NITNIX
AUTOMATION SOLUTIONS

DELTA PLC

STUDY MATERIAL

DVP20EX2 · WPLSoft V2.42 · Ladder Diagram Programming

Table of Contents

1. Hardware Specification . . . 3
2. Basics . . . 5
3. Timer . . . 10
4. Counter . . . 13
5. Memory / Data Register . . . 15
6. Comparison . . . 18
7. Positive Trigger & Negative Trigger . . . 20
8. Set & Reset . . . 22
9. Move . . . 23
10. Arithmetic Functions . . . 24
11. Jump . . . 27
12. Call Function / Subroutine . . . 28
13. Scaling . . . 30
14. Real Time Clock (RTC) . . . 33
Quick Reference . . . 35

01 Hardware Specification

Meet the controller used throughout this course: the Delta DVP20EX2.

Delta offers a wide range of DVP-series PLCs. The main families are: **DVP-EH3**, **DVP-PM**, **DVP-10PM**, **DVP-ES2 / EX2**, **DVP-SV2**, **DVP-SE**, **DVP-EC3**, **DVP-SX2**, **DVP-SA2**, **DVP-SS2**, **DVP-MC** and the **DVP-Slim** modules. This study material uses the **DVP20EX2**.



Figure 1.1 – Delta DVP20EX2 PLC (8 DI / 6 DO / 4 AI / 2 AO)

Key specifications

Parameter	Details
PLC model	DVP20EX2 (DVP-EX2 series)
Maximum I/O points	256 inputs + 16 outputs, *or* 256 outputs + 16 inputs
Built-in analog	12-bit, 4 analog inputs (4AD) and 2 analog outputs (2DA)
Program capacity	16k steps
Communication ports	3 built-in ports: COM1 = RS-232, COM2 and COM3 = RS-485. All can work independently (Master / Slave).
Programming software	WPLSoft V2.42
Programming language	Ladder Diagram

Communication details

Port	Type	Used for
`COM1`	RS-232	PC to PLC (programming and download)
`COM2`	RS-485	PLC to HMI

Addressing details

In **DVP20EX2**, the number **20** is the total number of I/O points: 8 digital inputs + 6 digital outputs + 4 analog inputs + 2 analog outputs = 20.

I/O type	Addresses	Quantity
Digital inputs	`X0, X1, X2 … X7`	8
Digital outputs	`Y0, Y1, Y2 … Y5`	6
Analog inputs	`D1110, D1111, D1112, D1113`	4
Analog outputs	`D1116, D1117`	2

REMEMBER

Inputs **X** and outputs **Y** are numbered in **octal**: X0–X7, then X10–X17 (there is no X8 or X9). The addressable range is X0–X377 and Y0–Y377.

02 Basics

Create your first project in WPLSoft, write a start/stop program, download it and run it on the PLC.

A. Creating a new project

1	Open **WPLSoft** (Delta's PLC programming software).
2	Click the **File** menu and choose **New**.
3	Enter any **Program Title**, then select the **EX2** series (we are using the EX2 series) and click **OK**.

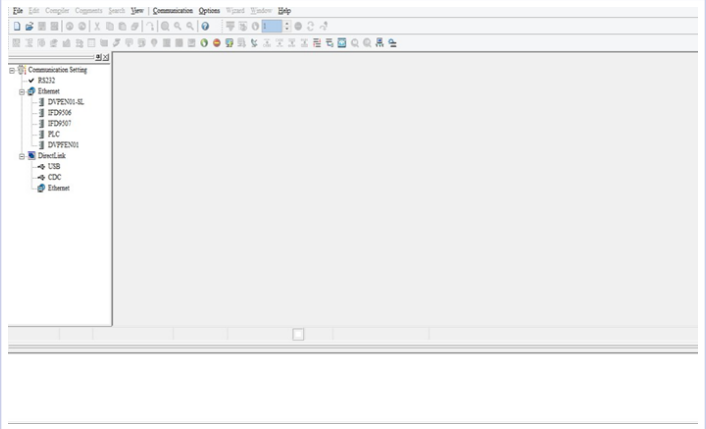


Figure 2.1 – WPLSoft main window

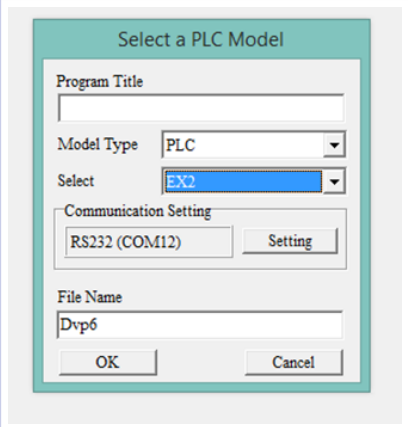


Figure 2.2 – Select a PLC Model dialog – choose EX2

B. Input instructions

Contact	Instruction	Meaning
Normally open (NO)	LD – Load A contact	Starts a rung from the left bus (or a new contact circuit). Saves the present contents and loads the contact status into the accumulator register.
Normally closed (NC)	LDI – Load B contact	Same as LD but loads the **inverse** of the contact status.

Operands that can be used: **X**, **Y**, **M**, **S**, **T**, **C**.

C. Addresses of inputs and outputs

Device	Range	Capacity
External inputs	X0 – X377 (octal)	256 points max.
External outputs	Y0 – Y377 (octal)	256 points max.
Total		256 + 16 I/O

D. Writing a simple bit-logic program (Start / Stop)

1. Click the **normally open contact**. Give the input address (for example **X0** … **X7**). You may type a name in the **Comment** box (for example *start*).
2. Repeat for **X1** using a **normally closed contact** (comment *stop*).
3. Click the **output coil** and give the output address (for example **Y0** … **Y5**, comment *coil*).
4. Add a normally open contact of **Y0** in parallel with **X0** – this holds the output ON after Start is released (seal-in).

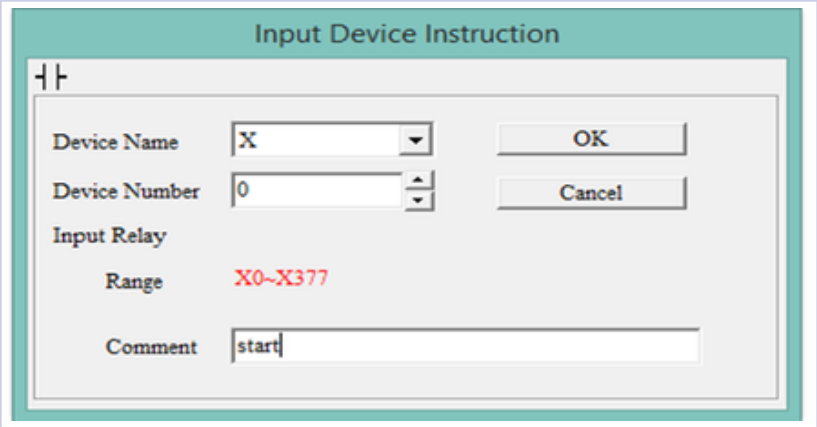


Figure 2.3 – Input Device Instruction dialog – contact

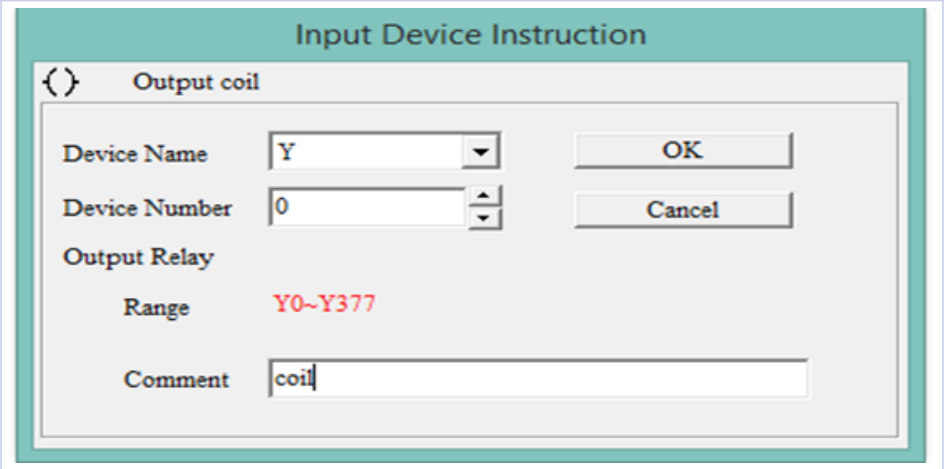
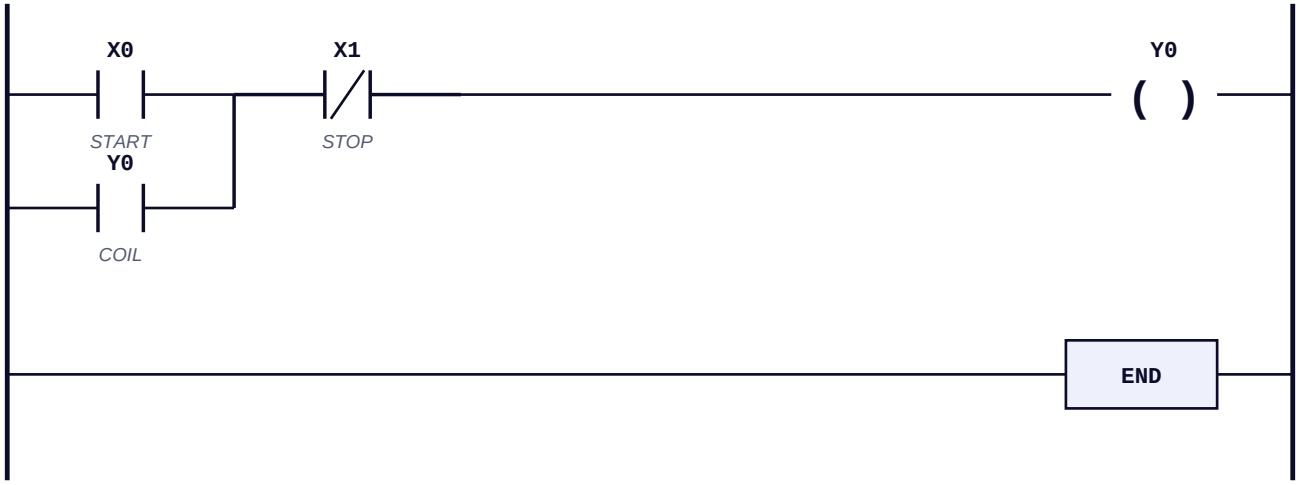


