How to display a running timer on an HMI in TIA Portal

This guide shows you how to display a running timer on an HMI in TIA Portal. You add a TONR timer to a function block, convert its elapsed time (LTime) to a string with an SCL function, download the program to the PLC, and bind the string to an I/O field on the HMI root screen. You then run the HMI simulation to watch the timer update in real time.

Prerequisites

Before you begin, make sure you have the following:

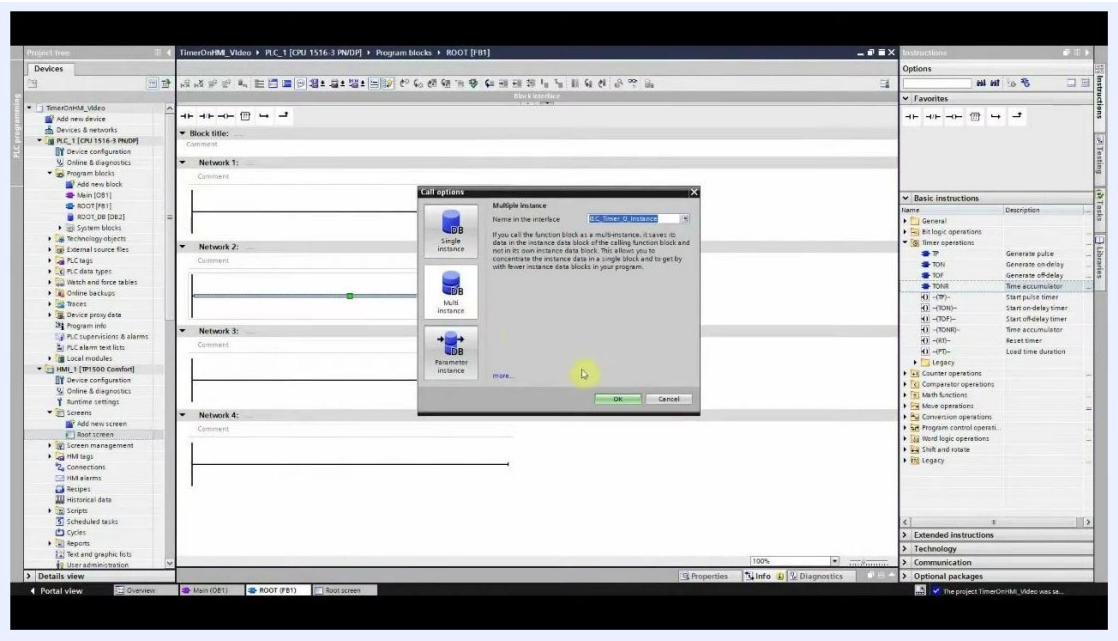
- A Siemens TIA Portal project with a PLC device (for example, PLC_1 [CPU 1516-3 PN/DP]) and a function block you can edit (for example, ROOT [FB1]).
- An HMI device in the same project (for example, HMI_1 [TP1500 Comfort]) with a **Root screen**.
- A technical article or web page that provides SCL code for converting LTime to a string, including its local variable definitions.
- For the optional download and test steps, a PLC connected to your PC (the example uses CPU 1516-3 PN/DP at address 192.168.0.1 over PN/IE).

Add and configure a TONR timer

Step 1. Open the function block for editing

1. In the project tree, go to **Program blocks** under your PLC project.
2. Open the function block you want to edit (for example, ROOT [FB1]).

The workspace shows several empty networks (Network 1, Network 2, and so on). The **Basic instructions** list on the right includes the timer operations TP, TON, TOF, TONR, and others.

