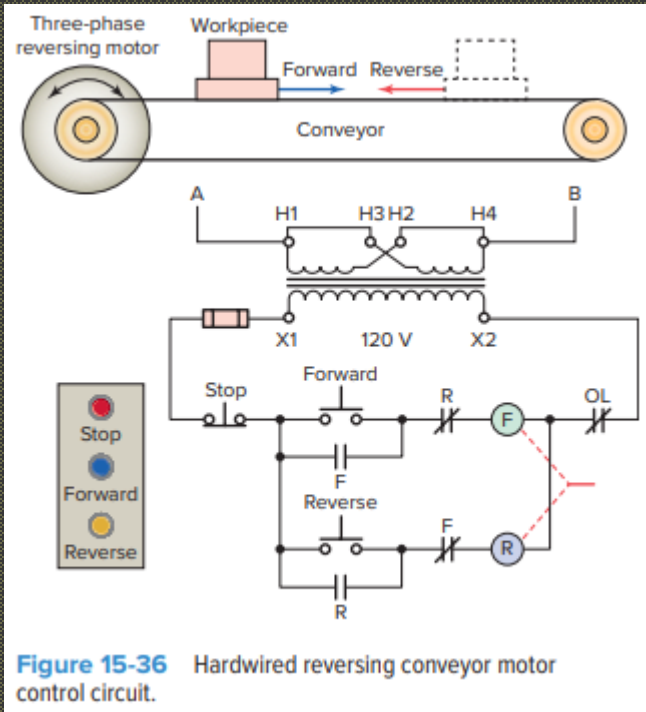


Figure 15-30 Tag-based address implementation.

ControlLogix Controllers Program – Motor Control



To reverse the motor using this control circuit, the operator needs to press the stop button first. This action de-energizes the coil and closes the normally closed contact again.

ControlLogix Controllers

Program – Light Control

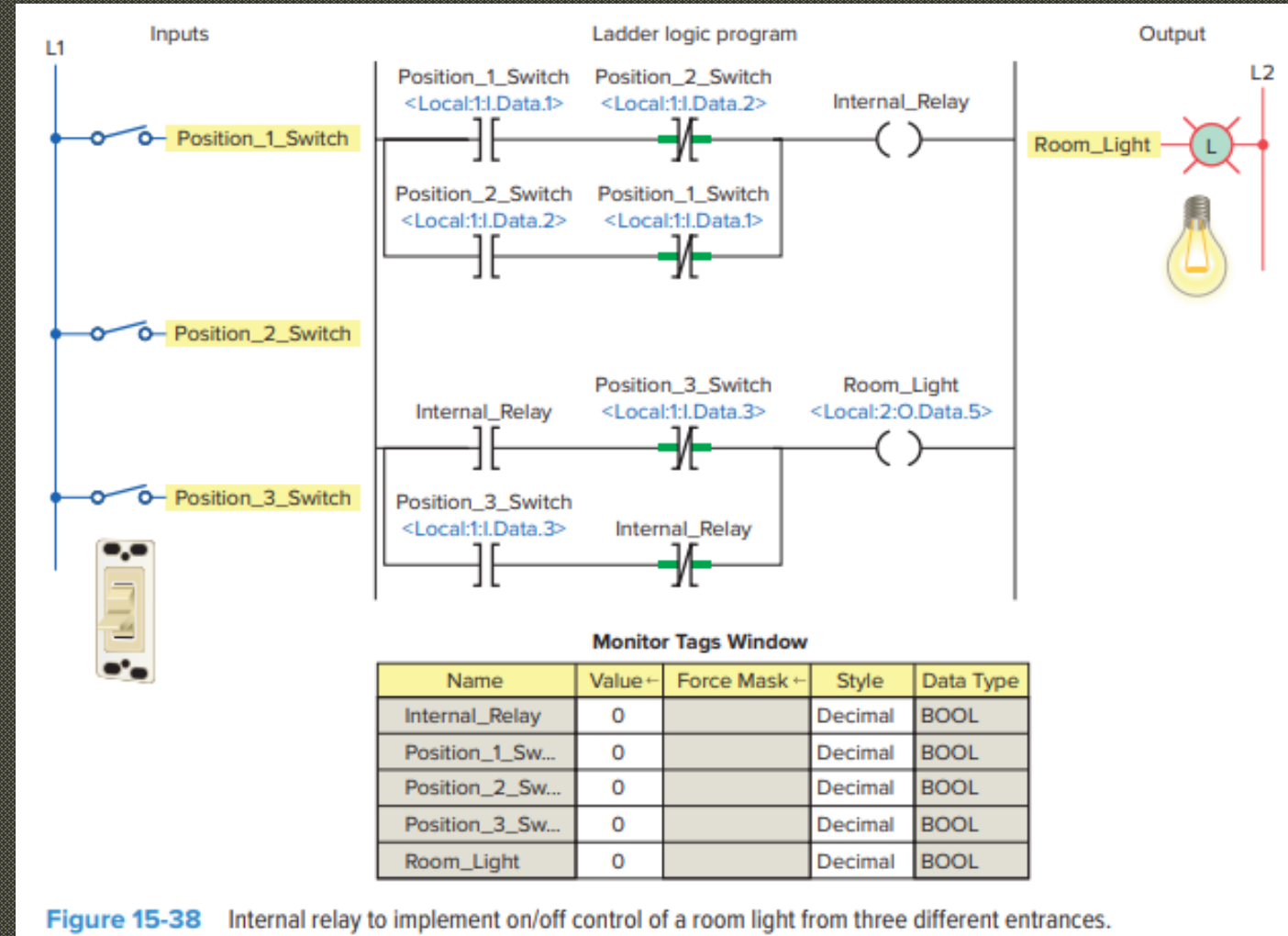
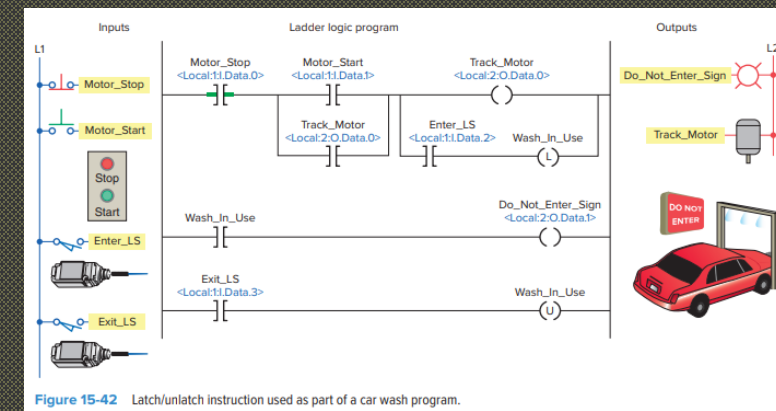


Figure 15-38 Internal relay to implement on/off control of a room light from three different entrances.

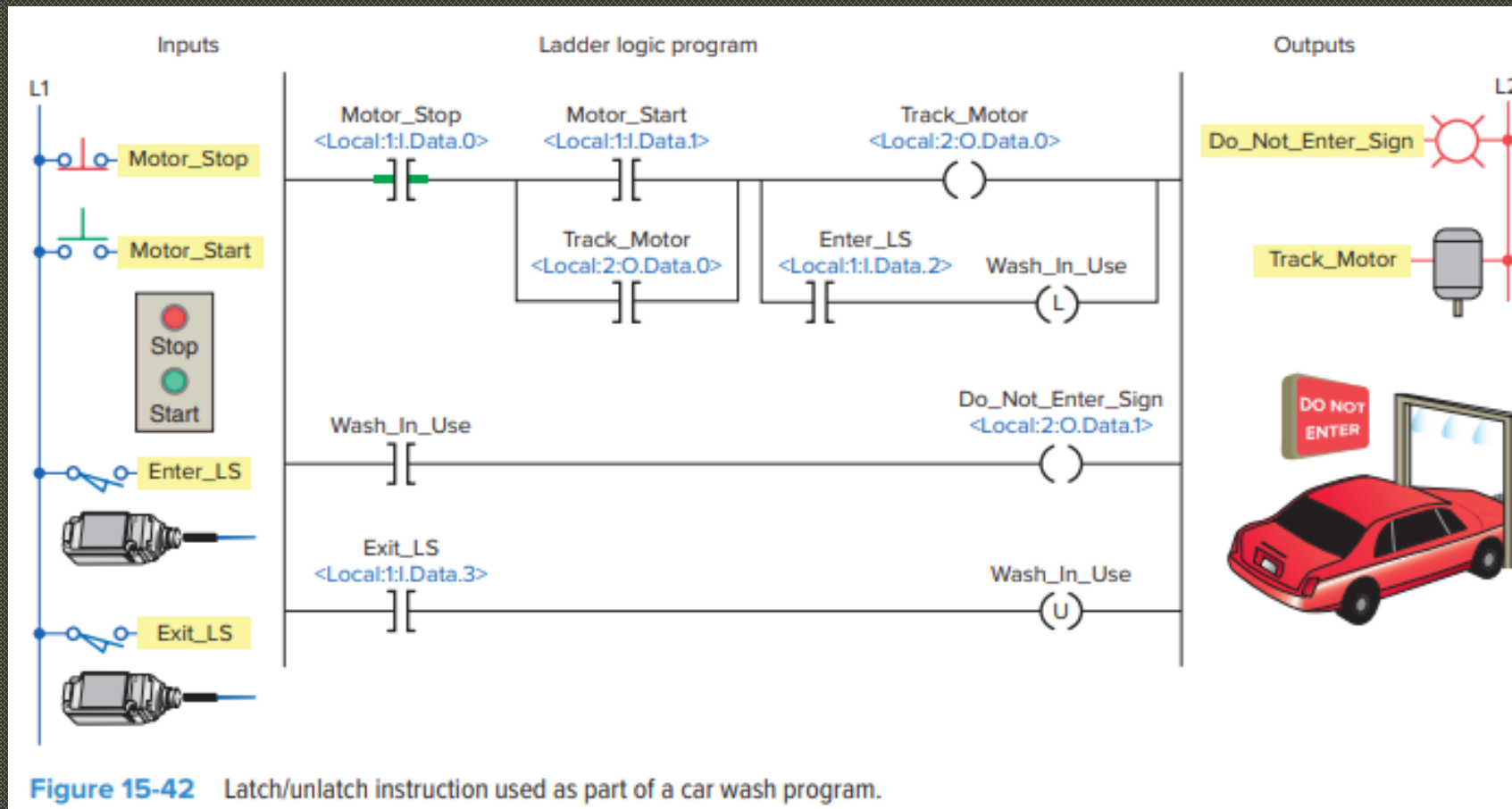
ControlLogix Controllers Program – Car Wash



The operation of the program can be summarized as follows:

- The car wash only washes one car at a time.
- The operator controls the ON/OFF operation of the track motor with the track motor Stop and Start PB.
- When the Motor_Start button is closed the Track_Motor is energized and performs a seal-in function that keeps the motor operating when the Motor_Start button is released.
- As the car enters the car wash, the Enter_LS contact momentarily closes to energize the Wash_In_Use latch instruction.
- This, in turn, energizes the Do_Not_Enter_Sign to indicate that the car wash is in use.
- When the car exits the car wash, the Exit_LS contact momentarily closes and energizes the Wash_In_Use unlatch instruction.
- This, in turn, de-energizes the Do_Not_Enter_Sign to indicate the car wash is not in use.
- The track motor remains running ready to restart the process once another car enters.
- Once running, the track motor can be stopped at any time by operating the Motor_Stop button to de-energize the Track_Motor.

ControlLogix Controllers Program – Car Wash



ControlLogix Controllers

Program – Conveyor Speed Control

The system comes equipped with a weight detector and three-speed motor controller.

It is designed to move the conveyor belt at a certain speed when a specific value of weight is on the conveyor.

- If the weight exceeds the preset value, the conveyor speed increases to compensate for the increase in weight.
- If the weight falls below the preset value, the conveyor speed is reduced to compensate for the decrease in weight.