Figure 2.4 – Input Device Instruction dialog – output coil

The finished program looks like this:



Inputs **X0** = Start and **X1** = Stop are given from the PLC input terminals; **Y0** is the output coil.

E. Downloading and running the program

1	Go to **online mode** (the button marked with the red box in Figure 2.5).
2	In online mode, click **PC to PLC** communication to download the program to the PLC (the green indication), then click **OK** in the Transfer Setup window.
3	When asked *"Recover the executing status before the program was downloaded?"* click **Yes**.
4	After downloading, the program window changes to **run mode**. Contacts and coils turn green when they are ON.

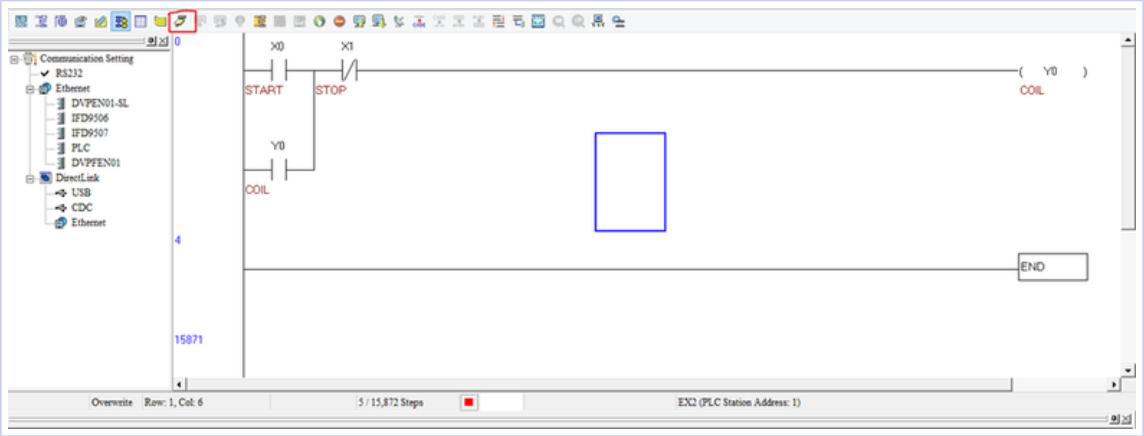


Figure 2.5 – Online mode button (red box)

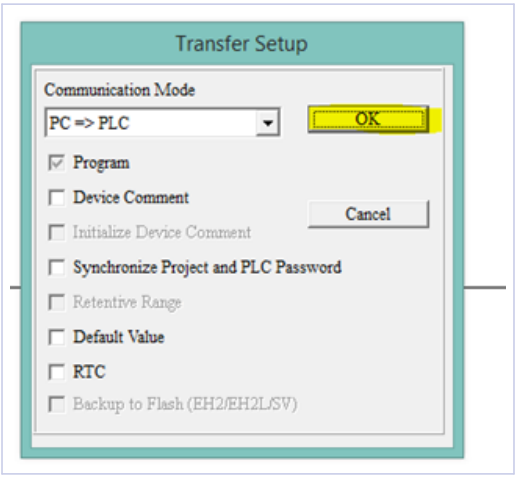


Figure 2.6 – Transfer Setup – Communication Mode: PC => PLC

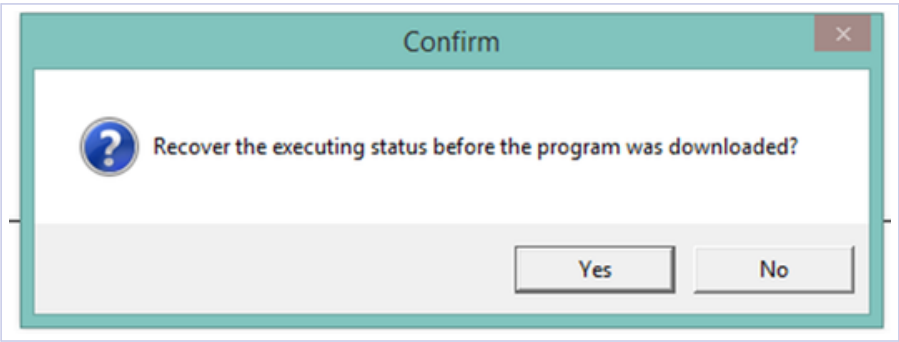


Figure 2.7 – Confirmation message after download – click Yes

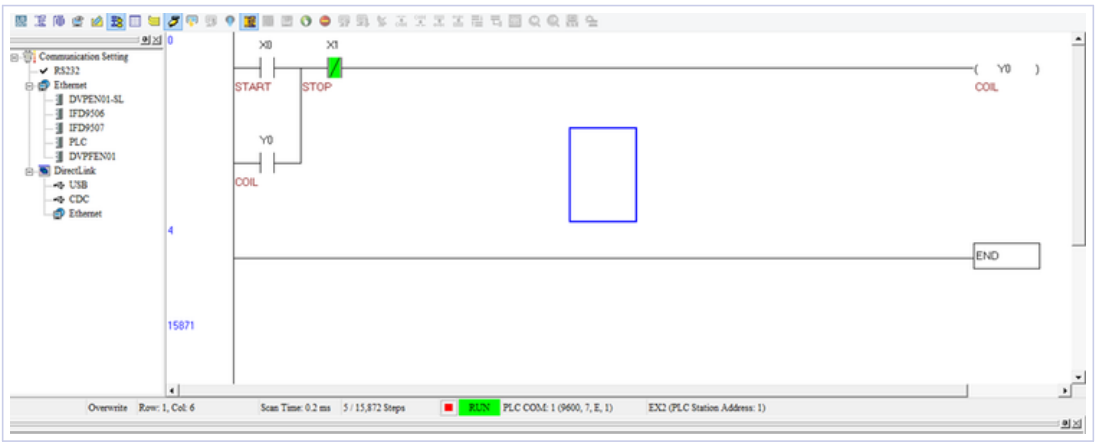


Figure 2.8 – Program window in run mode

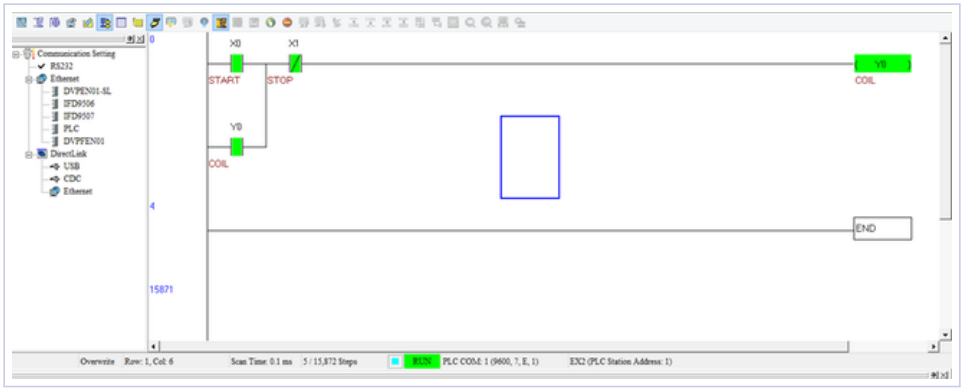


Figure 2.9 – Input X0 (Start) ON – coil Y0 energised and held by its own contact

IMPORTANT

A program **cannot be edited while in run mode**. To edit it, first deselect run mode (go back to offline / edit mode).

TIP

In real panels a Stop push button is usually wired **normally closed**. In that case use a *normally open* contact of that input in the ladder logic. Here a normally closed contact of **X1** is used to follow the original example.

03 Timer

Measure time and perform an action after a delay – for example, turn a motor ON after 5 seconds.

A **timer** is a special instruction that measures time and performs an operation after a set delay. Timers control events that must happen after a given duration – turning ON a motor after 5 seconds, or turning OFF a light after a delay.