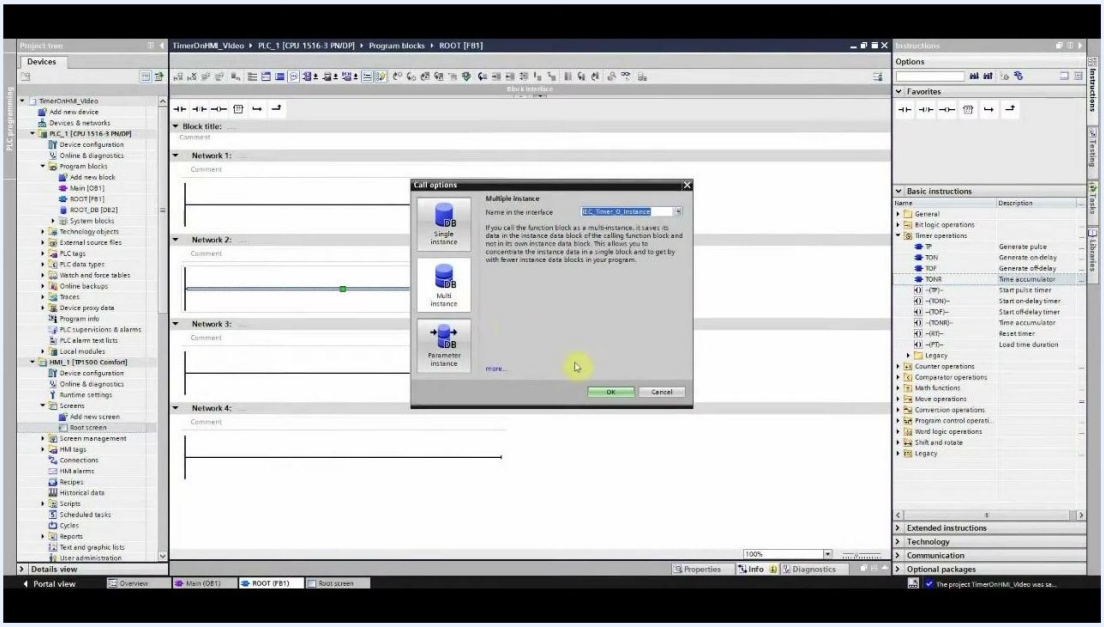


TIA Portal workspace with empty networks and timer instructions visible

Step 2. Insert a TONR timer instruction

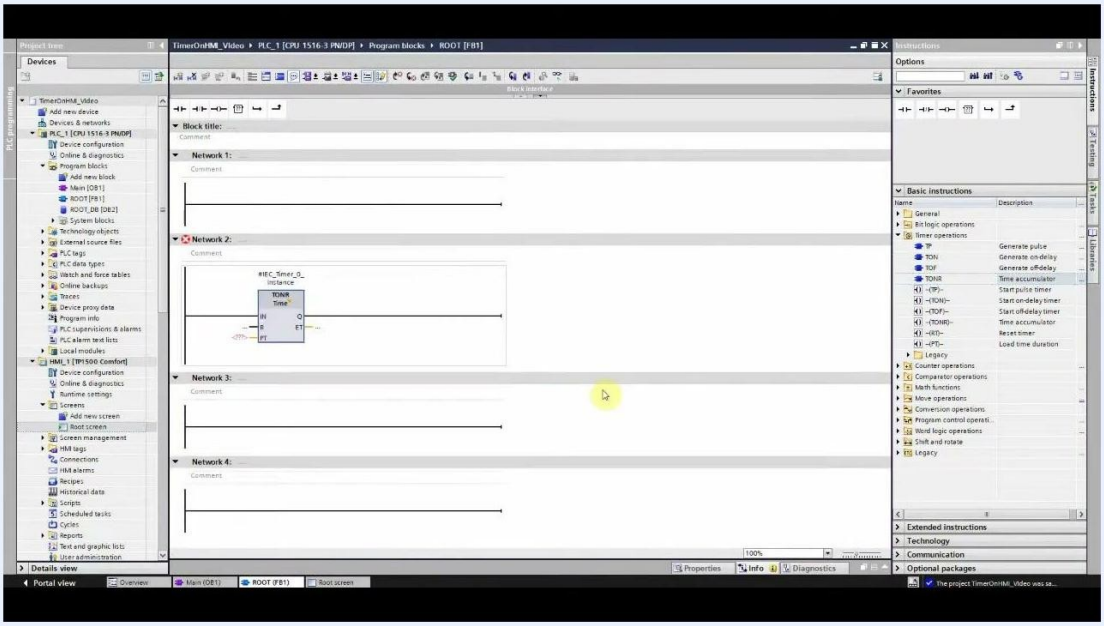
1. In the **Basic instructions** panel, locate **TONR** under **Timer operations**.
2. Drag the **TONR** instruction into Network 2 of the function block.
3. In the **Multiple instance** dialog, review the interface name for the timer instance. The default name is IEC_Timer_0 Instance.

4. Click **OK** to confirm and insert the timer.



Multiple instance dialog for timer insertion, with IEC_Timer_0 Instance selected

Confirm that the TONR timer block now appears in Network 2. The block is labeled #IEC_Timer_0_Instance and has the pins IN, Q, ET, and PT. It is not yet connected to any logic.



TONR timer block inserted in Network 2, not yet wired

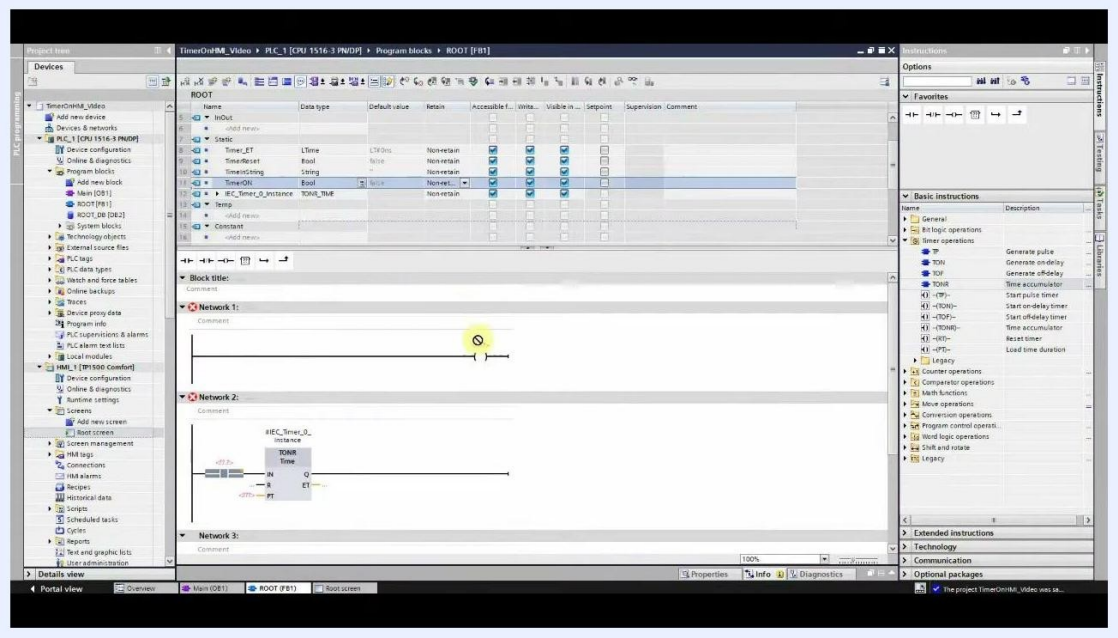
Step 3. Define and connect the control variables