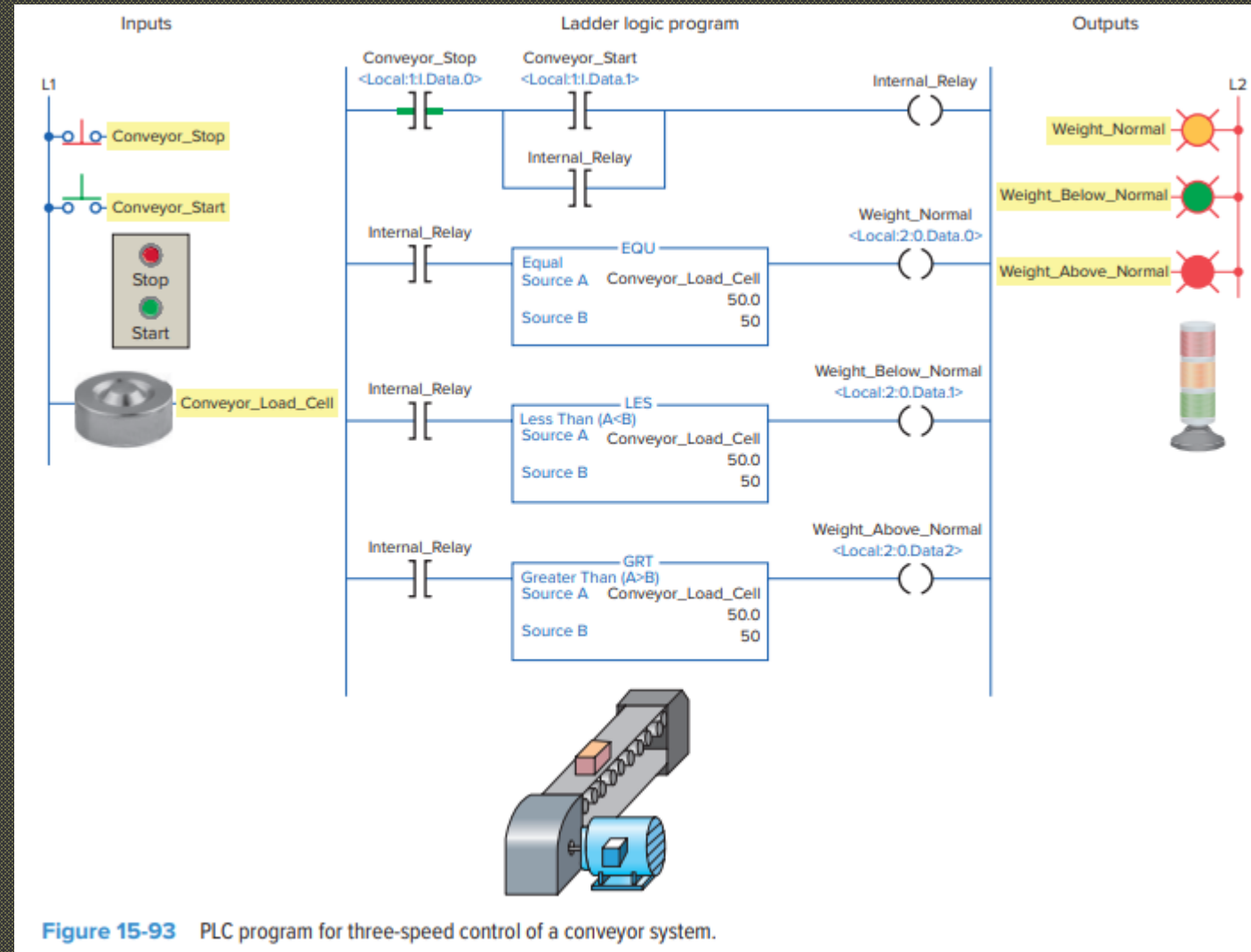
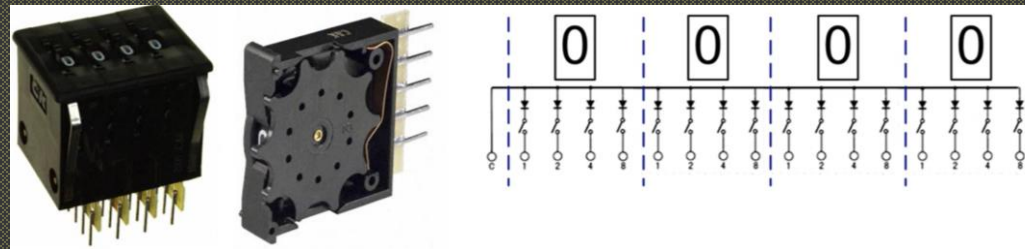
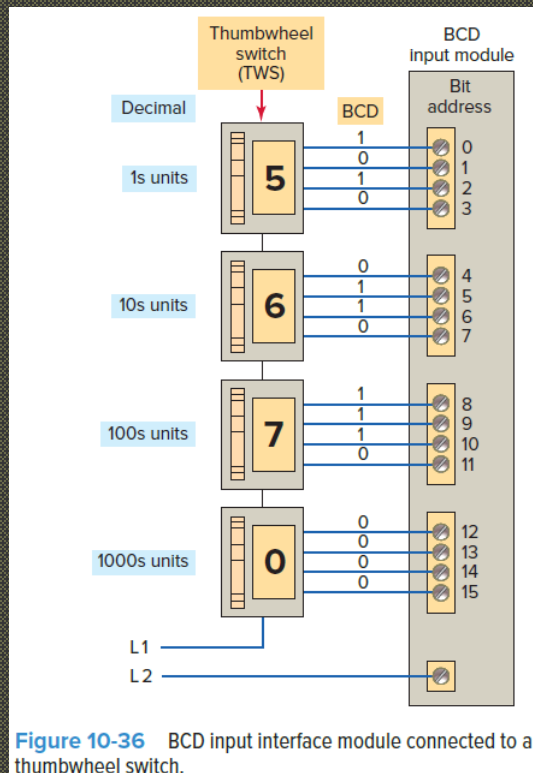


Figure 15-93 PLC program for three-speed control of a conveyor system.

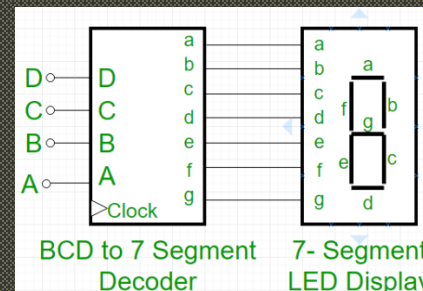
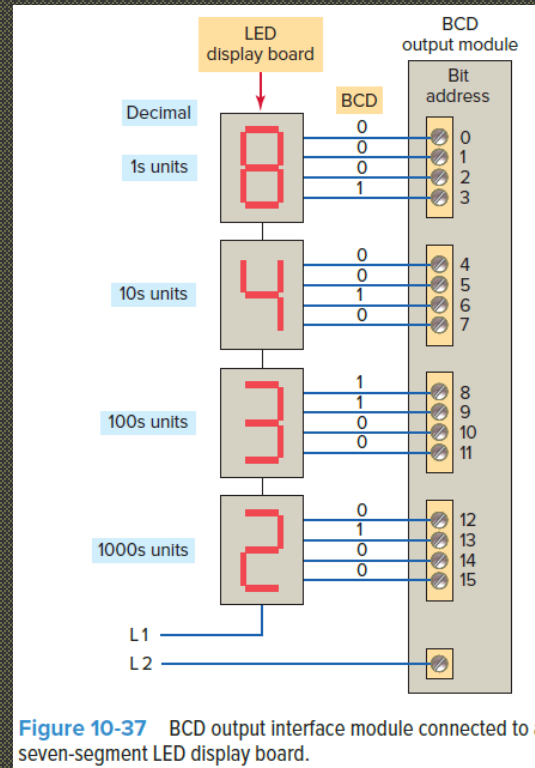
BINARY CODE DECIMAL - BCD

- Thumbwheel switches (TWS) are BCD input devices.
- Each of the four switches outputs four binary digits, representing the selected decimal number.
- TWS converts a decimal digit into four binary digits.
- Data manipulation instructions allow users to change set points, timers, or counter presets externally without modifying the control program.



BINARY CODE DECIMAL - BCD

- The seven-segment LED display is a BCD output device.
- It displays a decimal number corresponding to the BCD value received at its input.
- The conversion of four binary bits into a single decimal digit is performed by the LED display.



A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

BINARY CODE DECIMAL - BCD

- Modern PLC software no longer requires special BCD input/output modules.
- Standard digital I/O modules can be used with BCD conversion instructions (FRD & TOD).
- The PLC does not recognize BCD as a distinct format but interprets it as a binary number with different weight positions.

Binary Weighting															
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	1	0	0	0	1	1	1	0	1	1	0	1

- For example: Assume 79 is displayed on the thumbwheels.



- The BCD value would be 7 9: 0 1 1 1 1 0 0 1

8	4	2	1	8	4	2	1
0	1	1	1	1	0	0	1

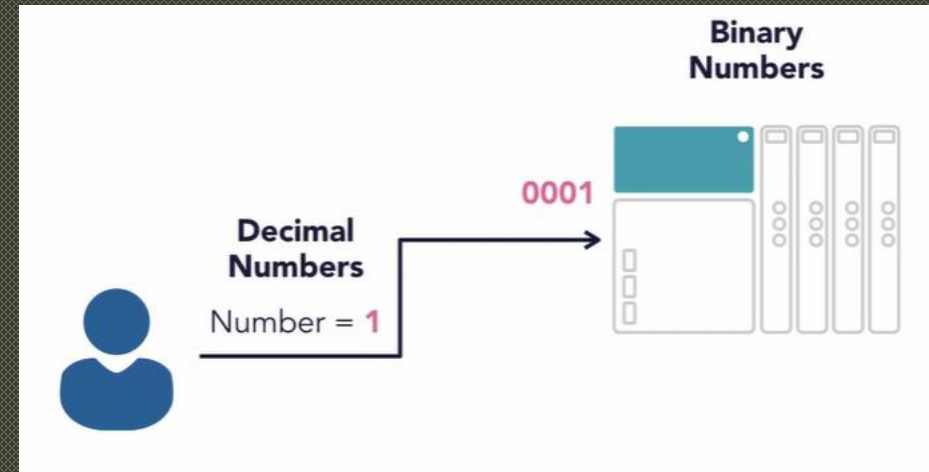
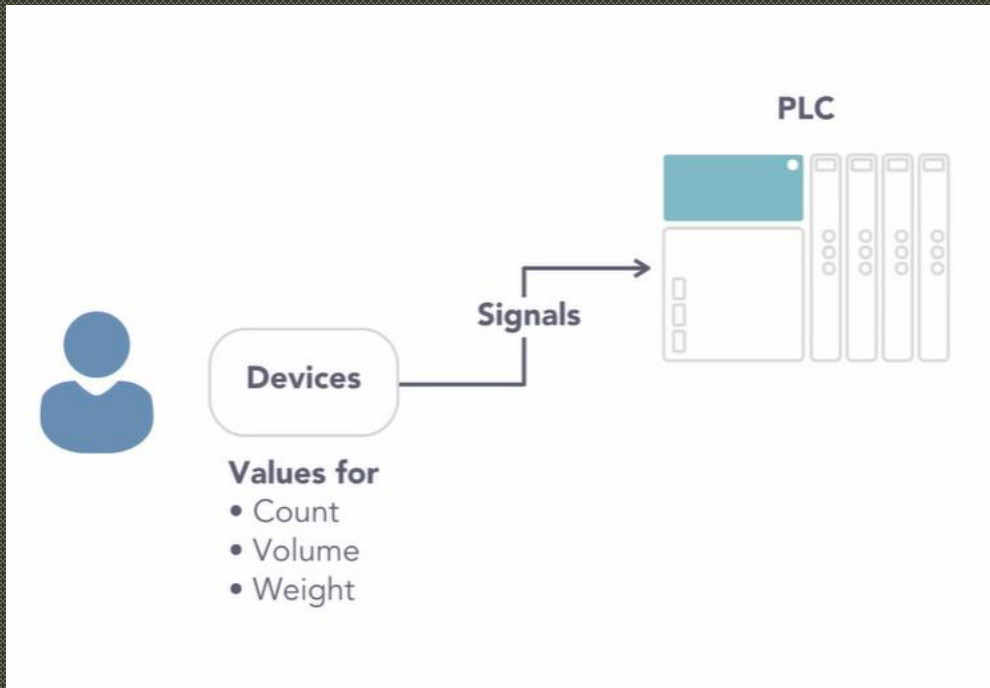
- The PLC views it as 121:

64	32	16	8	1
0	1	1	1	1

- The FRD (convert to integer) instruction is used to convert a BCD value to a decimal value.
- The TOD (convert to BCD) instruction is used to convert a decimal value to a BCD value.