Delta PLCs provide two types of timers:

- **On-delay timer** – resets when its input turns OFF.
- **Retentive timer** – keeps its elapsed time when the input turns OFF.

On-delay timer

- If the timer input stays continuously ON, the current value counts up from 0.
- When the current value ≥ the preset value, the timer output bit (the T contact of the same number) turns ON.
- When the input turns OFF the current value is cleared and the output turns OFF.

Resolution	Addresses	Points	Note
100 ms	`T0 – T63`	64	
10 ms	`T64 – T126`	63	Runs at 10 ms when M1028 = ON; at 100 ms when M1028 = OFF
1 ms	`T127`	1	
Total		128	

Syntax

```
TMR   S1   S2
  S1 : timer number   (T)
  S2 : set value      (K, H, D)
```

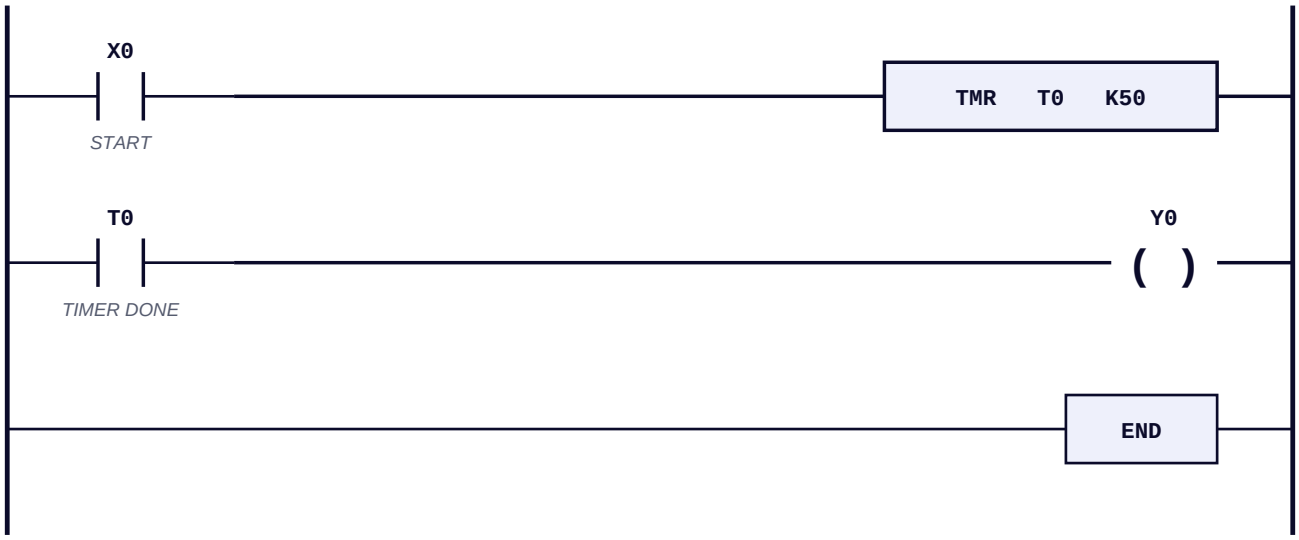
Timer time = **preset value × timer resolution**.

Example

In WPLSoft press **F6** (function block) and type **`TMR T0 K50`**.

Part	Meaning
`TMR`	Timer instruction
`T0`	Timer address (100 ms based timer)
`K50`	Preset value of the timer

T0 is a 100 ms timer, so 50 × 100 ms = 5000 ms. After **5 seconds** the T0 contact turns ON and drives Y0.



Retentive timer

A **retentive timer** stores (retains) its elapsed time even if the input turns OFF or power is interrupted. In simple terms: the timer *pauses* when the input turns OFF and *resumes from the same value* when the input turns ON again.

Resolution	Addresses (EX2 PLC)
100 ms	`T250 – T255`
10 ms	`T240 – T245`
1 ms	`T246 – T249`

Why is a reset necessary?

- A retentive timer does **not** reset automatically when the input turns OFF.
- It stores the elapsed time and continues from the same value when the input turns ON again.
- So a manual reset is necessary to bring it back to zero – use the **`RST`** instruction (for example **`RST T250`**).
- The timer value becomes zero **only** when the reset instruction is given.



Figure 3.1 – Retentive timer T250 (preset K60) holding its value; X2 resets it with RST T250

KEY POINTS

- Time = preset × resolution (T0, K50 → 50 × 100 ms = 5 s).
- On-delay timers (T0–T127) clear when the input goes OFF.
- Retentive timers (T240–T255) hold their value and must be cleared with RST.

04 Counter

Count input pulses and act when a preset number is reached.

A **counter** is a function block that increments (or decrements) a value every time a signal pulse is received. When the count reaches the preset value, the counter output turns ON. This can be used to stop a motor, turn ON a light, or reset a process.

Counter types and addresses

Counter type	Addresses	Points	Retention
16-bit up	C0 – C111	112	General
16-bit up	C112 – C127	16	Latched
16-bit up	C128 – C199	72	General
32-bit up/down	C200 – C223	24	General
32-bit up/down	C224 – C231	8	Latched
32-bit high-speed (software) – 1-phase 1-input	C235 – C242	8	Latched
32-bit high-speed (software) – 2-phase 2-input	C232 – C234	3	Latched
32-bit high-speed (hardware) – 1-phase 1-input	C243 – C244	2	Latched
32-bit high-speed (hardware) – 1-phase 2-input	C245 – C250	6	Latched
32-bit high-speed (hardware) – 2-phase 2-input	C251 – C254	4	Latched

Total: 232 points (16-bit and 32-bit) and 23 high-speed points. *General* counters lose their count at power-off; *latched* counters keep it.

Syntax

```
CNT   S1   S2
  S1 : 16-bit counter number   (C)
  S2 : set value               (K, D)
```

- The counter counts when the counter coil is driven from **OFF to ON** (one count per pulse).
- When the count equals the set value, the counter NO contact turns ON (continuity) and the NC contact turns OFF.
- Further pulses after the set value is reached do not change the contacts or the count. To count again, clear it with **RST**.

Example

1. Press **F6** and type **`CNT C0 K5`**. After 5 counts the output turns ON. (**CNT** = counter instruction, **C0** = counter address, **K5** = preset value.)
2. To reset the counter, press **F6** and type **`RST C0`**. When **`X1`** turns ON, the current value of C0 becomes zero.

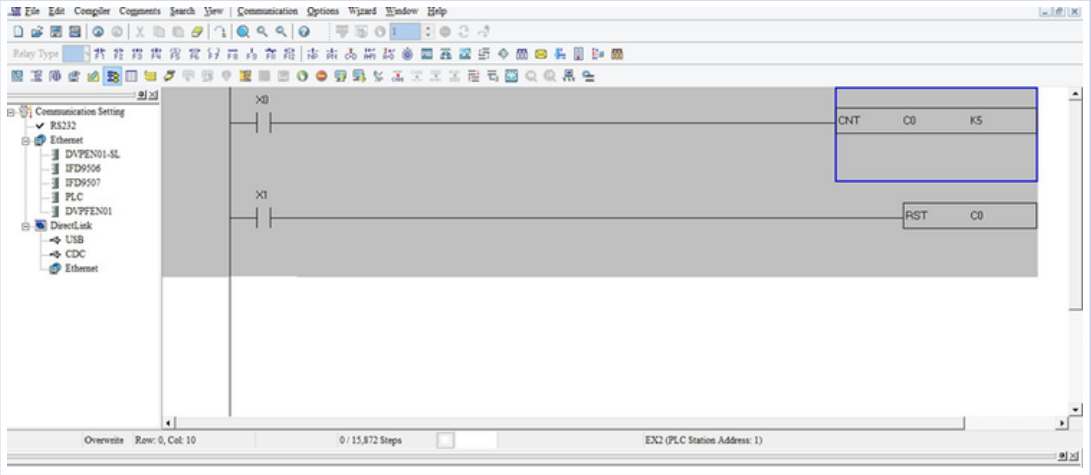


Figure 4.1 – X0 pulses are counted by C0 (preset K5); X1 resets C0

KEY POINTS

- CNT counts OFF→ON pulses; it needs a fresh pulse for every count.
- After the preset is reached the count freezes until RST is executed.
- Use C0–C199 for ordinary 16-bit counting; use 32-bit counters for large values.

05 Memory / Data Register

Where the PLC stores bits and numbers, and how big each data type is.

Three kinds of memory are used in everyday programming:

Type	Addresses	Size / usage
Bool (bit)	`M0 – M4095`	One bit: 0 or 1 (internal relay)
Integer / Word	`D0 – D9999`	16-bit data register
Dword / DINT	`D0, D1 … (pairs)`	32-bit value held in two consecutive registers

Memory (bit) instructions store, retain and control ON/OFF states inside the program even after the input condition changes. They are essential for **latching**, **sequencing** and **interlocking**.

Data register (D)

Class	Addresses	Words
General	`D0 – D407`	408
General	`D600 – D999`	400
General	`D3920 – D9999`	6080
Latched	`D408 – D599`	192
Latched	`D2000 – D3919`	1920
Special	`D1000 – D1999`	1000 (some are latched)
Special modules	`D9900 – D9999`	100
Index	`E0 – E7, F0 – F7`	16

Total: 10000 points.