1. At the top of the workspace, add or verify these variables in the function block's variable table:

    - Timer_ET (LTime)
    - TimerReset (Bool)
    - TimerString (String)
    - TimerON (Bool)
    - IEC_Timer_0_Instance (TONR_TIME)

2. In Network 1, add a normally open contact labeled #TimerON.
3. In Network 2, connect the leftmost input of the timer to #TimerON.
4. Connect the reset input to #TimerReset.

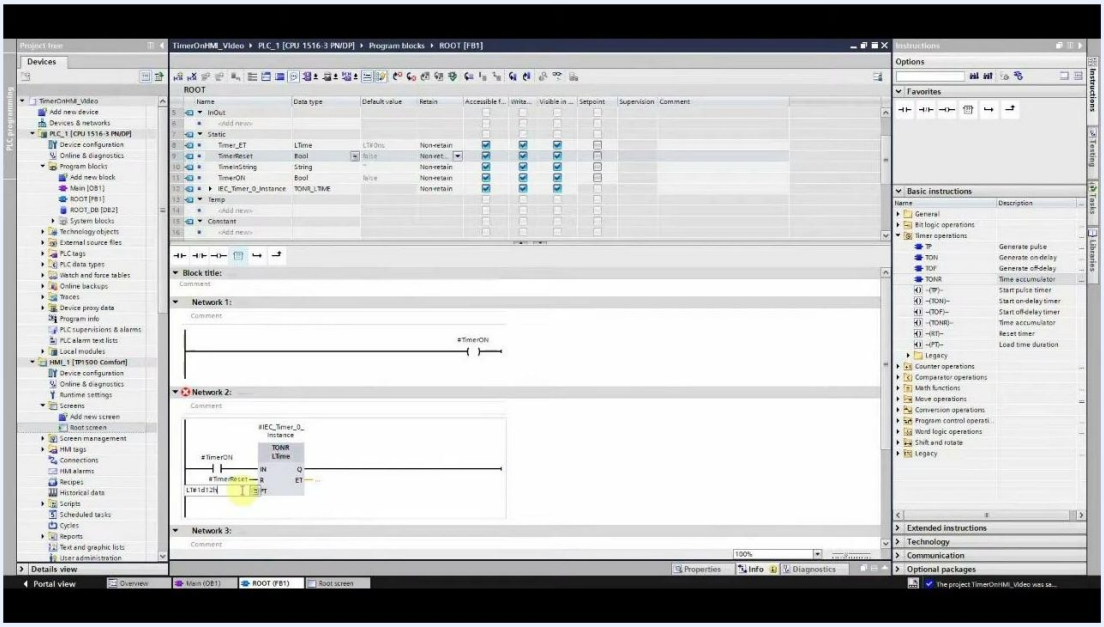
The timer block's PT (preset time) input is now ready for configuration.



Variables defined and TONR timer block partially wired

Step 4. Configure the preset time (PT)

Enter the preset time value in the PT input field. The example uses L#1d12h23m45s20ms, which is 1 day, 12 hours, 23 minutes, 45 seconds, and 20 milliseconds. This value sets the timer's delay duration.