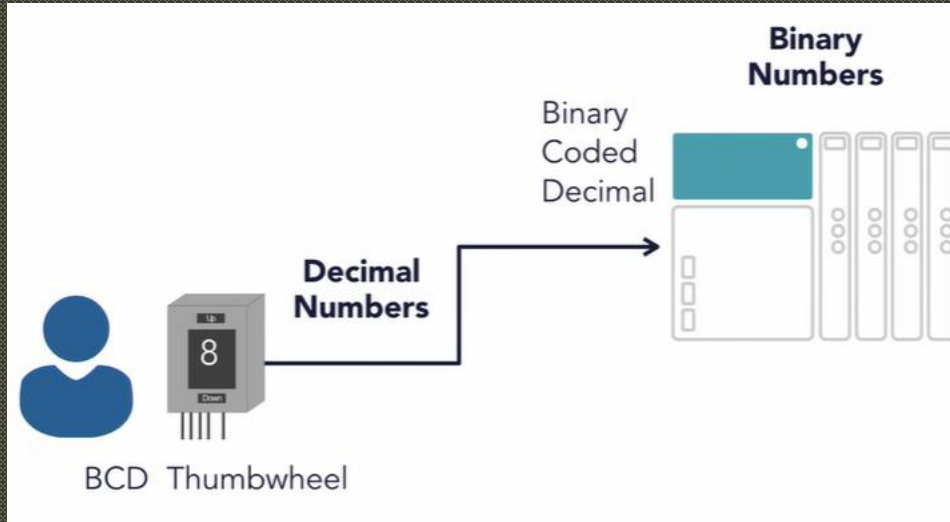
BINARY CODE DECIMAL - BCD

It is preferred to work with decimal numbers when using PLCs in industrial applications, while the PLC operates binary information.




BINARY CODE DECIMAL - BCD

TWS is the example of inputting a numerical data into the PLC.
In the PLC, the numerical data from TWS is entered as BCD.



The Binary Coded Decimal (BCD)

Decimal  Binary

The Binary Coded Decimal (BCD) System

Uses 4 bits to represent each decimal digit

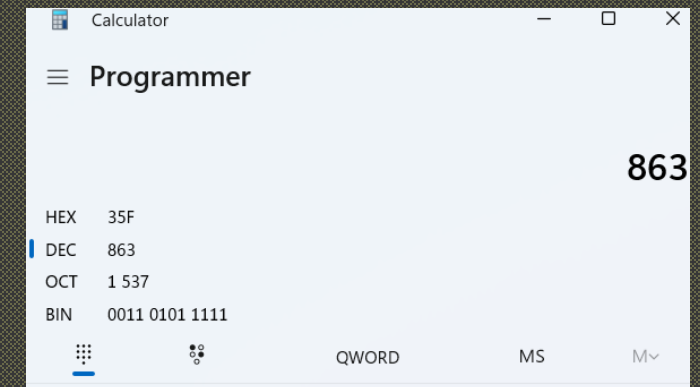
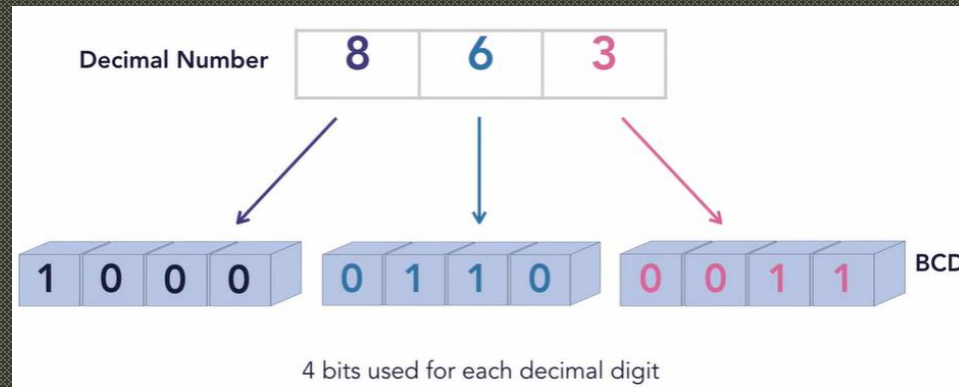
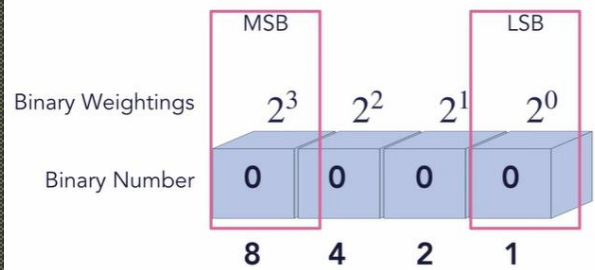


The binary equivalents of the numbers from 0 to 9

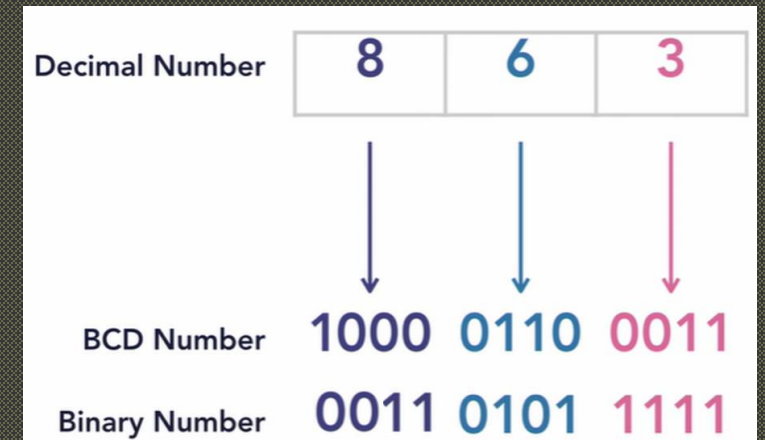
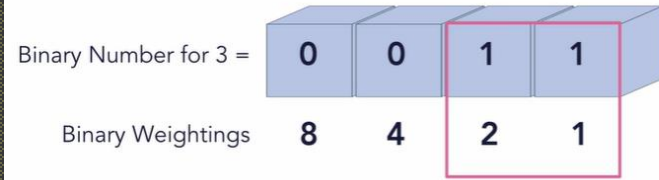
BINARY CODE DECIMAL - BCD

In the BCD format the largest decimal number that can be displayed by any four digits is 9.

Binary Number

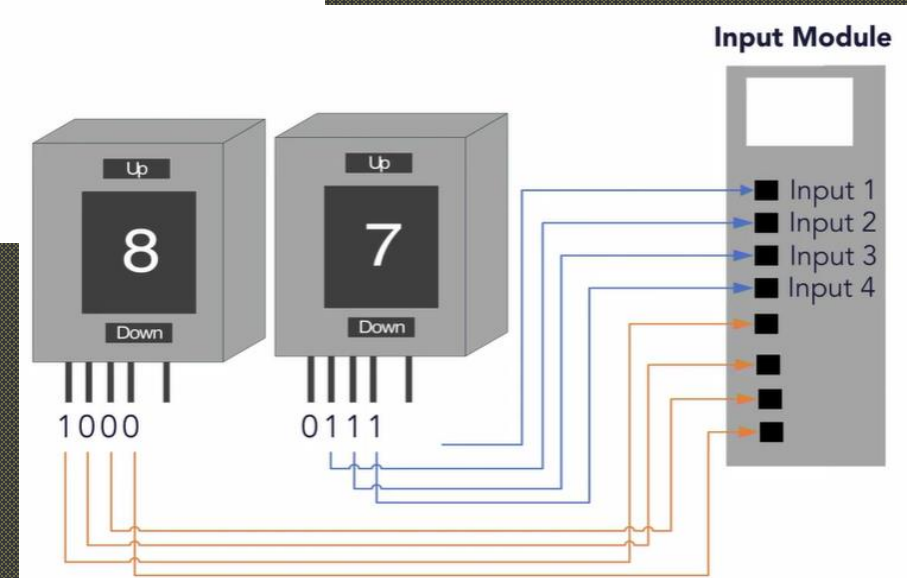
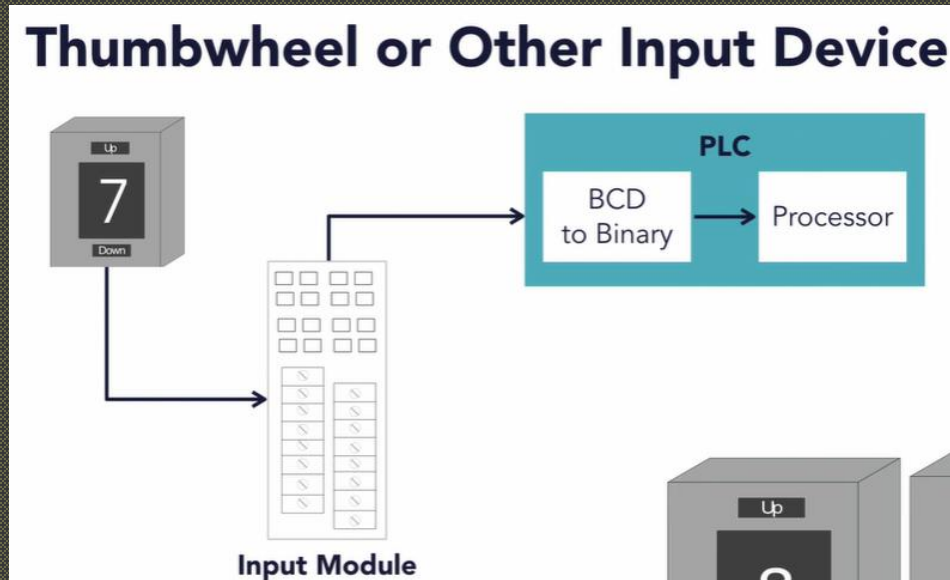
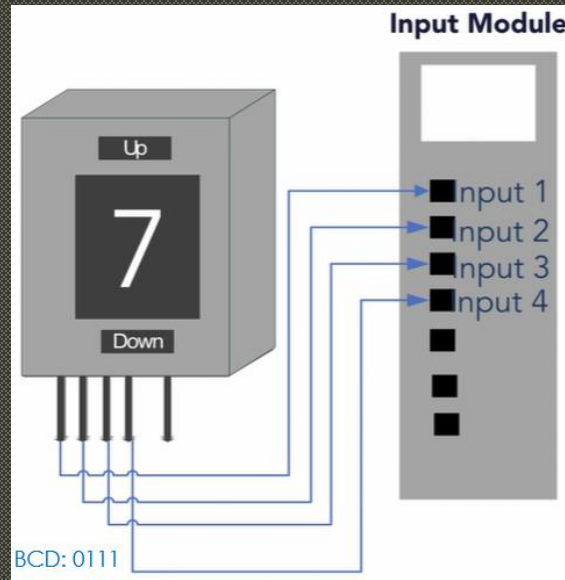


Decimal Number = 3



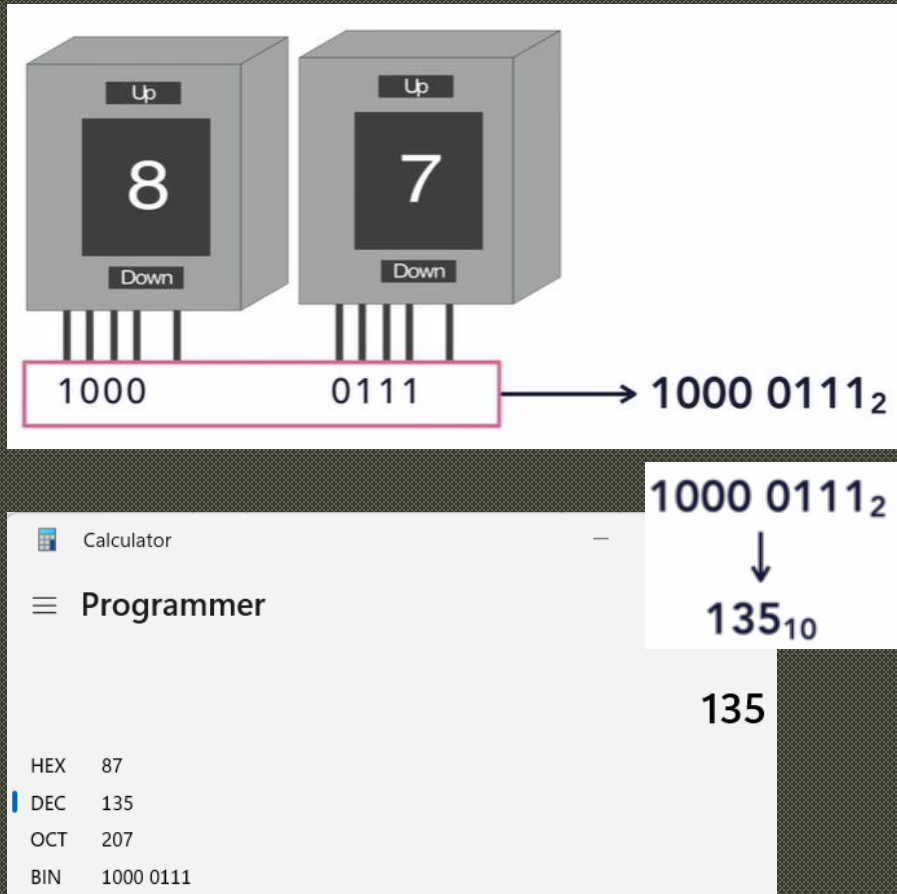
BINARY CODE DECIMAL – THUMBWHEEL

Inside the PLC: BCD - Binary conversion for the processor.