Auxiliary relay (M)

Class	Addresses	Points
General	`M0 – M511`	512
General	`M768 – M999`	232
General	`M2000 – M2047`	48
Latched	`M512 – M767`	256
Latched	`M2048 – M4095`	2048
Special	`M1000 – M1999`	1000 (some are latched)

Total: 4096 points. *General* devices are cleared at power-off; *latched* devices keep their state.

Memory space

Unit	Size	Range
Bit	1 bit	0 and 1
Nibble	4 bits	0 to 15
Byte	8 bits	0 to 255 (unsigned) · −128 to 127 (signed)
Word / Integer	16 bits (2 bytes)	0 to 65,535 (unsigned) · −32,768 to 32,767 (signed)

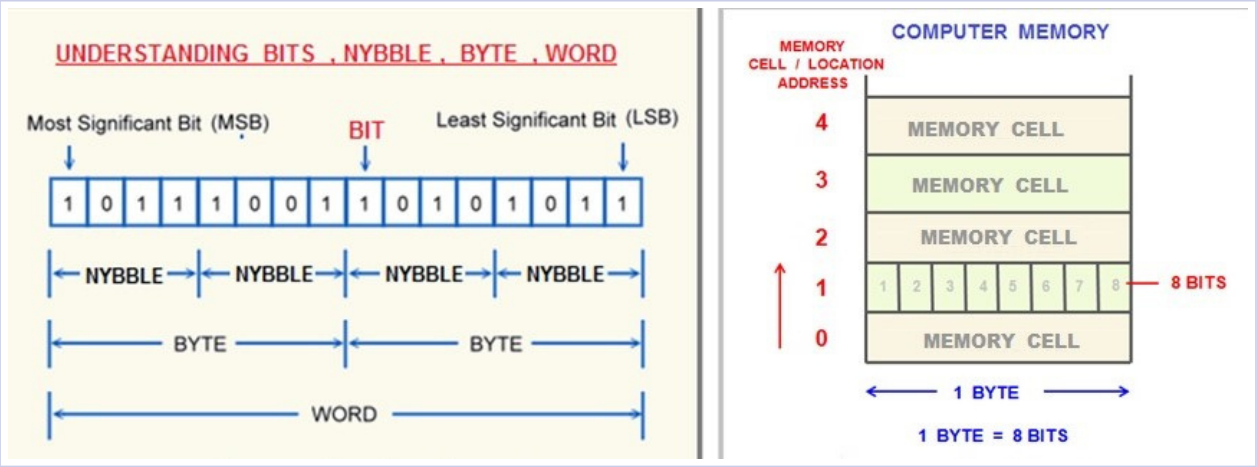


Figure 5.1 – Bits, nibbles, bytes and words

Data types

Data type	Size	Range	Typical uses
Integer (INT)	16-bit	−32,768 to +32,767	Counting values (product count), timer / counter presets, simple arithmetic (ADD, SUB, MUL)
Double Integer (DINT)	32-bit	−2,147,483,648 to +2,147,483,647	Large counting values, production totals
Float (REAL)	32-bit	Decimal numbers	Temperature, pressure, speed calculation, analog signal processing

Example – using an internal relay

Input **X0** energises the internal relay **M0**. The **M0** contact starts timer **T0** (preset K90 = 9 s), and when **T0** finishes it turns ON **Y0**.

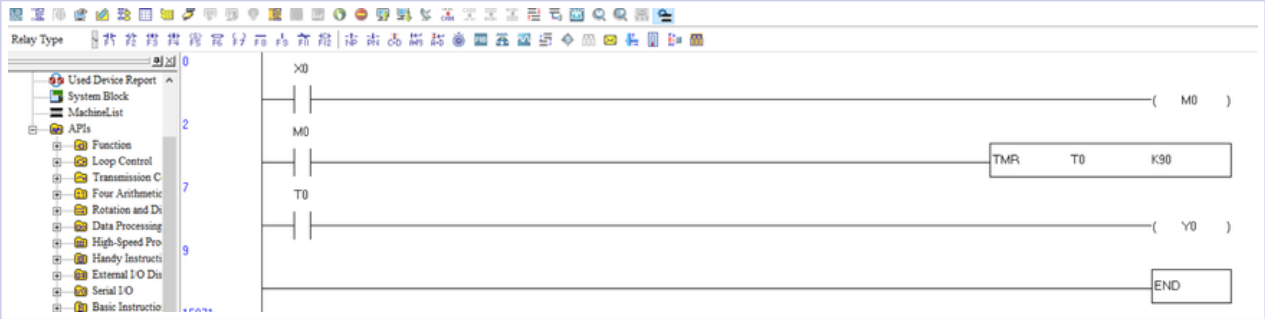


Figure 5.2 – Internal relay M0 starts timer T0 (K90), which drives Y0

KEY POINTS

- M = bit memory (0/1); D = 16-bit word memory; two D registers together = 32-bit.
- Latched ranges keep their value through power loss; general ranges do not.
- A signed 16-bit value runs from −32,768 to 32,767 – use 32-bit instructions for anything larger.

06 Comparison

Compare two values and turn a contact ON or OFF according to the result.

A **comparator** compares two data values (registers or constants). It checks conditions such as greater than, less than or equal to, and passes power (turns the output ON) or blocks it (OFF) based on the result.

Comparator	Symbol	Checks whether…
Equal	=	two values are equal
Not equal	<>	two values are not equal
Greater than	>	one value is greater than the other
Less than	<	one value is less than the other
Greater than or equal	>=	a value is greater than or equal to the other
Less than or equal	<=	a value is less than or equal to the other

Contact comparison instructions (API 224–230)

Syntax: **`LD*  S1  S2`** where S1 and S2 are data source devices (K, H, KnX, KnY, KnM, KnS, T, C, D, E, F). The instruction connects directly to the left bus.

API	16-bit	32-bit	Continuity when	Discontinuity when
224	LD=	DLD=	S1 = S2	S1 ≠ S2
225	LD>	DLD>	S1 > S2	S1 ≤ S2
226	LD<	DLD<	S1 < S2	S1 ≥ S2
228	LD<>	DLD<>	S1 ≠ S2	S1 = S2
229	LD<=	DLD<=	S1 ≤ S2	S1 > S2
230	LD>=	DLD>=	S1 ≥ S2	S1 < S2

- The comparison is algebraic (signed). In 16-bit mode, bit b15 = 1 means the value is negative; in 32-bit mode, b31 = 1 means negative.
- When comparing 32-bit counters (C200–C254), use the 32-bit (DLD*) instructions.

Using a comparator in WPLSoft

Double-click in the ladder area to open the **Input Instruction** box and type, for example:

```
LD>= D0 D1
```

D0 and D1 are the values being compared. When the condition is true, the contact passes power and the output turns ON.

Worked example (values shown in red in the screenshot)

Instruction	Value 1	Value 2	Result	Output
`LD>= D0 D1`	8	8	True	M0 ON
`LD<= D2 D3`	2	5	True	M1 ON
`LD> D4 D5`	12	14	False	M2 OFF
`LD< D6 D7`	16	20	True	M3 ON
`LD= D8 D9`	15	15	True	M4 ON
`LD<> D10 D11`	10	10	False	M5 OFF

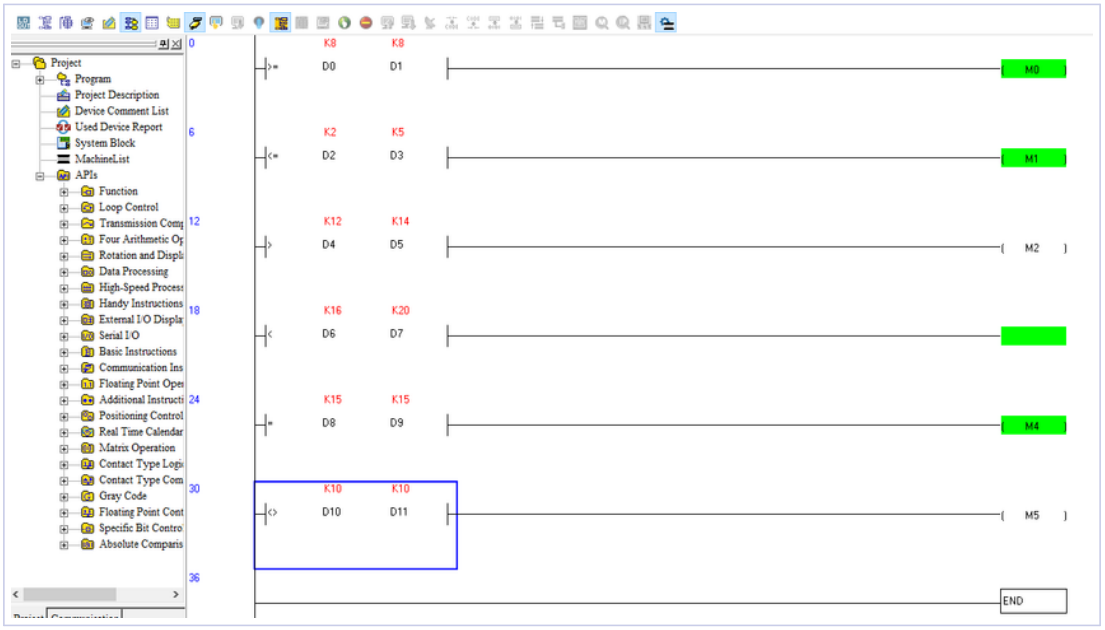


Figure 6.1 – Comparison contacts in run mode (green = condition true)

07 Positive Trigger & Negative Trigger

Detect the exact moment a signal changes – pressed or released.