Preset time L#1d12h23m45s20ms entered in PT input of TONR timer block

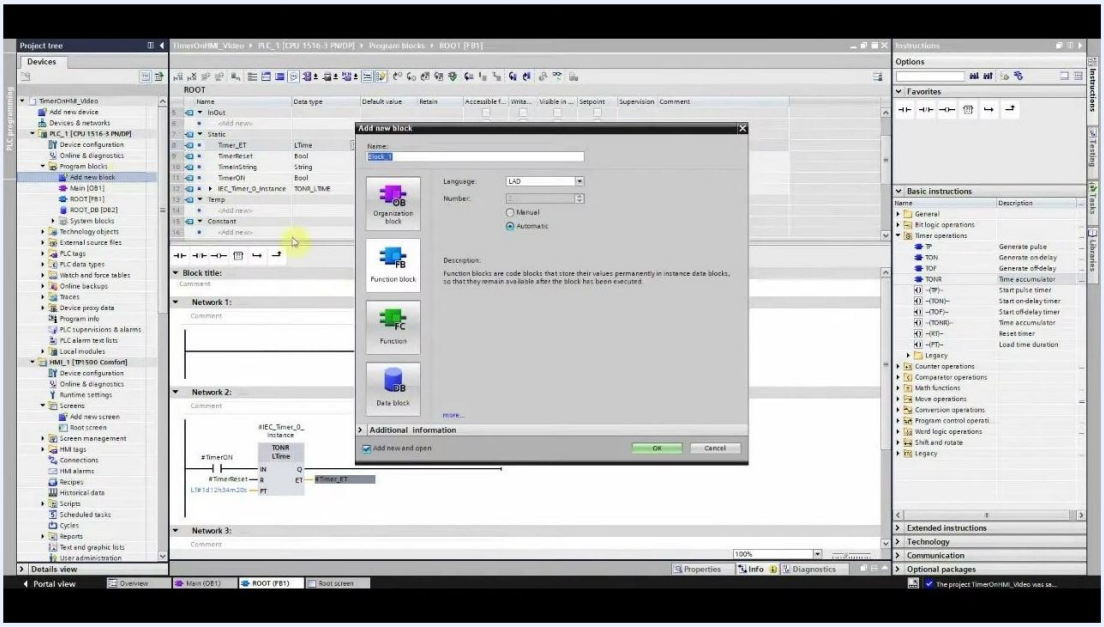
Create the LTime_To_String function

To show the elapsed time on a screen, you first convert the timer's LTime value to a string. The next steps create a function that does this.

Step 5. Add a new block

1. In the project tree, click **Add new block**.
2. In the **Add new block** dialog, select the block type (for example, Organization block, Function block, Function, or Data block).
3. Set the language (for example, LAD for Ladder Logic).
4. Choose **Manual** or **Automatic** numbering.

5. Click **OK** to create and open the new block.

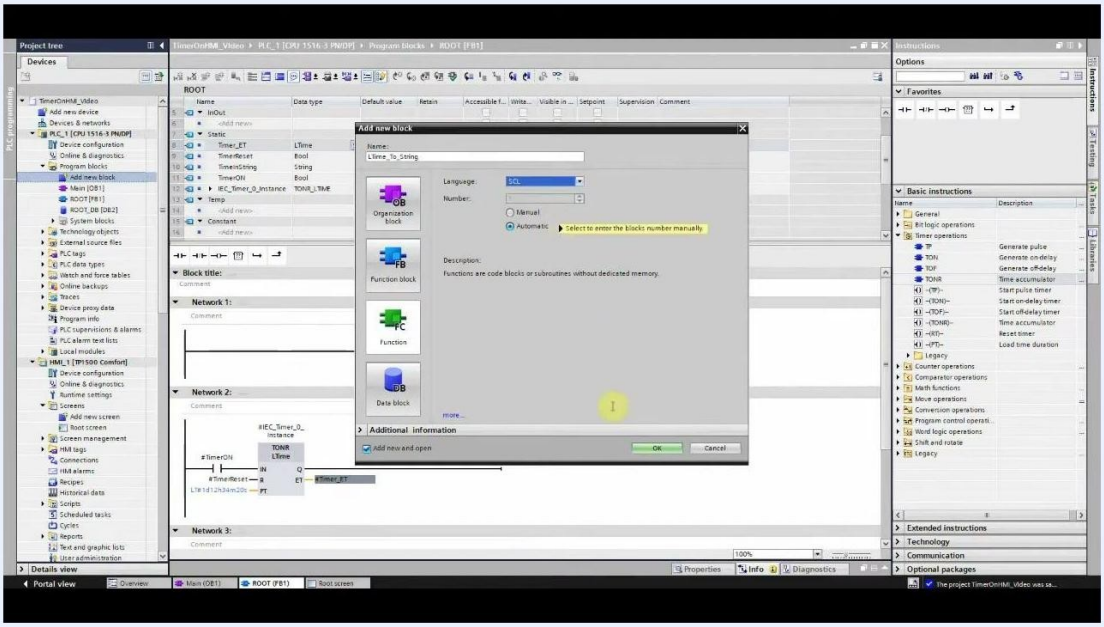


Add new block dialog with block type and language options

Step 6. Name the block and select SCL

1. In the **Add new block** dialog, enter the block name LTime_To_String.
2. Click the **Language** dropdown menu. The available languages are LAD, FBD, STL, and SCL.
3. Select **SCL** (Structured Control Language).

The block number remains set to **Automatic**.