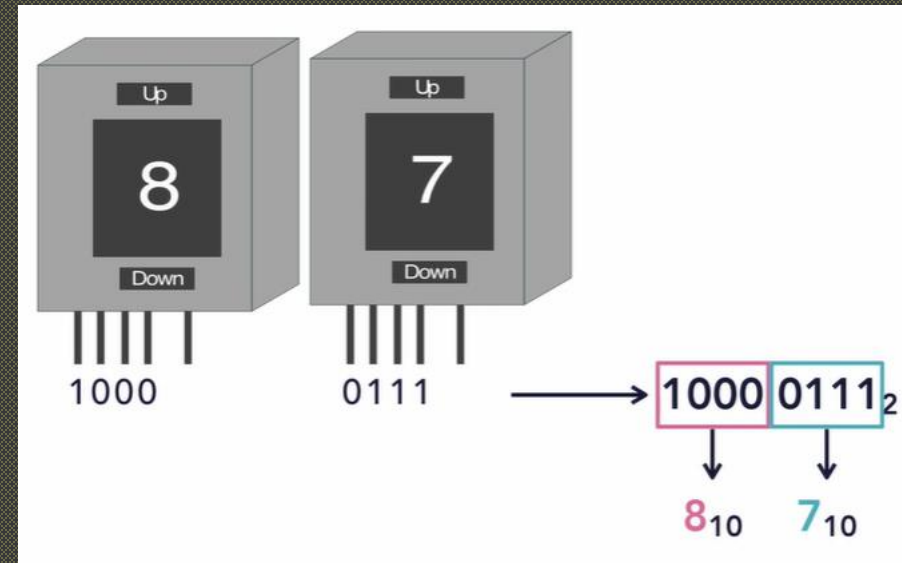


Single-digit BCD values (0–9) do not cause issues.
Problems arise when multi-digit numbers are used, requiring proper handling.

BINARY CODE DECIMAL - THUMBWHEEL

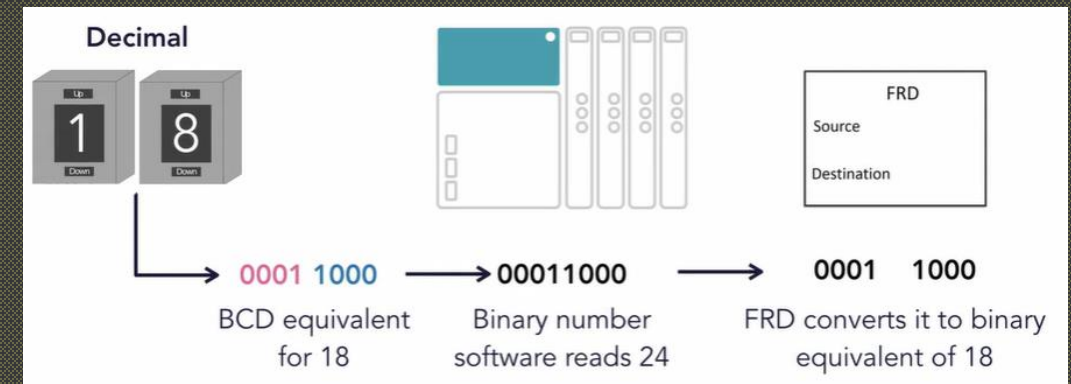
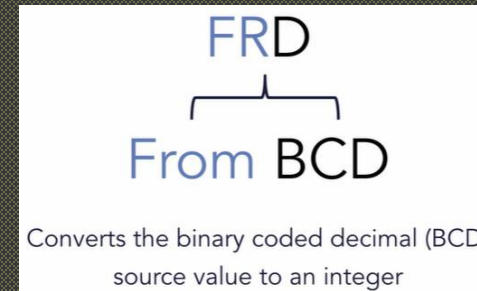
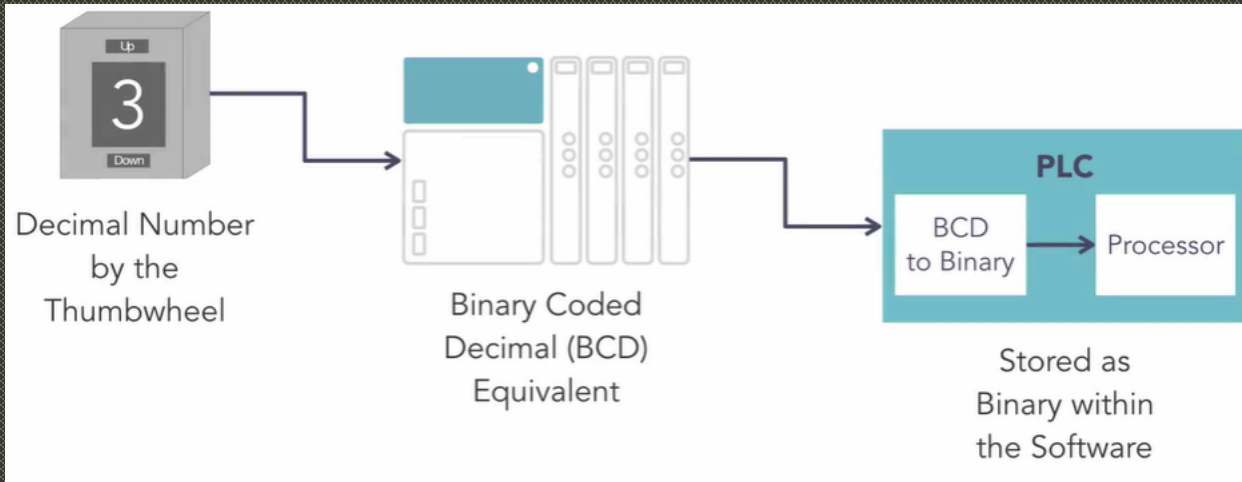


FRD is required for the right interpretation of the conversion.
FRD looks for each 4 bits as one number.



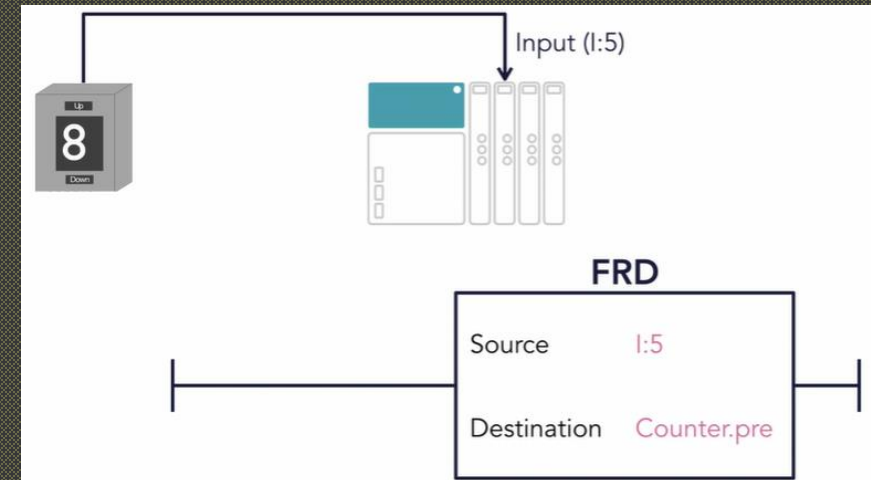
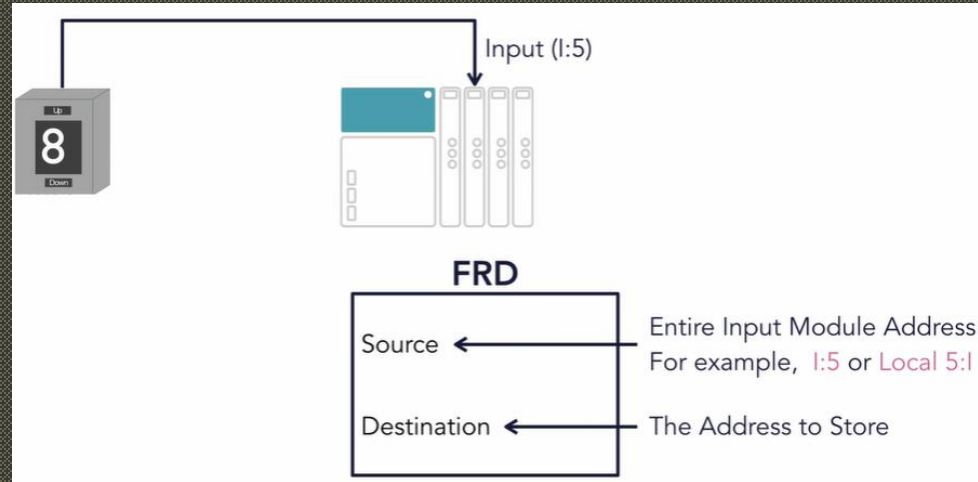
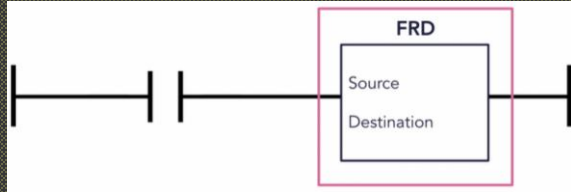
BINARY CODE DECIMAL - FRD

FRD (From BCD) instruction in PLC programming converts a BCD-encoded number into its equivalent integer (binary) format.