Positive trigger (rising edge)

A positive trigger is the transition of a signal from **OFF (0) to ON (1)**. The PLC detects the change at the exact moment the signal turns ON and executes the related instruction **only once** (for one scan cycle). It is commonly used to detect a push-button *press* or trigger event.

Negative trigger (falling edge)

A negative trigger is the transition of a signal from **ON (1) to OFF (0)**. The PLC detects the change at the exact moment the signal turns OFF and executes the related instruction **only once** (for one scan cycle). It is commonly used to detect a push-button *release*.

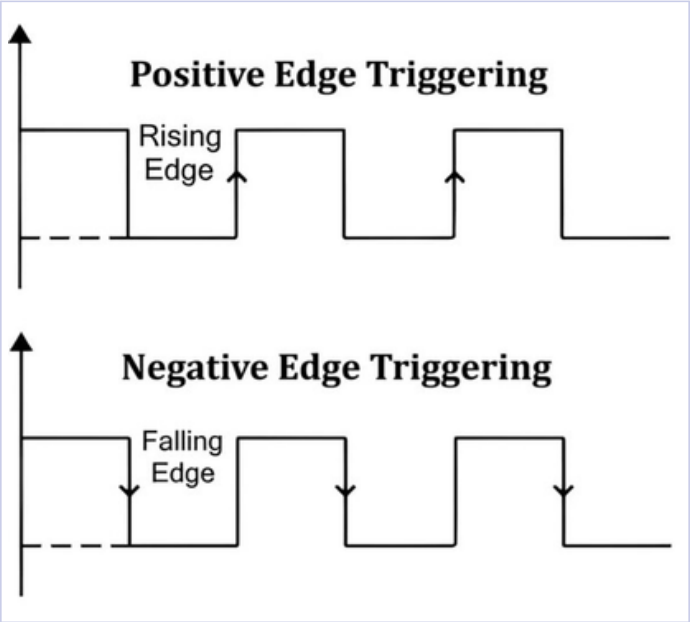


Figure 7.1 – Rising-edge (positive) and falling-edge (negative) triggering

Edge	Instruction	Example
Positive (rising) edge	`LDP`	`LDP X0`
Negative (falling) edge	`LDF`	`LDF X2`

In WPLSoft, type the instruction in the Input Instruction box (e.g. **`LDP X0`** or **`LDF X2`**).

Example

- **`X0`** (rising edge) with **`X1`** as stop – **`Y0`** turns ON the moment **`X0`** is pressed and is held by its own contact.
- **`X2`** (falling edge) with **`X3`** as stop – **`Y1`** turns ON the moment **`X2`** is released.



Figure 7.2 – LDP X0 and LDF X2 examples (Y0 and Y1 held by their own contacts)

KEY POINTS

- LDP = detects OFF→ON (button pressed); LDF = detects ON→OFF (button released).
- The output pulse lasts only one scan, so latch it (SET or a seal-in contact) if it must stay on.
- The same idea also exists for series and parallel contacts: ANDP / ANDF and ORP / ORF.

08 Set & Reset

Latch an output ON and unlatch it later.

Set (SET)

SET turns ON and **latches** a bit or output. Once SET activates the device, it **stays ON even if the input condition turns OFF**. The output stays ON until a **RESET (RST)** instruction is executed.

```
SET Y0
```

Reset (RST)

RST turns OFF and unlatches a bit or output that was turned ON by SET. When the reset condition becomes true, the PLC clears the device to OFF and it **remains OFF until it is set again**.

```
RST Y0
```

Example

`X0` sets `Y0` (it stays ON even after X0 is released). `X1` resets `Y0` (it stays OFF).

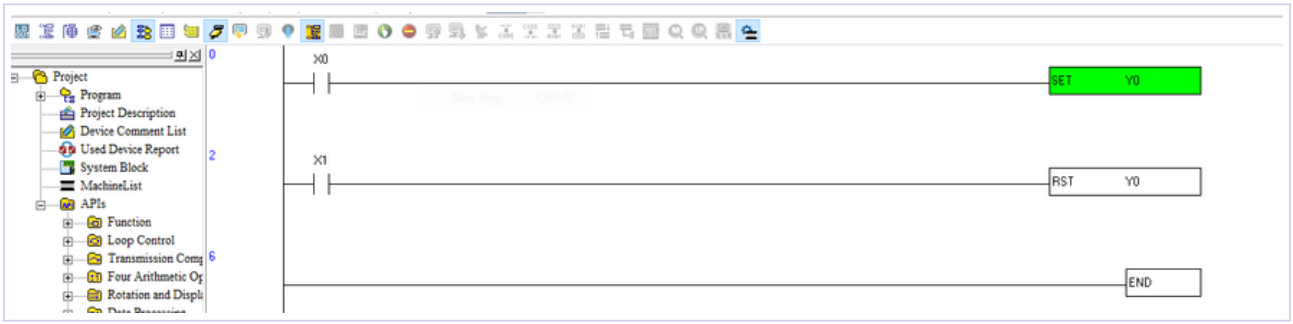


Figure 8.1 – SET Y0 (X0) and RST Y0 (X1)

	Output coil (OUT)	SET / RST
Behaviour	Follows its input – ON only while the rung is true	SET latches ON until RST is executed
Needs seal-in contact?	Yes, to hold ON	No

KEY POINTS

- SET and RST work as a pair on the same device.
- RST is also used to clear timers, counters and data registers (e.g. RST T250, RST C0).

09 Move

Copy a number from one place to another.

MOV – 16-bit move

MOV copies a 16-bit (word) value from a source to a destination register without changing the source data. It is commonly used for data transfer, initialisation and updating register values during program execution.

```
MOV K8 D0
```

Part	Meaning
MOV	The operation (move data)
K8	Source operand – a constant. Use the prefix **K** for a decimal constant
D0	Destination operand – where the data is stored

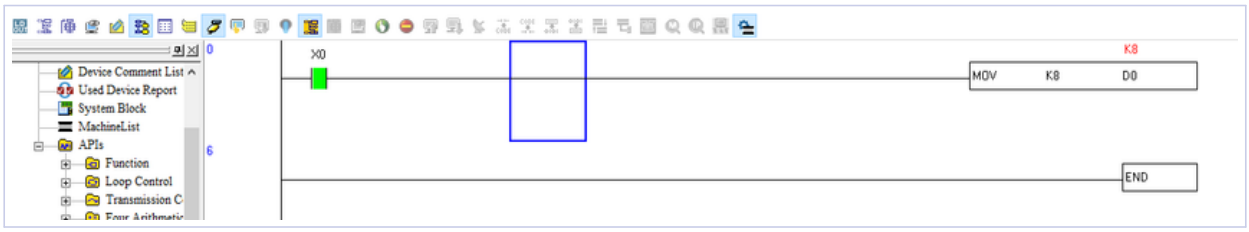


Figure 9.1 – MOV K8 D0 executed when X0 is ON

DMOV – 32-bit move

DMOV moves a 32-bit (double-word) value from the source to the destination, allowing larger numbers to be handled. A 32-bit destination uses **two consecutive registers** (D0 and D1 in this example).

```
DMOV K98654 D0
```



Figure 9.2 – DMOV (32-bit move) example

KEY POINTS

- A 16-bit MOV can only hold −32,768 to 32,767. The value 98654 is larger, so DMOV is required.
- The source is never changed by a move.

10 Arithmetic Functions

Add, subtract, multiply and divide – in 16-bit, 32-bit and floating-point.

Arithmetic functions perform addition (**ADD**), subtraction (**SUB**), multiplication (**MUL**) and division (**DIV**) on data in registers. They read the source values and store the result in a destination register. They are used in calculations, counters, timers and data processing.

The prefix **K** marks a constant (a fixed decimal value). General syntax:

```
OPERATION   Source1   Source2   Destination
```

> **ALWAYS ON (M1000)**
> **M1000** is a constant TRUE bit. It keeps logic active in every scan – useful for continuous operations, initialisation and unconditional instructions, without needing a physical input.