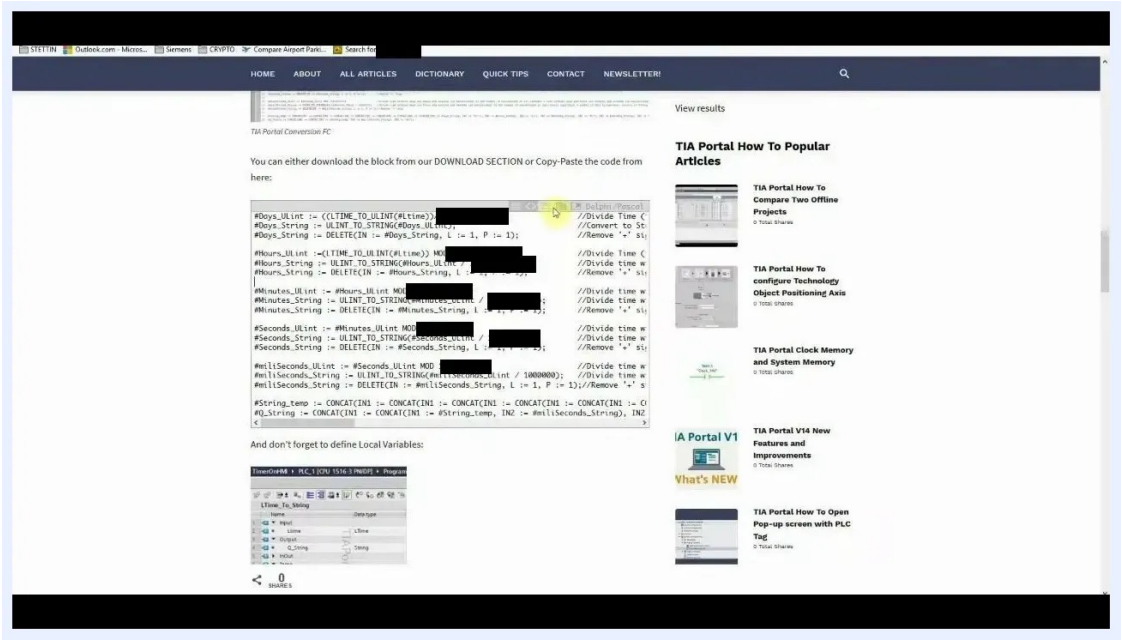


Block name entered as LTime_To_String, SCL language selected

Step 7. Find the conversion code

Open a web browser and go to a technical article or resource page that contains the LTime-to-string conversion code. The page shows variable assignments and string concatenation. The code calculates days, hours, minutes, seconds, and milliseconds, then joins them into a formatted string. The page also reminds you to define local variables.

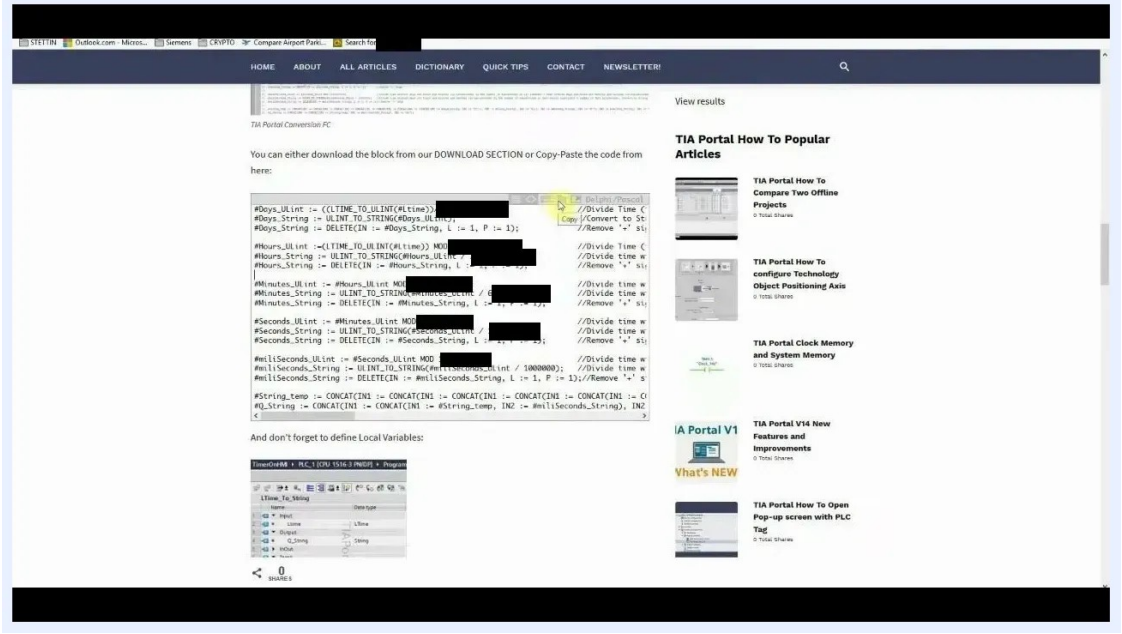
You don't need to take any action in this step. It prepares you for the next one.



Web page with LTime to string conversion code and variable definitions

Step 8. Copy the conversion code

Highlight the entire code block on the web page so you can copy it. The code includes comments for each calculation and string operation.