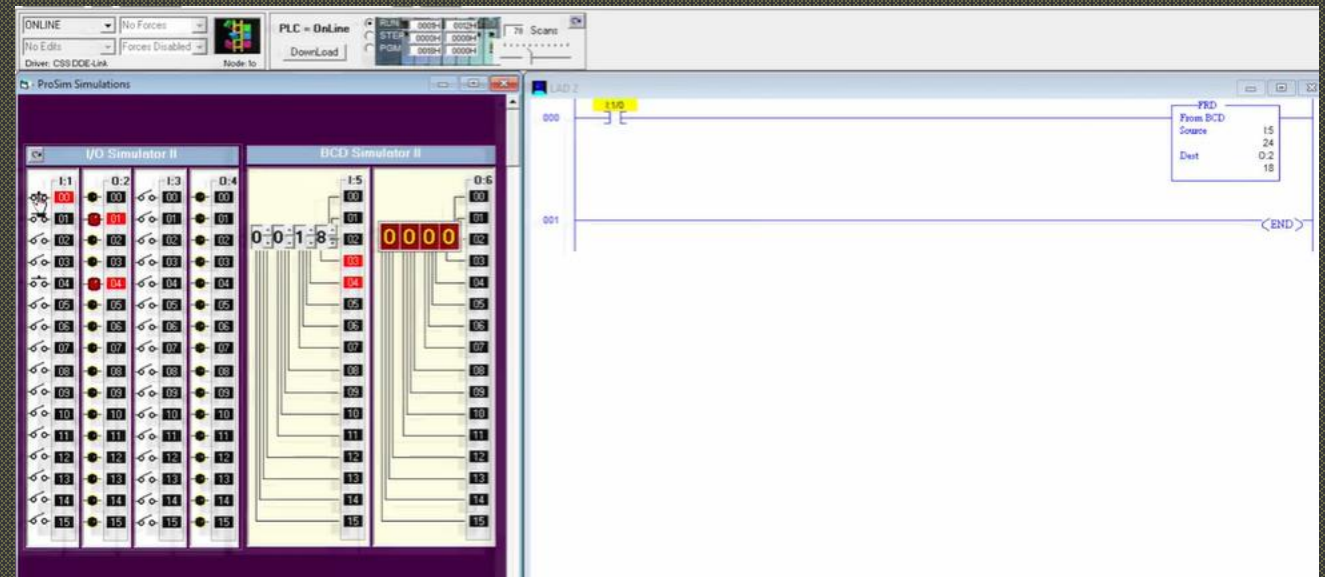
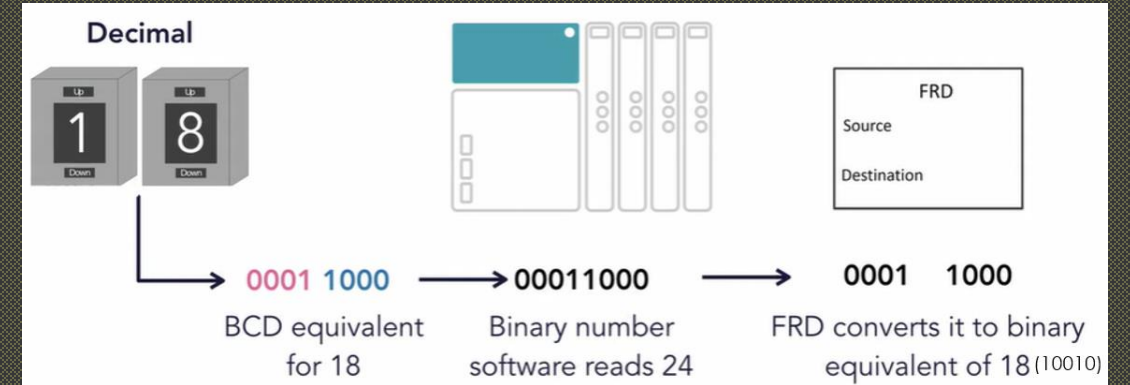
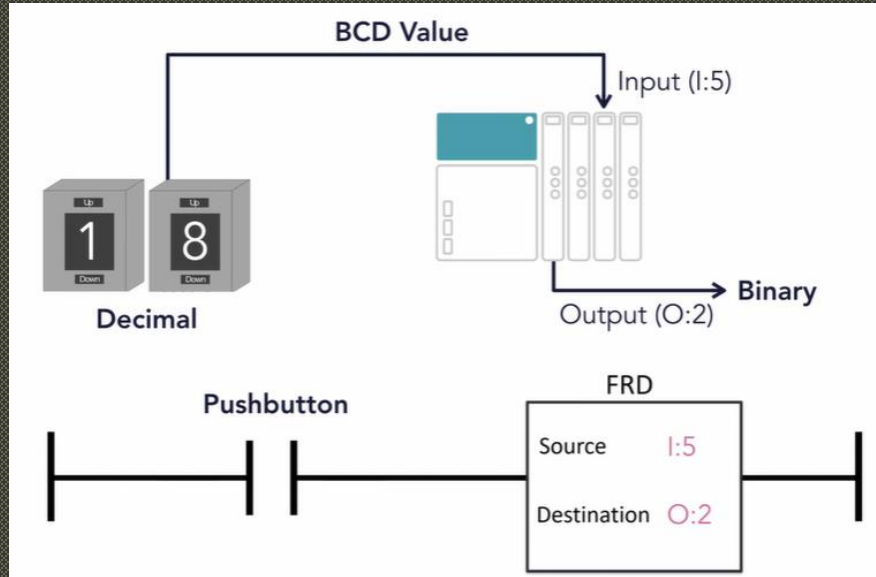
BINARY CODE DECIMAL - FRD

FRD Instruction Parameters



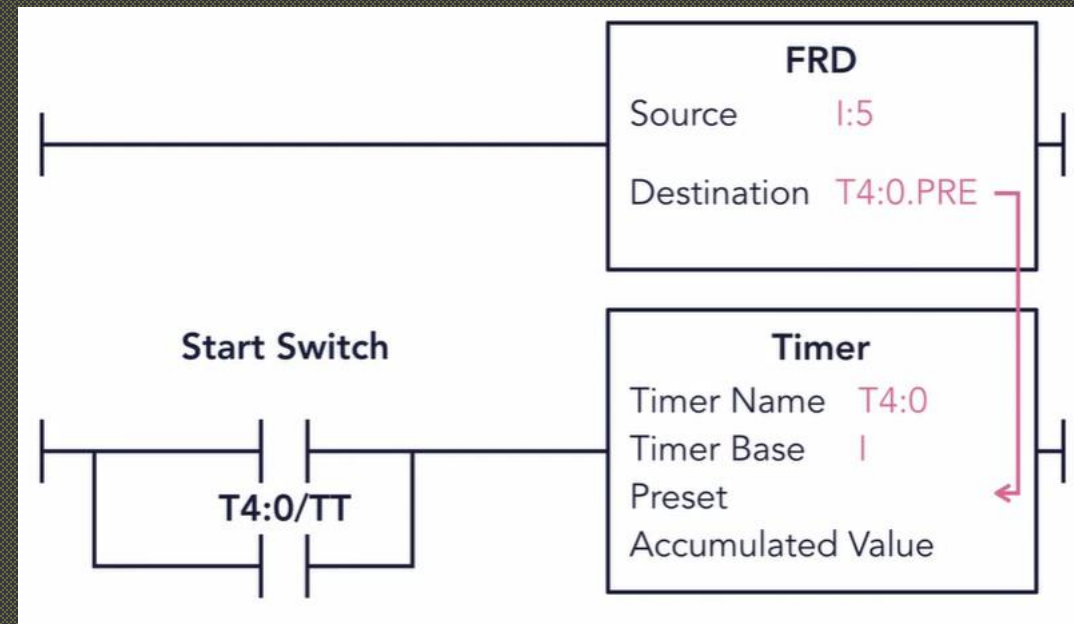
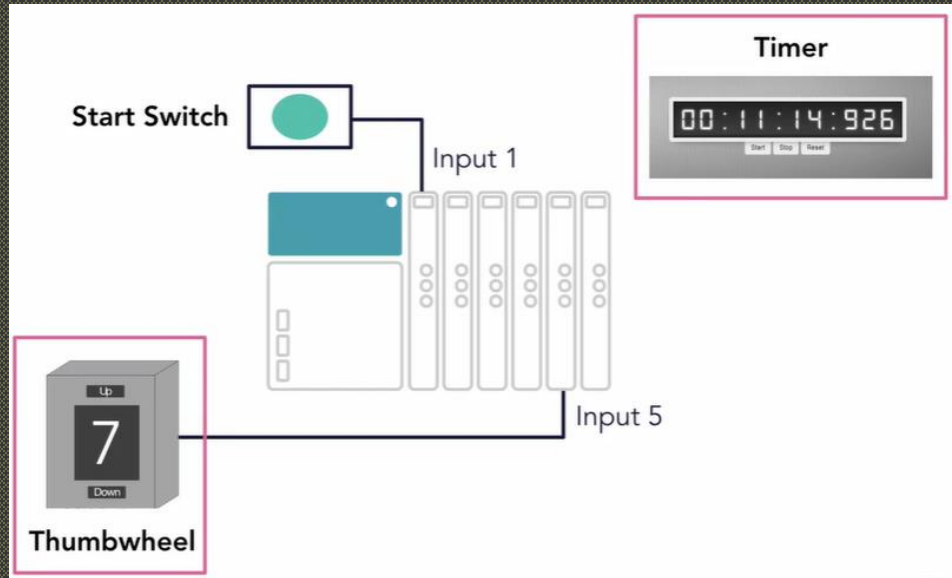
BINARY CODE DECIMAL - FRD

FRD Instruction



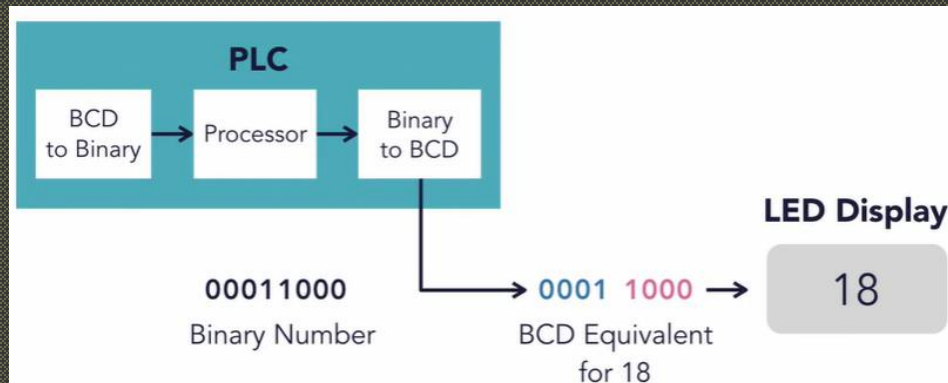
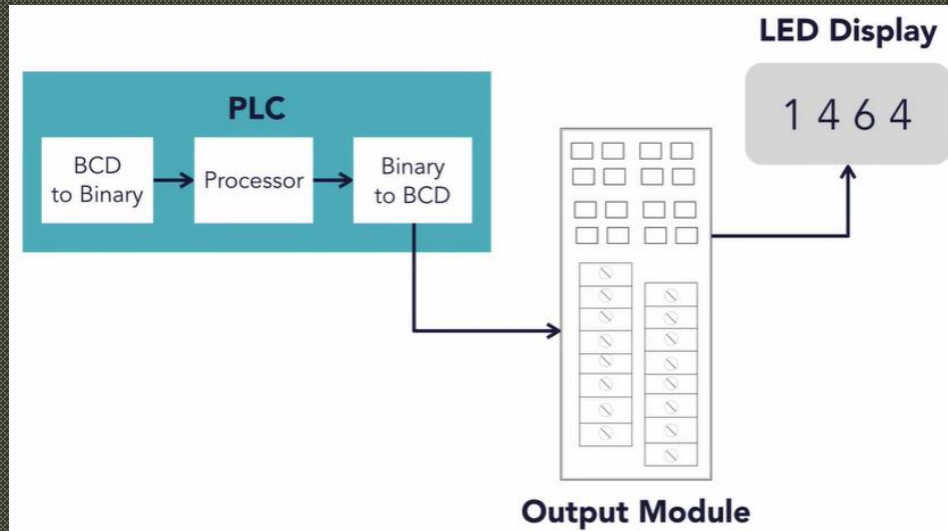
BINARY CODE DECIMAL - FRD

FRD Example – Use of Thumbwheel to preset a timer.

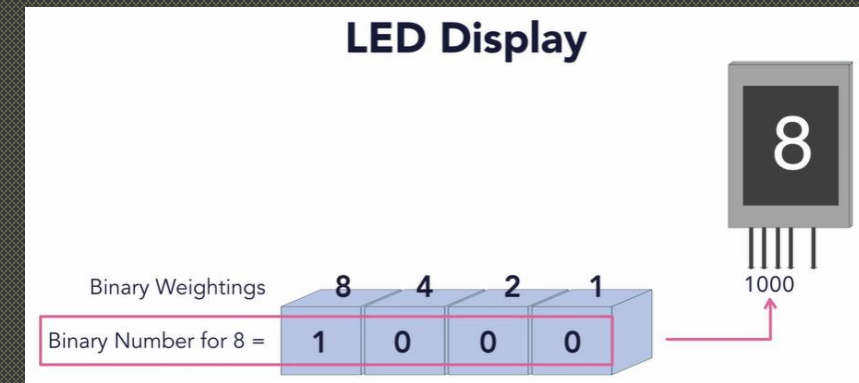
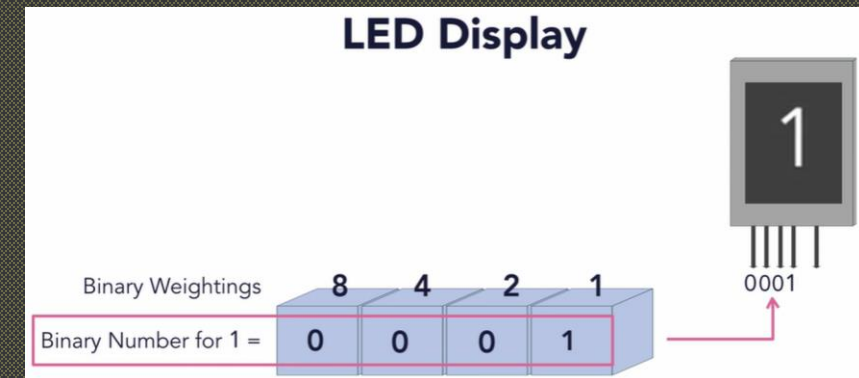


BINARY CODE DECIMAL - LED

LED displays are the example of PLC devices that use the BCD number system. They show a number 0 – 9. The binary number that need to be shown on the LED display gets converted in BCD format and send out to the output module of the PLC.