16-bit arithmetic

Instruction	Example	Calculation	Result
ADD	ADD D0 D1 D2	3 + 7	10
SUB	SUB D3 D4 D5	16 − 12	4
MUL	MUL D6 D7 D8	12 × 12	144
DIV	DIV D12 D14 D16	525 ÷ 5	105

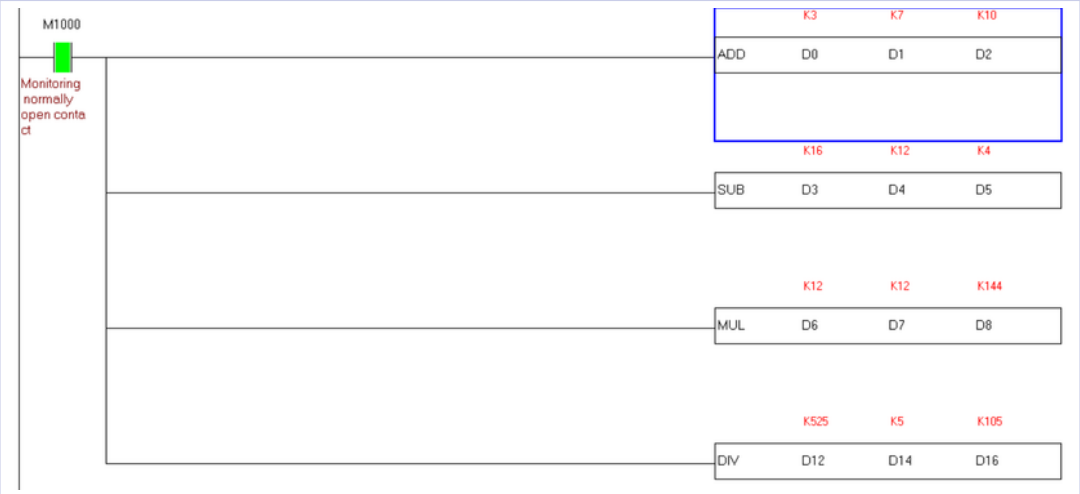


Figure 10.1 – 16-bit ADD, SUB, MUL and DIV driven by M1000

A 16-bit register stores only 16-bit values. If a calculation exceeds its range, the extra bits are truncated (**overflow**), and only the lower 16 bits are stored. Both the inputs and the result must fit within 16 bits.

NOTE

A 16-bit MUL produces a 32-bit result stored in two registers (D8 and D9 here), and a 16-bit DIV stores the quotient in the destination and the remainder in the next register (D16 and D17 here). Keep those neighbouring registers free.

32-bit (double integer) arithmetic

32-bit operations need the **D**-prefixed instructions: **DADD, DSUB, DMUL** and **DDIV**. The normal instructions handle only 16-bit data. The D prefix makes the PLC use two consecutive registers, giving a much larger range without overflow in most cases. If a result exceeds 32 bits, the higher bits are truncated.

Instruction	Example	Calculation	Result
DSUB	DSUB K490 K50 D2	490 − 50	440
DMUL	DMUL K20 K5 D4	20 × 5	100
DDIV	DDIV K30 K5 D6	30 ÷ 5	6

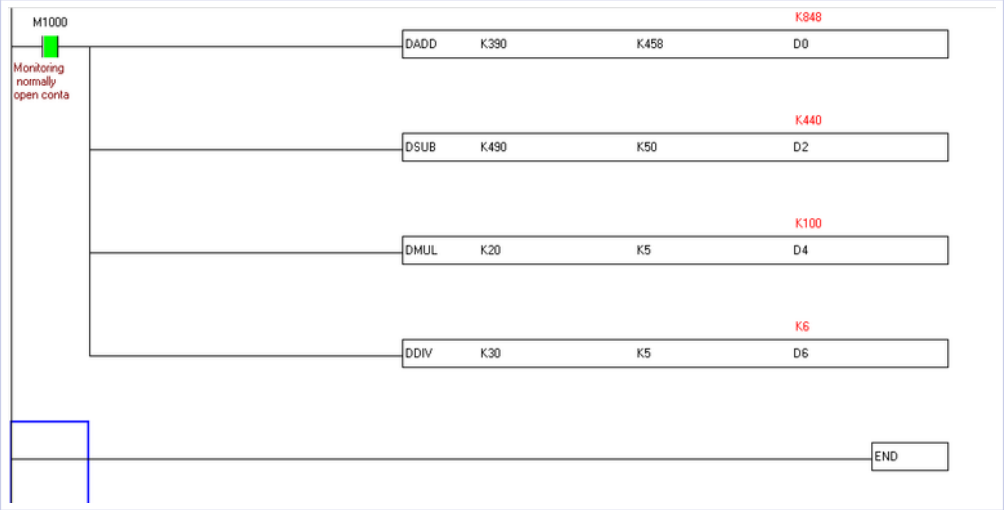


Figure 10.2 – 32-bit integer arithmetic

32-bit floating-point (REAL) arithmetic

A 32-bit floating-point (REAL) value is different from a double integer (DINT): a DINT stores only whole numbers, while a float stores decimals. Float operations use the **D** prefix and an **R** suffix (**DADDR, DSUBR, DMULR, DDIVR**), and constants use the prefix **F** (for example F5.500).

Instruction	Example	Result
DADDR	DADDR F390.000 F458.000 D0	F848.000
DSUBR	DSUBR F490.000 F50.000 D2	F440.000
DMULR	DMULR F20.000 F5.000 D4	F100.000
DDIVR	DDIVR F30.000 F5.000 D6	F6.000

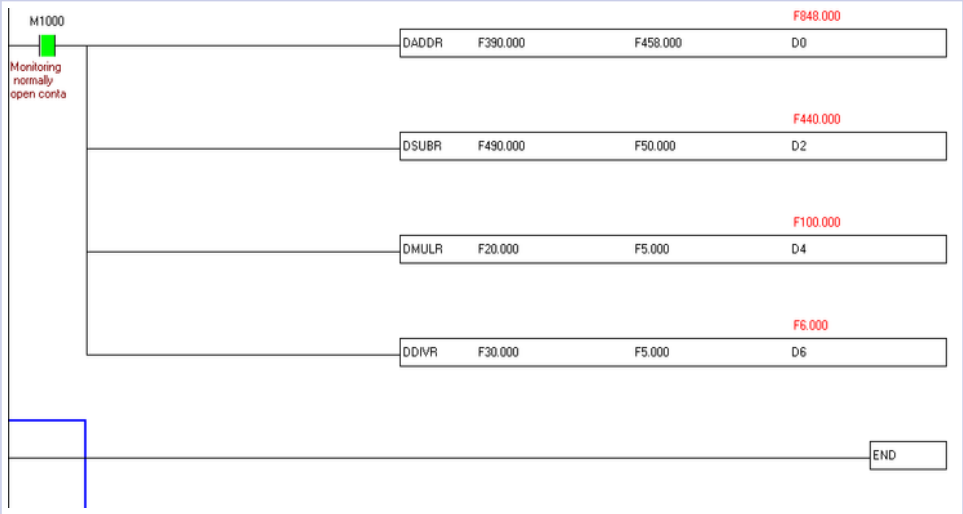


Figure 10.3 – Floating-point arithmetic (results shown in float format)

Monitoring floating-point values

Floating-point values must be viewed as **Float**. Monitoring them as integers truncates the decimals and shows incorrect results. Always match the register type with the display type.

Set it in WPLSoft: **View → Monitoring Data Format → Float**.

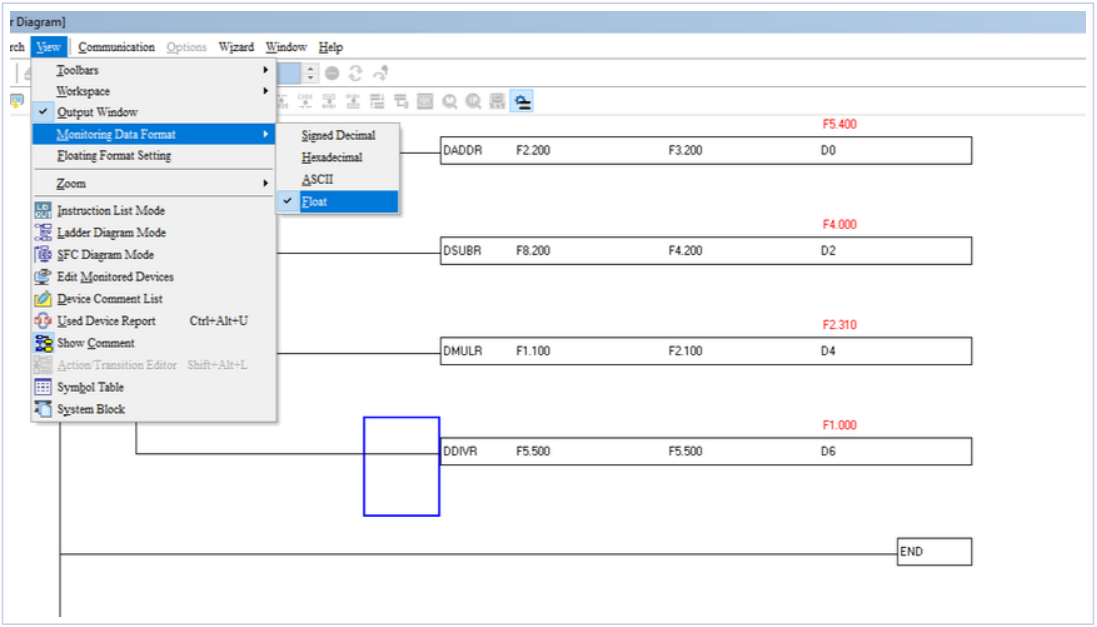


Figure 10.4 – View → Monitoring Data Format → Float (e.g. 2.2 + 3.2 = 5.4)

KEY POINTS

- MOV/ADD/SUB/MUL/DIV = 16-bit; D-prefix = 32-bit integer; D-prefix + R suffix = 32-bit float.
- Use K for integer constants and F for float constants.
- Always monitor float registers in Float format.