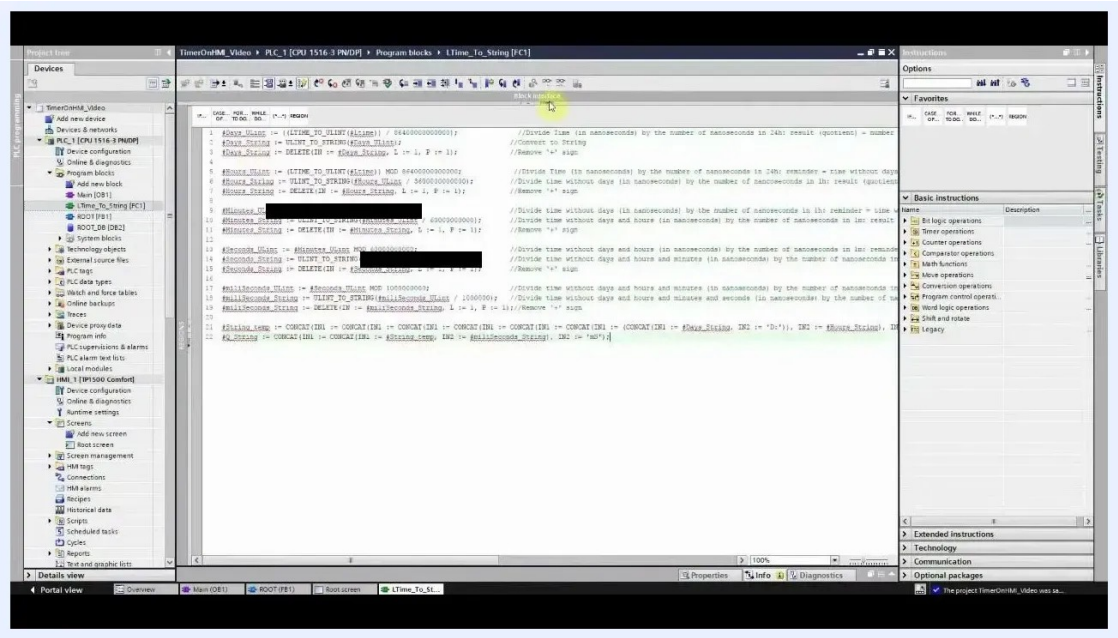


Code block for LTime to string conversion highlighted for copying

Step 9. Paste the code into the new function

1. Return to TIA Portal. The new function LTime_To_String [FC1] is open in the editor.
2. Paste the copied code into the code editor area.

The code appears in the main workspace, ready for editing or adaptation. The project tree shows the new block under **Program blocks**.

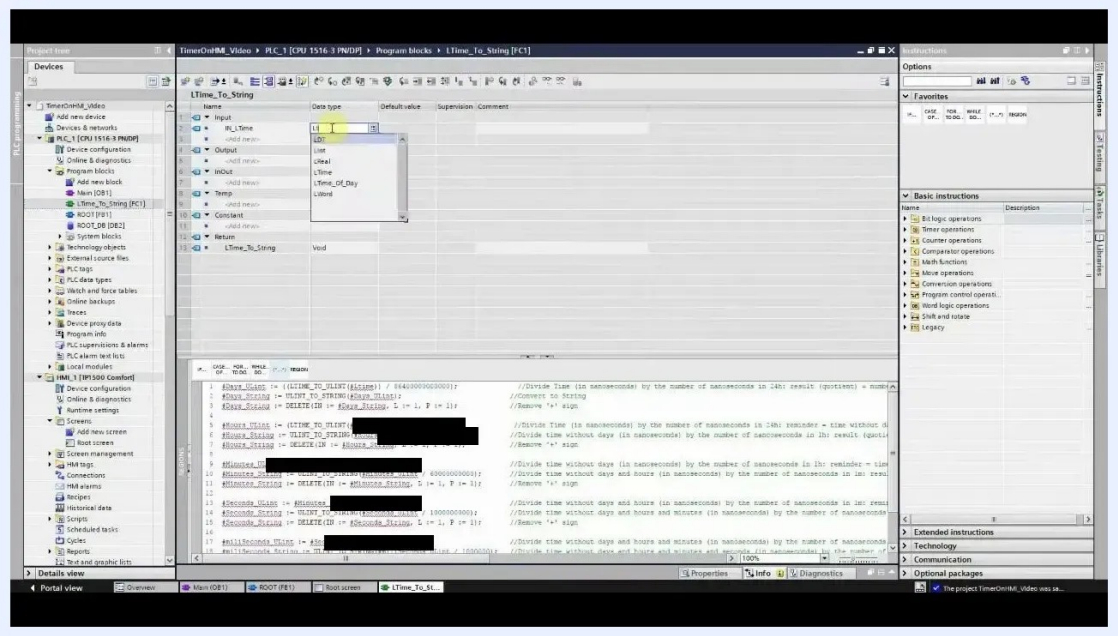


TIA Portal with LTime_To_String function block open and code pasted

Step 10. Define the input variable

1. In the function editor, go to the **Input** section.
2. Add a variable named IN_LTime.
3. Set its data type to LTime by selecting it from the dropdown menu.

You can define variables in the **Output**, **Temp**, **Constant**, and **Return** sections as the function needs them.