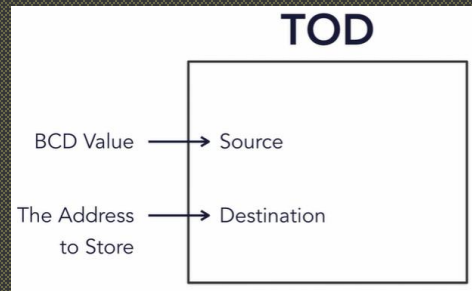
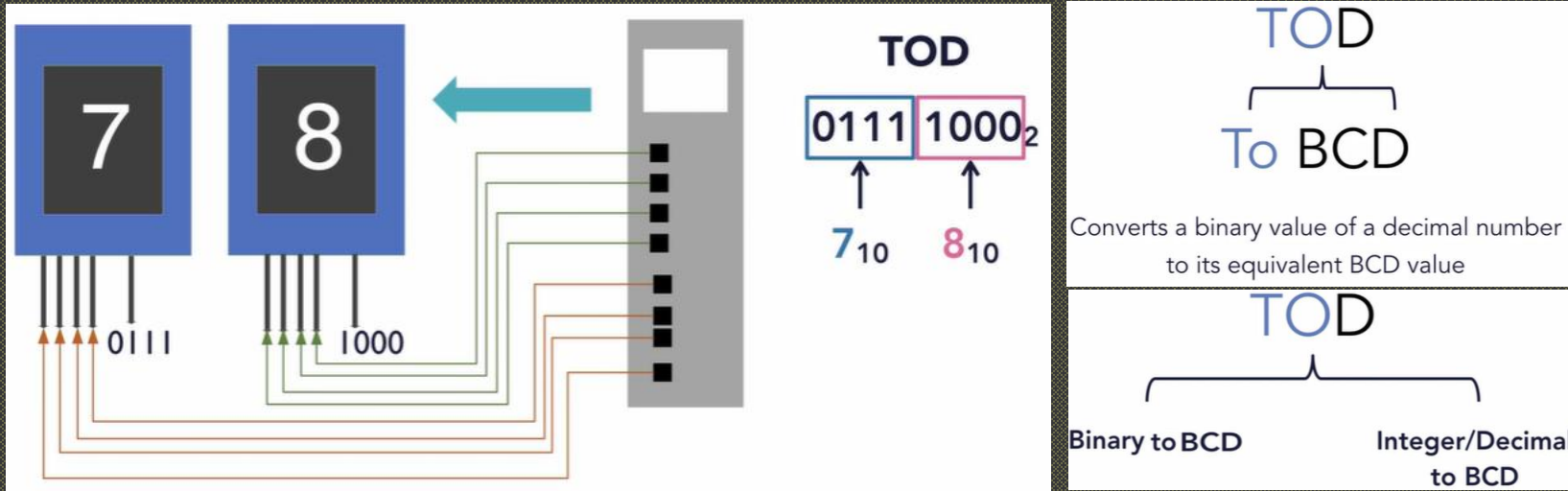


PLC sends out a BCD number equivalent to a binary number and gets displayed as decimal number.



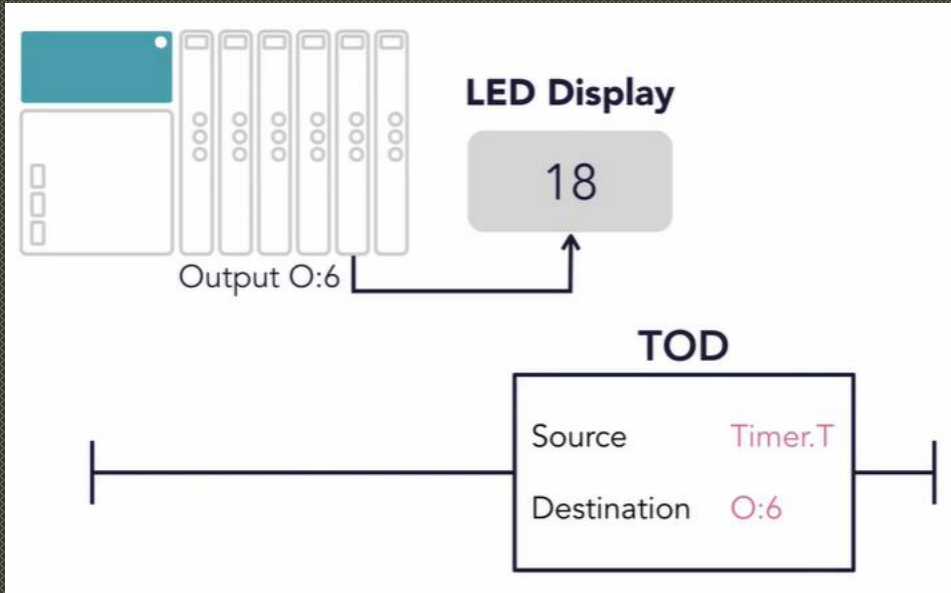
BINARY CODE DECIMAL - TOD

TOD (To BCD) instruction in PLC programming converts a binary value (representing a decimal number) into its equivalent BCD (Binary Coded Decimal) format.



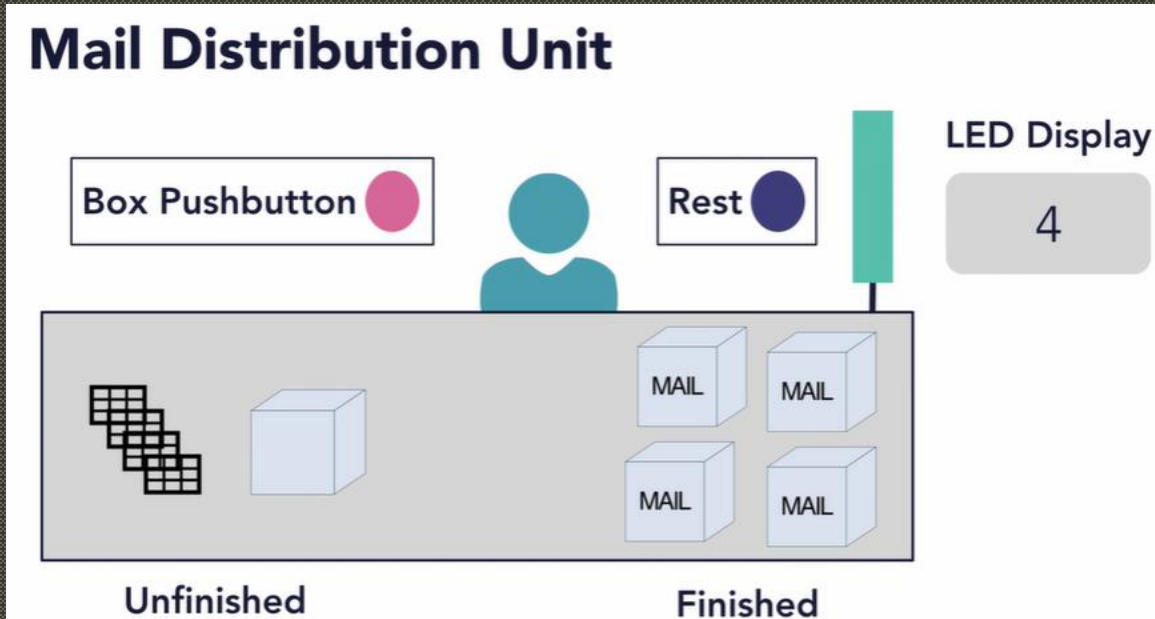
BINARY CODE DECIMAL - TOD

TOD converts a binary value of a decimal number to its equivalent BCD number.

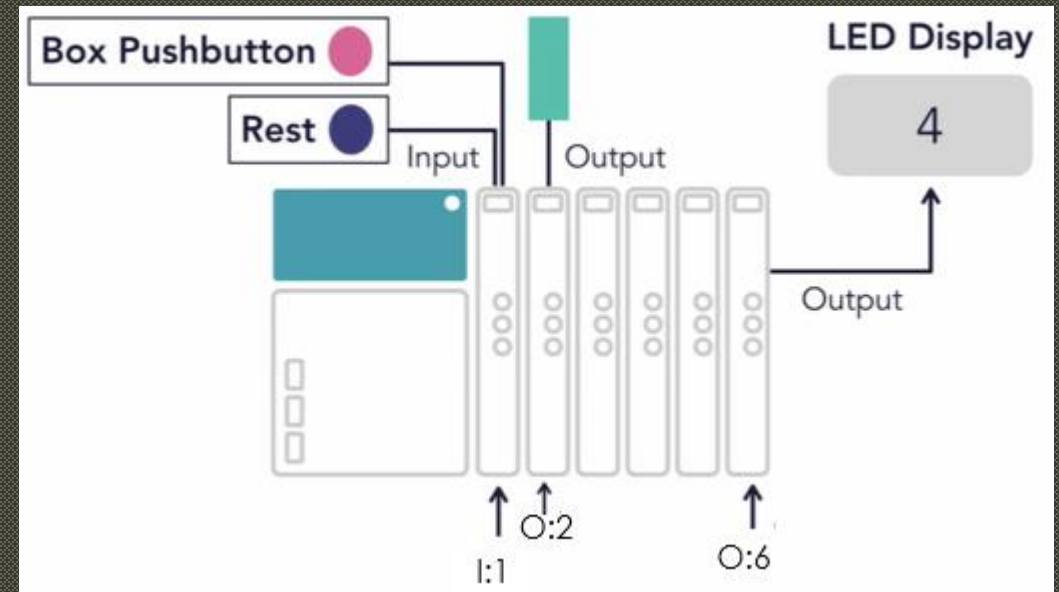


BINARY CODE DECIMAL - TOD

TOD Example.

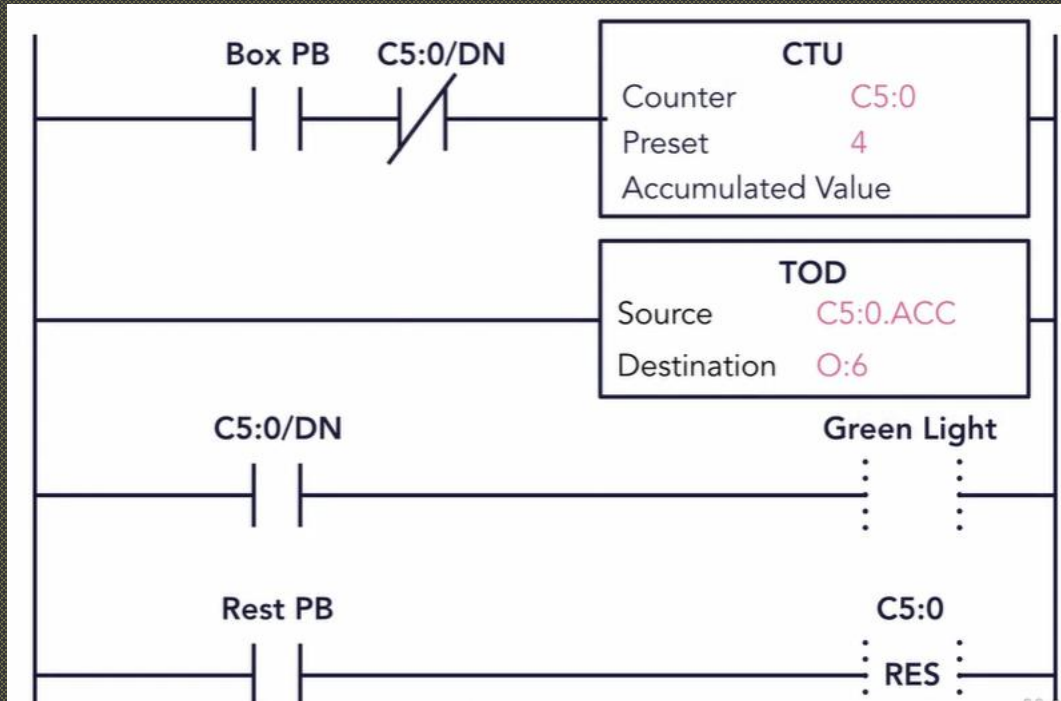


- Count the number of boxes to be shipped
- Every time a full box is completed, the operator presses the Box PB to start counting
- When four boxes are completed, the Green Light turns on
- The Reset PB is used to reset the counter to zero
- Number of boxes should show in the LED Display



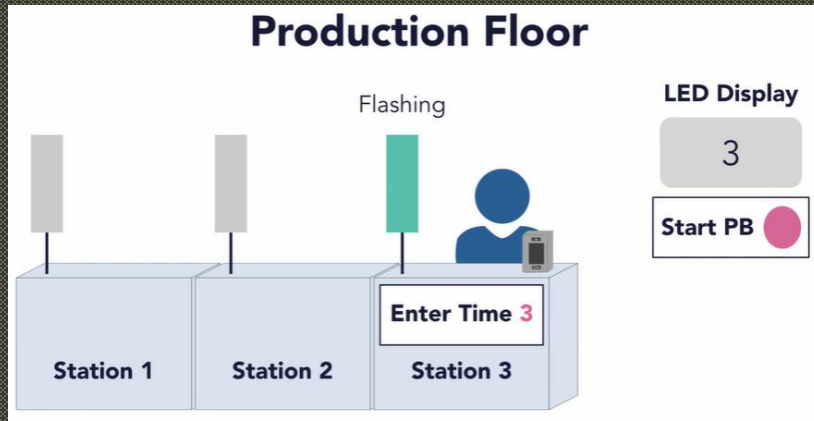
BINARY CODE DECIMAL - TOD

TOD Example.