11 Jump

Skip part of the program, or jump to a labelled position.

The **jump** instruction lets the program skip to a specific label (pointer) in the ladder logic. It controls program flow, enabling conditional execution, loops or the skipping of unnecessary rungs. In Delta PLCs the conditional jump instruction is **CJ**, used with a pointer such as **P0**.

A **label** (pointer) is a marker position in the program. It does not execute any logic; it only defines a location. The CJ instruction uses it to transfer control, and the PLC continues execution from the labelled point when the jump condition is met.

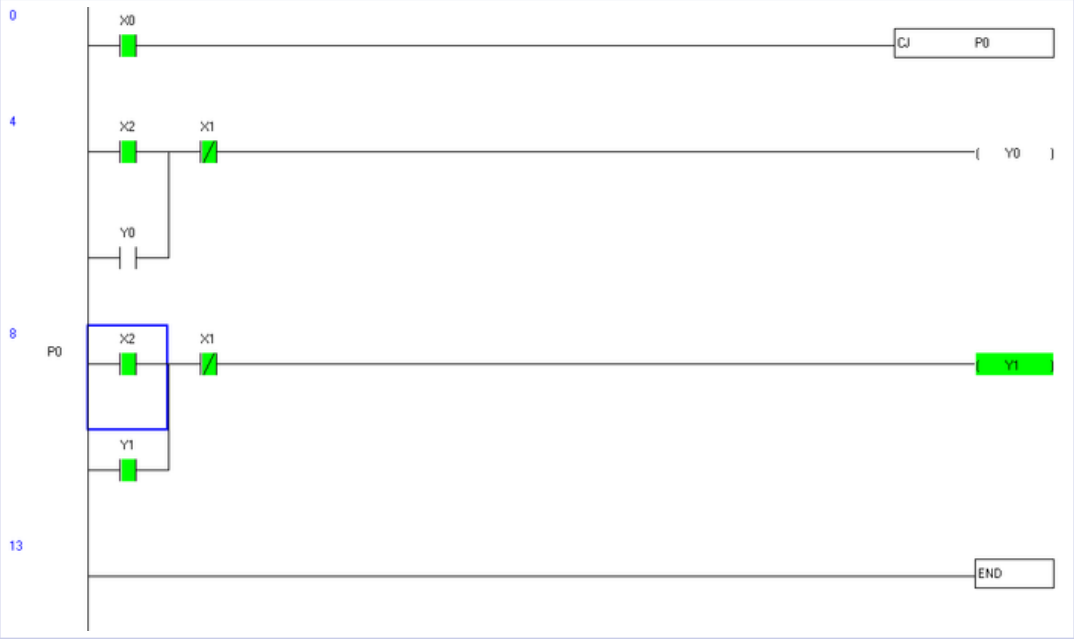


Figure 11.1 – X0 closes the jump: CJ P0 skips the rungs between the instruction and label P0

Execution flow

1	When the PLC scans the rung `X0` → `CJ P0` and the condition is ON, the jump is executed.
2	The PLC immediately jumps to the rung with label **P0**.
3	Any rungs between the CJ instruction and P0 are **not executed** in this scan cycle.
4	The program continues normally after P0.

KEY POINTS

- Skipped rungs are not scanned, so their coils are not updated while the jump is active.
- Make sure every pointer P is unique and correctly placed to avoid unintended jumps.

12 Call Function / Subroutine

Write a block of logic once and call it whenever it is needed.

The **CALL** instruction calls a subroutine from the main program (or from another subroutine). A subroutine is written **after FEND** and ends with **SRET**.

CALL (API 01)

```
CALL   S
  S : pointer (P) of the subroutine to call
```

- Write the subroutine, labelled by its pointer, after the **FEND** instruction.
- The pointer P used by CALL cannot be the same as a pointer used by CJ.
- If only CALL is used, it can call the subroutine of the same pointer number any number of times.
- Subroutines can be nested up to **5 levels**, including the first CALL. Entering a sixth level means the subroutine will not be executed.

FEND (API 06) – end of main routine

- **FEND** marks the end of the main program. When executed by the PLC it has the same function as END.
- CALL subroutines must be written after FEND, and each must end with **SRET**. Interrupt programs must end with IRET.
- If several FEND instructions are used, place the subroutines and interrupt programs between the **last FEND and END**.

Example

X0 ON → subroutine **P0** is executed (P1 is not). **X0** OFF → subroutine **P1** is executed (P0 is not).

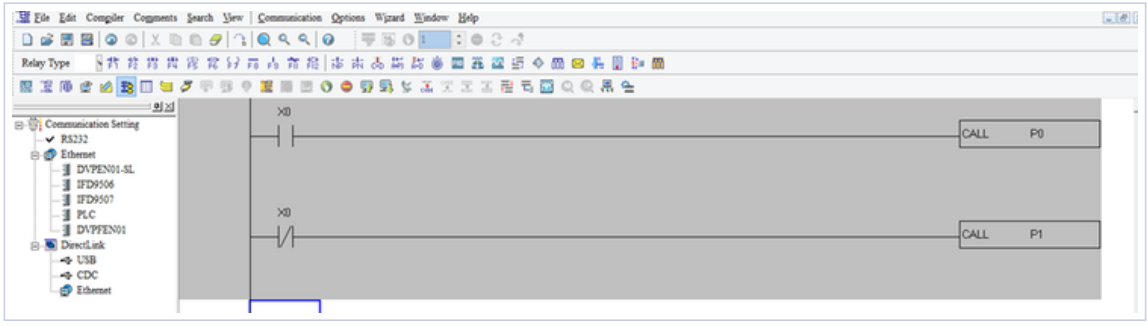


Figure 12.1 – X0 (normally open) calls P0; X0 (normally closed) calls P1

Program structure in instruction-list form:

```
LD  X0
CALL P0          ; X0 ON  -> run subroutine P0
LDI X0
CALL P1          ; X0 OFF -> run subroutine P1
FEND             ; end of main program
P0               ; subroutine P0
  ...logic...
SRET             ; return to main program
P1               ; subroutine P1
  ...logic...
SRET
END
```

KEY POINTS

- Main program → FEND → subroutines (each ending with SRET) → END.
- Maximum nesting depth is 5 levels.

13 Scaling

Convert a raw analog value into a meaningful engineering value.

Scaling converts a raw analog value from a sensor or transmitter into an engineering value that we can understand and use in the program. A sensor sends a raw electrical signal (4–20 mA or 0–10 V) to the PLC. The PLC converts this raw value into a real measurement such as temperature, pressure, level, speed or flow.

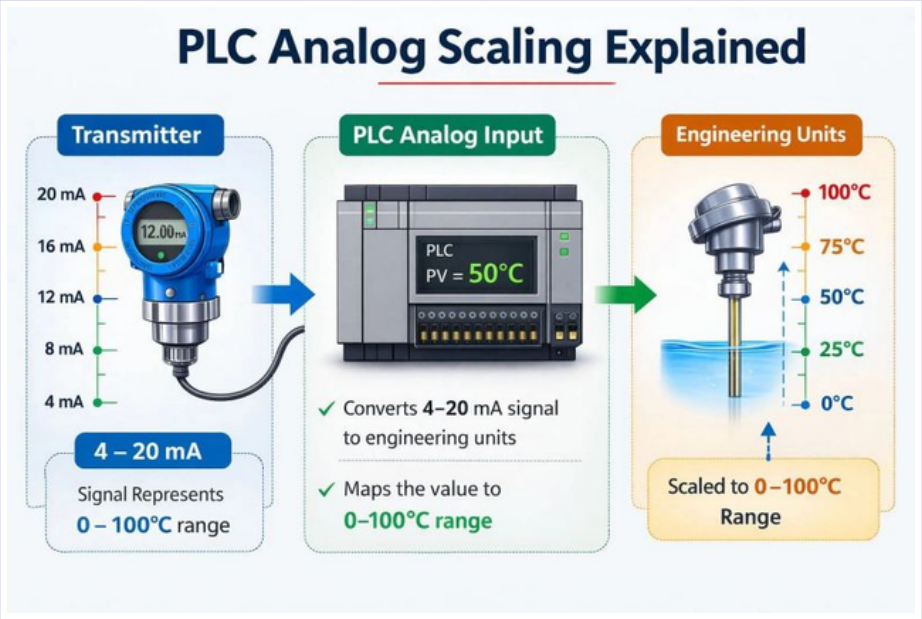


Figure 13.1 – A 4–20 mA signal scaled to a 0–100 °C range

SCLP – parameter proportional calculation (API 203)

```
SCLP   S1   S2   D
  S1 : source value       (K, H, D)
  S2 : parameters         (D)  – occupies 4 consecutive registers
  D  : operation result   (D)
```

Register	Parameter	Range
S2	Maximum source value	−32768 to 32767
S2 + 1	Minimum source value	−32768 to 32767
S2 + 2	Maximum destination value	−32768 to 32767
S2 + 3	Minimum destination value	−32768 to 32767

SCLP performs a proportional (linear) calculation:

```
D = (S1 - SourceMin) x (DestMax - DestMin) / (SourceMax - SourceMin) + DestMin
```

Example – convert 0–2000 to 0–1000

Register	Holds	Value
D1110	PLC's analog input (channel 1)	raw value
D0	Maximum source value	K2000
D1	Minimum source value	K0
D2	Maximum destination value	K1000
D3	Minimum destination value	K0
D10	Output (scaled value)	0 – 1000

```
M1000 --- SCLP D1110 D0 D10
      --- MOV K2000 D0
      --- MOV K0    D1
      --- MOV K1000 D2
      --- MOV K0    D3
```

Worked check: a raw input of 1000 gives (1000 − 0) × (1000 − 0) ÷ (2000 − 0) + 0 = **500**.