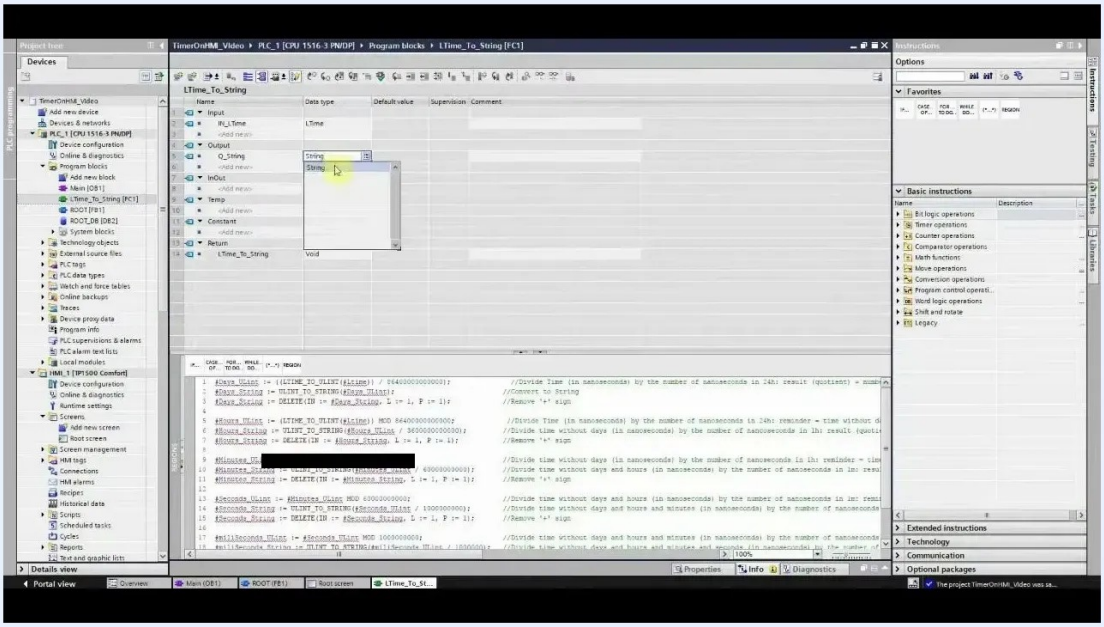


Input variable IN_LTime being defined as LTime in function block

Step 11. Add the output variable for the result string

1. In the **Output** section, add a variable named OUT_String.
2. Click the **Data type** dropdown for this variable.
3. Select String from the list.

This variable holds the formatted string result of the LTime conversion.



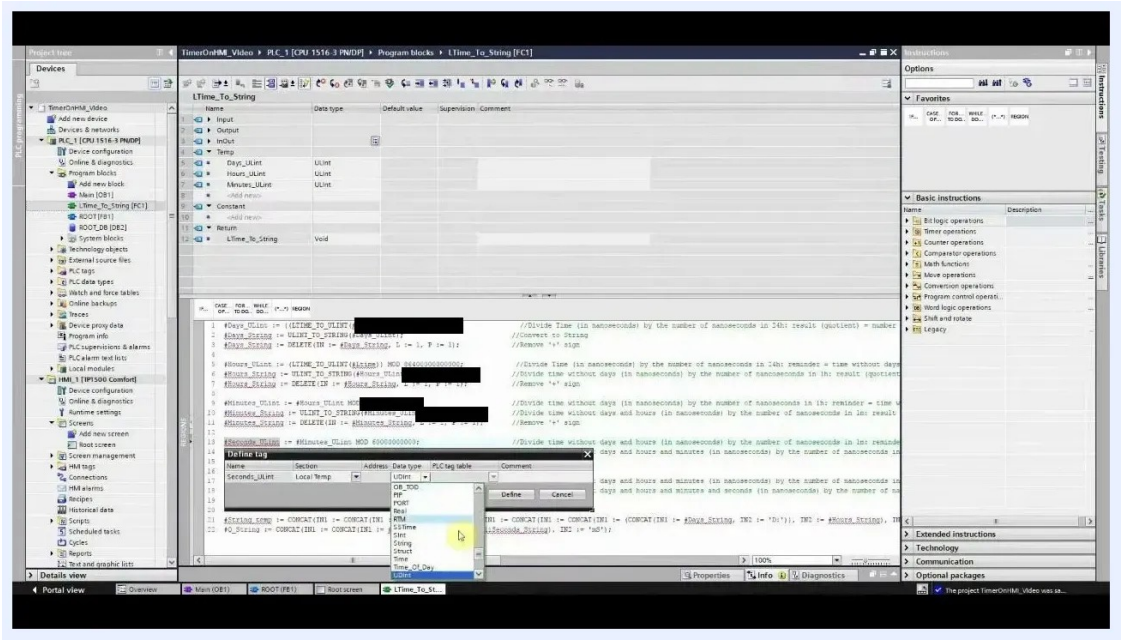
Adding output variable OUT_String and selecting String data type

Step 12. Define temporary variables for the time components

In the **Temp** section, add variables to store intermediate calculation results. Set each one to the data type ULint (Unsigned Long Integer), selecting it from the dropdown menu:

- Days_ULint
- Hours_ULint
- Minutes_ULint
- Seconds_ULint

These variables correspond to the days, hours, minutes, and seconds extracted from the LTime value.