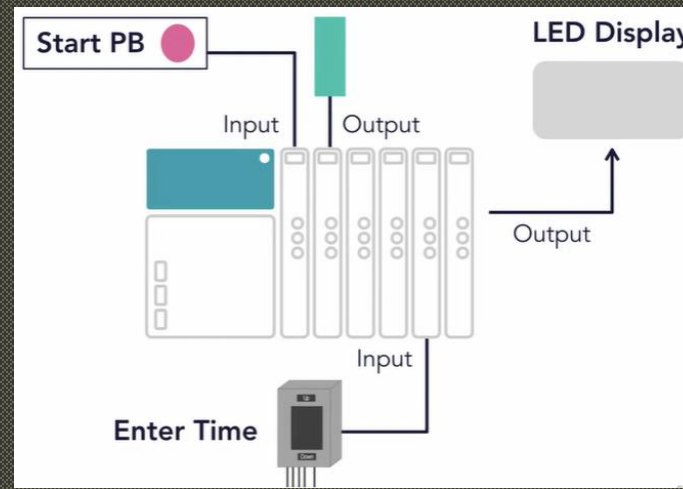


FRD - TOD

FRD & TOD Example.

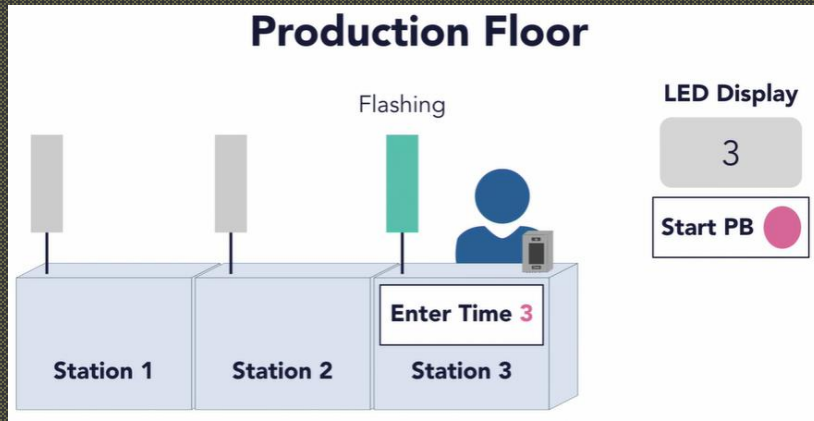


- Three stations with a flashing light each.
- A technician can work in any of the stations to perform a task.
- When a station is used, the technician sets the time for the flashing light to blink, using the thumbwheel.
- The faster the flashing, the shorter the time needed.
- The time is shown in the LED Display.
- To start the process, the technician presses the Start PB.

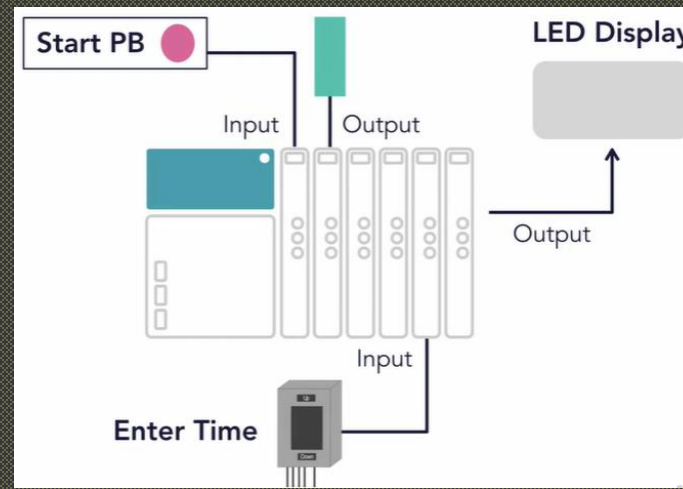


FRD - TOD

FRD & TOD Example.

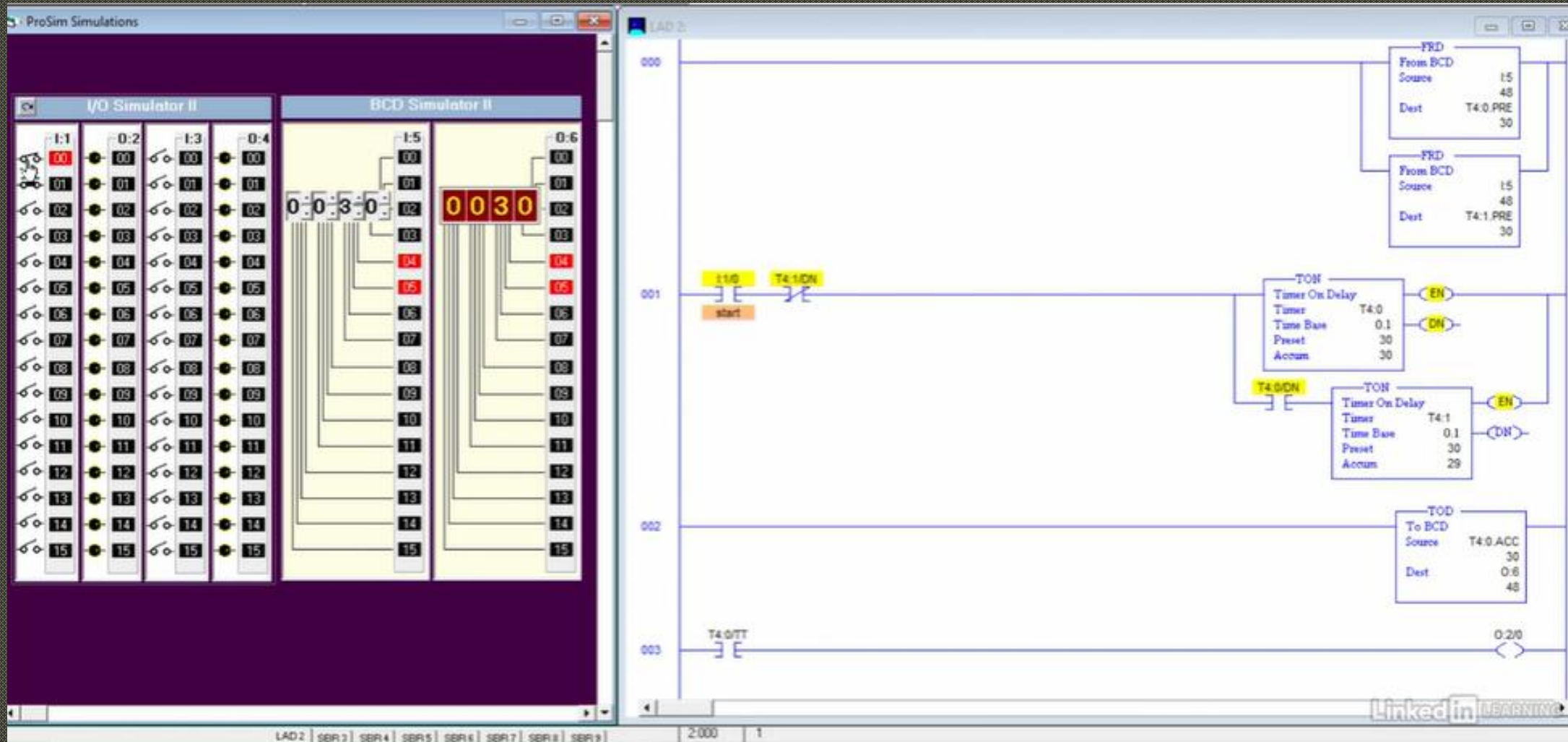


- Three stations with a flashing light each.
- A technician can work in any of the stations to perform a task.
- When a station is used, the technician sets the time for the flashing light to blink, using the thumbwheel.
- The faster the flashing, the shorter the time needed.
- The time is shown in the LED Display.
- To start the process, the technician presses the Start PB.



FRD - TOD

FRD & TOD Example.



FRD - TOD

FRD Example.

Develop a PLC-based PWM system using Ladder Logic to dynamically control the ON/OFF time of an output signal.

1. Requirements:

- Pulse Control: Two timers (T4:0, T4:1) alternate the output's ON/OFF states while maintaining a constant total cycle time.
- User Input: A 2-digit input (0–99) adjusts the ON time; future expansion to 3- or 4-digit inputs is possible.
- BCD to Decimal Conversion: The FRD instruction converts BCD input to decimal before setting the timer's preset.
- Timer Handling:
 - T4:0.PRE stores the converted input value.
 - T4:1.PRE is calculated as $100 - T4:0.PRE$ to maintain cycle consistency.

2. Duty Cycle & Display:

- The ON time is adjustable from 0% (OFF) to 99% (ON).
- A 50% duty cycle is achieved when the input is 50.
- Display the ON/OFF time on an HMI screen.

3. Error Handling & Constraints:

- LIM instruction prevents flickering when input = 0.
- Out-of-range inputs (>99 for 2-digit systems) are rejected.
- Expansion to 3- or 4-digit inputs requires subtracting from 1000 or 10000 instead of 100.

1. Timers Configuration:

- RUNG000 to RUNG003 are used to control Timers T4:0 and T4:1.
- RUNG004 generates the output pulse with a specific width.

2. Input Handling:

- The input address I:5 provides a 2-digit BCD input (0-99).
- This input is converted into an integer and stored in T4:0.PRE (Preset of Timer T4:0).
- The ON time of output O:2/0 is determined by T4:0.PRE.

3. Preset Calculation for Timers:

- The preset value of T4:1 depends on T4:0.PRE.
- Since the total cycle time is 100, the OFF time is calculated as:
- OFF Time = 100 - ON Time; This OFF Time is assigned to T4:1.PRE.

4. Pulse Width and Duty Cycle:

- If the input is 50, both timers will have the same preset value, creating a square wave with a 50% duty cycle.
- By varying the input (0-99), the ON Time of O:2/0 is adjusted, altering the pulse width.