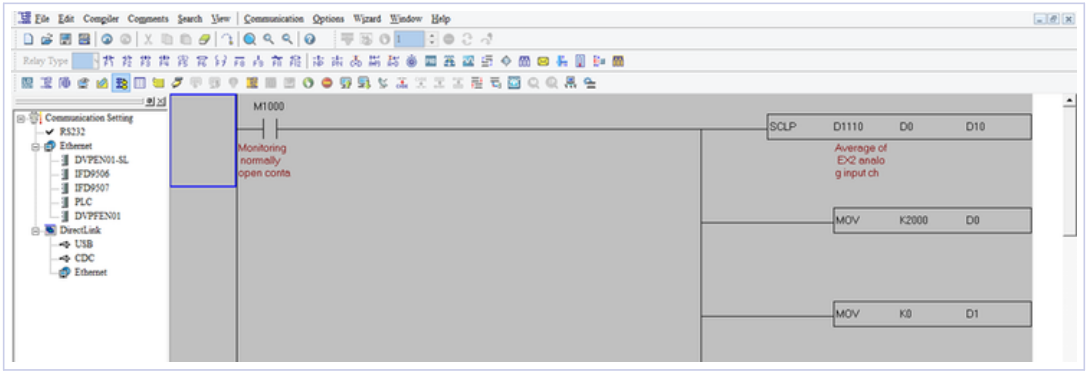


Figure 13.2 – SCLP D1110 D0 D10 and the source parameters

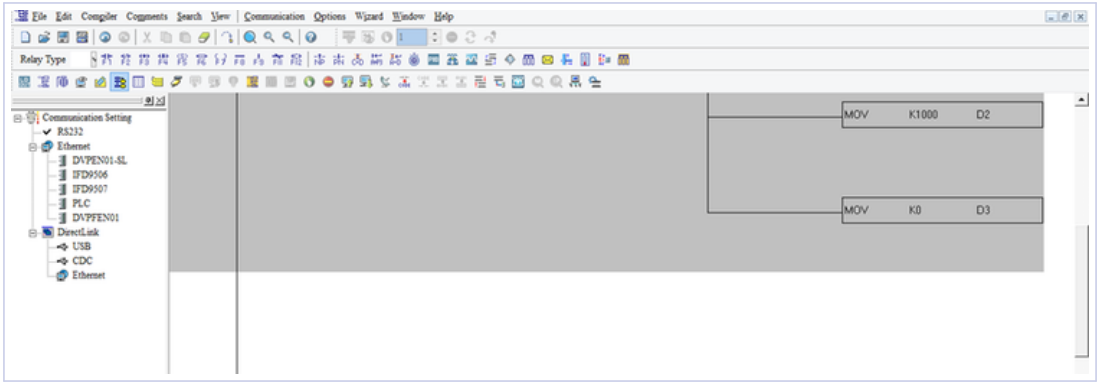


Figure 13.3 – Destination parameters D2 and D3

KEY POINTS

- Set the four parameter registers (S2 … S2+3) before SCLP uses them – an always-ON contact (M1000) keeps them loaded.
- Choose the source range to match your analog input range; check it in the module manual.

14 Real Time Clock (RTC)

Use the PLC's built-in clock and calendar for time-based control.

The **real time clock** keeps time and calendar data inside the PLC. It provides seven kinds of time data: **second, minute, hour, day, week, month and year**. This lets you do 24-hour control throughout the year – for example switching an output at a certain time of day.

Special registers and relays for the RTC

Device	Name	Function
D1313	Second	0 – 59
D1314	Minute	0 – 59
D1315	Hour	0 – 23
D1316	Day	1 – 31
D1317	Month	1 – 12
D1318	Week	1 – 7
D1319	Year	0 – 99 (last 2 digits of the year A.D.)
M1016	Year (A.D.) in RTC	OFF: shows the last 2 digits of the year. ON: shows the last 2 digits plus 2000.
M1017	±30 seconds correction	OFF → ON enables the correction. Seconds 0–29: the minute is kept and seconds reset to 0. Seconds 30–59: minute + 1 and seconds reset to 0.
M1076	RTC malfunction	ON when the set value exceeds the range or the battery is dead.
M1082	Flag change on RTC	ON: the RTC is being modified.

Example

Every minute, output coil **Y0** is ON from second 30 to second 59:

```
LD>= D1313 K30
OUT Y0
```

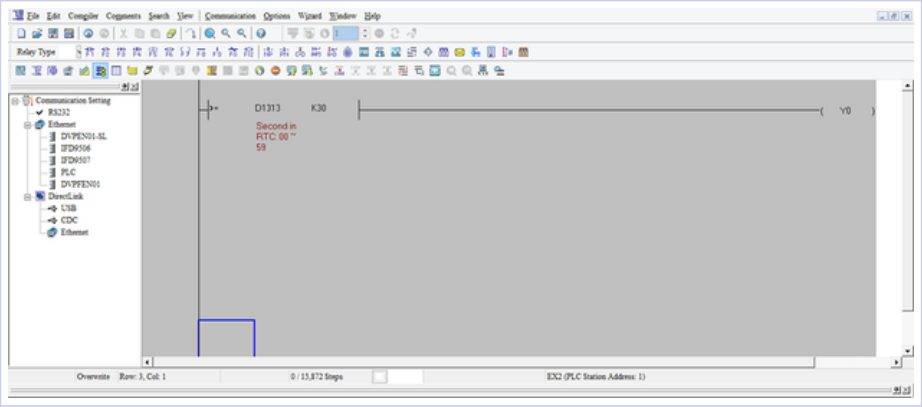


Figure 14.1 – Comparison of D1313 (seconds) with K30 drives Y0

Notes on using the RTC

- If the RTC set value is incorrect, the time is restored to **Saturday, 00:00, Jan. 1, 2000** when the PLC is powered on again.
- Per the Delta manual, D1313–D1319 are updated with the RTC only when the **TRD** instruction is used or WPLSoft is in monitoring mode.
- To correct the time: use the **TWR** instruction (SA/SX/EH2/SV/EH3/SV2 series; see API 167), or use **WPLSoft** or the digital display panel **DU-01**.
- If an invalid value is set, the RTC shows: Second → 0, Minute → 0, Hour → 0, Day → 1, Month → 1, Week → 1, Year → 0.
- The RTC of the ES2/EX2/SX2 (V2.0) and SA2/SE (V1.0) series can keep running for about one or two weeks after power-off (it varies with ambient temperature). If the machine has not been powered for that long, **reset the RTC**.

KEY POINTS

- Read the clock from D1313–D1319 and use comparison contacts to build time-based logic.
- After a long power-off, check and reset the RTC before relying on it.

Quick Reference

Device ranges (DVP20EX2 / EX2 series)

Device	Meaning	Range
X	Digital inputs (octal)	X0 – X377 (DVP20EX2 has X0–X7)
Y	Digital outputs (octal)	Y0 – Y377 (DVP20EX2 has Y0–Y5)
M	Auxiliary relays	M0 – M4095 (special M1000 – M1999)
T	Timers	T0 – T255 (retentive T240 – T255)
C	Counters	C0 – C254
D	Data registers	D0 – D9999 (special D1000 – D1999)
D1110 – D1113	Analog inputs	4 channels
D1116 – D1117	Analog outputs	2 channels

Instruction cheat sheet

Instruction	Function	Example
LD / LDI	Load normally open / normally closed contact	LD X0
OUT	Drive an output coil	OUT Y0
SET / RST	Latch ON / unlatch (RST also clears T, C, D)	SET Y0 RST C0
TMR	Timer	TMR T0 K50
CNT	16-bit counter	CNT C0 K5
LDP / LDF	Rising edge / falling edge contact	LDP X0
LD= LD> LD< LD<> LD<= LD>=	Contact comparison (DLD* for 32-bit)	LD>= D0 D1
MOV / DMOV	16-bit / 32-bit move	MOV K8 D0
ADD SUB MUL DIV	16-bit arithmetic	ADD D0 D1 D2
DADD DSUB DMUL DDIV	32-bit integer arithmetic	DSUB K490 K50 D2
DADDR DSUBR DMULR DDIVR	32-bit floating-point arithmetic	DMULR F20.0 F5.0 D4
CJ / P	Conditional jump / pointer label	CJ P0
CALL / SRET / FEND	Call subroutine / return / end of main program	CALL P0
SCLP	Proportional scaling	SCLP D1110 D0 D10
END	End of program	END

Download checklist

1	Create the project and select the **EX2** series.
2	Write the ladder program and check for errors.
3	Connect the PC to **COM1 (RS-232)** on the PLC.
4	Go online → **PC => PLC** → OK → Yes.
5	Run the PLC and test your inputs. Deselect run mode to edit.