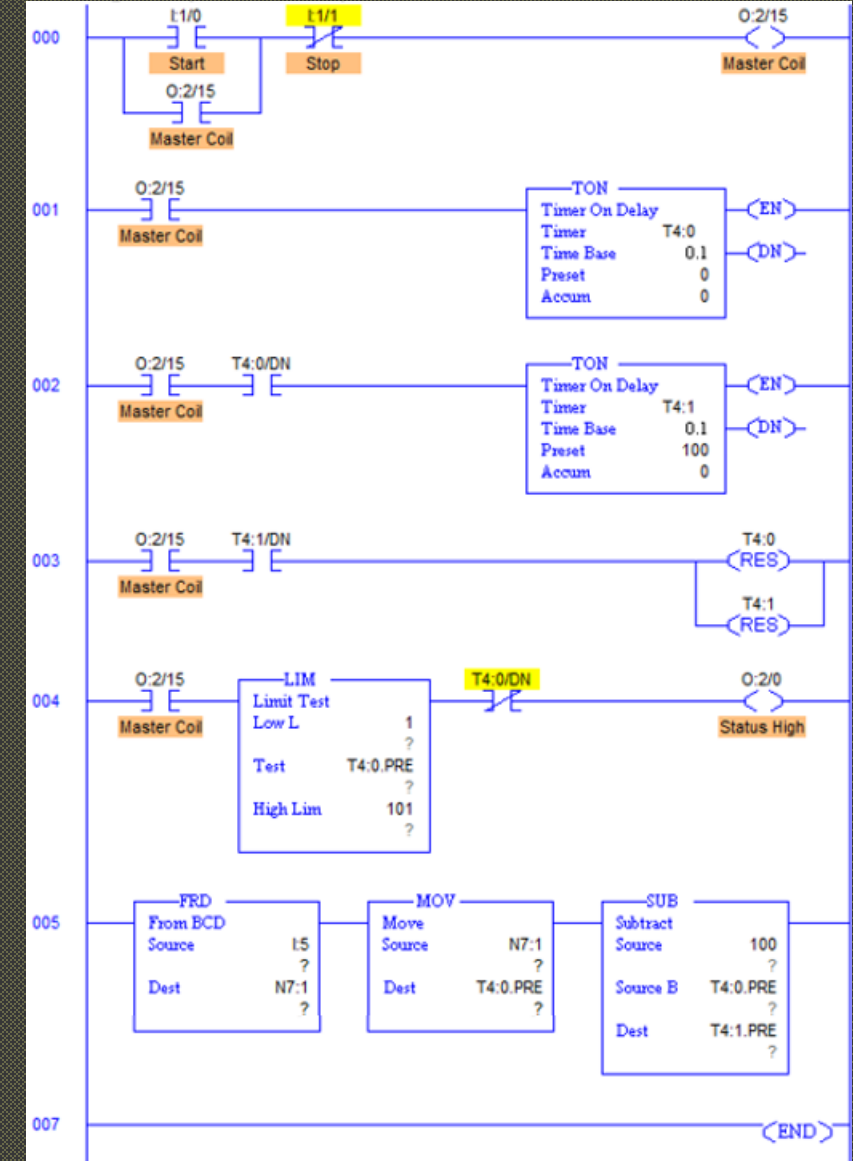
5. Error Handling Using LIM Instruction:

- When I:5 = 0, T4:1 completes OFF Time, causing O:2/0 to blink momentarily.
- To prevent this unwanted blinking, a LIM (Limit Test) instruction is used to ensure valid input values.

6. Extending to Larger Inputs:

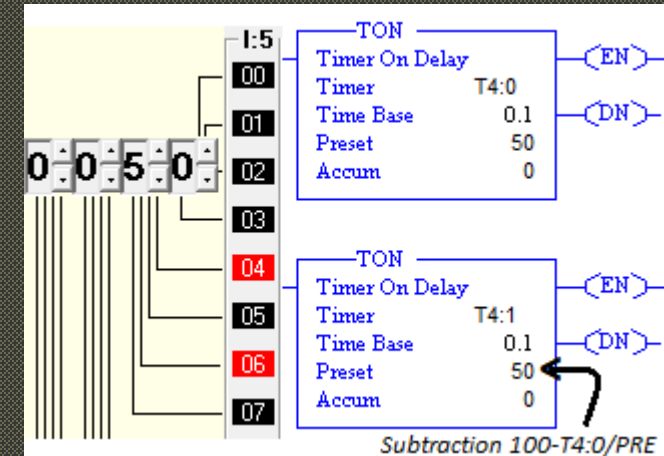
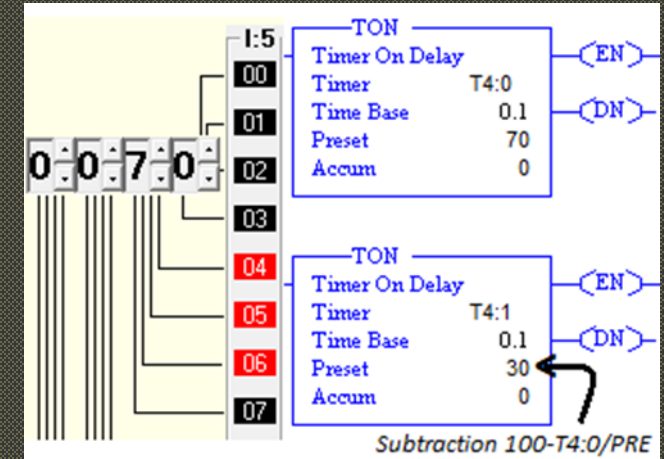
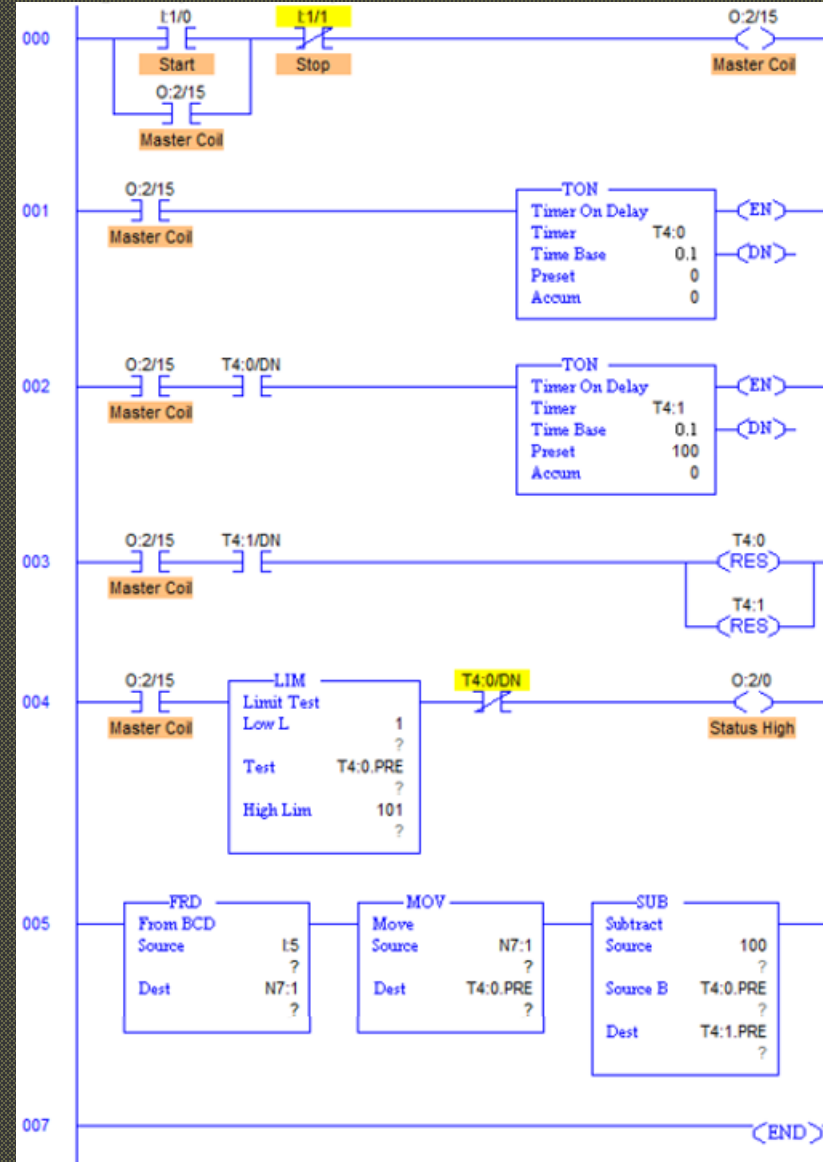
- For 3-digit (0-999) or 4-digit (0-9999) inputs, the total cycle time must be adjusted:
 - 3-digit input: Subtract from 1000.
 - 4-digit input: Subtract from 10,000.

This setup enables precise control of pulse width modulation by dynamically adjusting the ON/OFF Time using BCD input values.



FRD - TOD

FRD Example.



FRD - TOD

TOD Example.

Implement continuous level monitoring of a 10m (1000 cm) tank using a PLC, for the following system:

1.Sensor Selection:

- Use an Ultrasonic, Radar, Capacitive, or Pressure gauge level sensor with sensor output: 0 – 20 mA analog signal proportional to the tank level.

2.Analog Input Module:

- Converts the 0 – 20 mA signal into a digital (hex) value using a 16-bit ADC.

3.PLC Logic:

- Convert the raw ADC value into a meaningful tank level (cm).
- Display the converted value on an HMI screen or 7-segment display.

FRD - TOD

TOD Example.

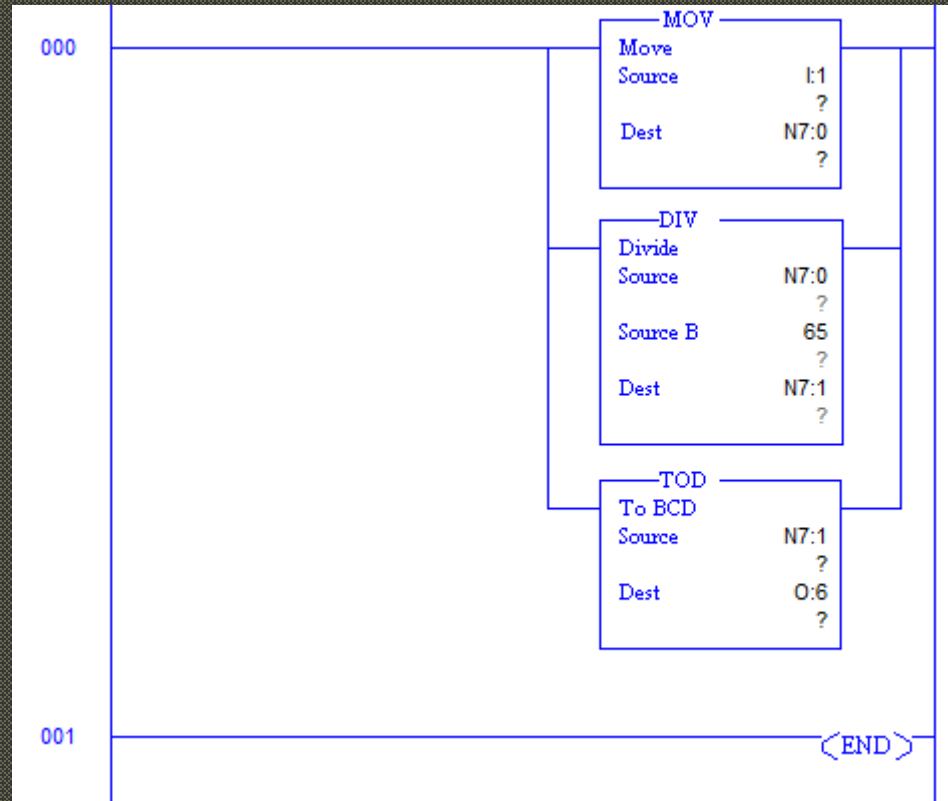
PLC Logic for Tank Level Display (0 – 20 mA to 1000 cm)

1. Analog Conversion in PLC: The 0 – 20 mA signal is converted into 0 – 65,000 counts (approx.) in PLC. Since this corresponds to 0 – 1000 cm, each 1 cm rise in tank level results in an increase of 65 counts.
2. PLC Memory Mapping:
 - I:1 → Analog Input (Raw Data from Sensor);
 - N7:0 → Stores the raw sensor data (continuously updated);
 - N7:1 → Stores the calculated tank level in centimeters;
 - O:6 → Output module for display (BCD format).
3. PLC Program:
 - MOV instruction is used to continuously transfer the input data from I:1 to N7:0;
 - DIV (Divide) instruction: $N7:1 = N7:0 / 65$; This converts the ADC counts into centimeters.
 - TOD (Integer to BCD) instruction: $O:6 = TOD(N7:1)$; Converts the tank level from integer format to BCD format.
4. Accuracy Calculation: $(65535 - 65000) \times 100 / 65535 = \pm 0.8\%$; This means the error in measurement is $\pm 0.8\%$.

Binary Weighting															
2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	1	0	0	0	1	1	1	0	1	1	0	1

FRD - TOD

TOD Example.



PROGRAMMABLE LOGIC CONTROLLERS

STUDIO 5000, FRD & TOD INSTRUCTIONS

Frank D. Petruzella Programmable Logic Controllers, 6th edition – Chapter 15

<https://www.linkedin.com/learning/learning-plc-ladder-logic-2-diving-deeper/introduction-to-binary-coded-decimal>

PROGRAMMABLE LOGIC CONTROLLERS

MENG 3500

Thank you!

Discussions?