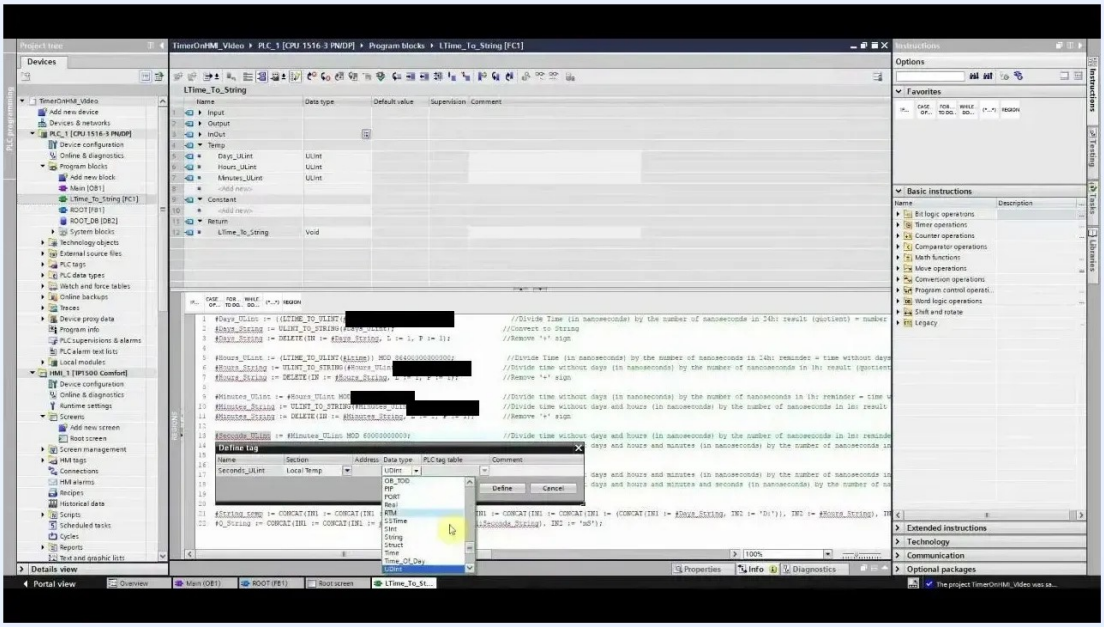
Defining temporary variables Days_ULint, Hours_ULint, Minutes_ULint, and Seconds_ULint as ULint

Step 13. Define additional temporary variables

The code may need more temporary variables, such as one for milliseconds or for string fragments. Define each one as follows:

1. If the editor prompts you, right-click the variable name in the code editor.
2. Select **Define tag** from the context menu.
3. In the **Define tag** dialog, set the section to **Local Temp** and choose the appropriate data type (for example, ULint).
4. Click **Define** to add the variable to the **Temp** section.

Repeat this for each variable the code editor highlights as undefined.



Defining a temporary variable via the Define tag dialog, selecting ULint data type

Step 14. Review the variable table and the code editor

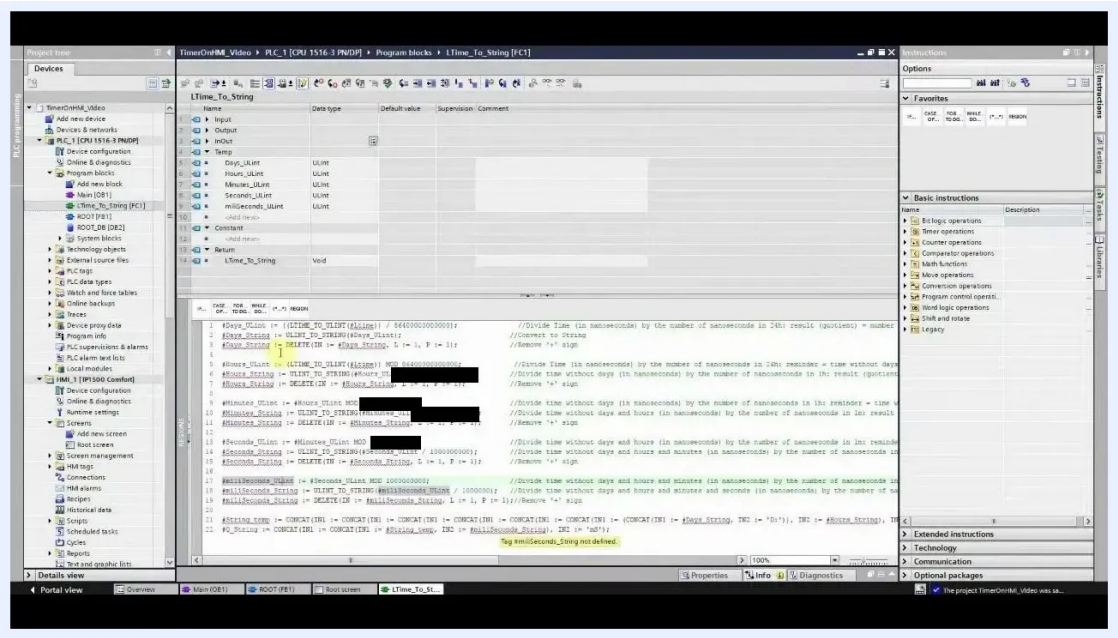
At the top of the screen, check the variable table for LTime_To_String. It shows the sections **Input**, **Output**, **Temp**, **Constant**, and **Return**. Confirm that these variables are present:

- **Input:** IN_LTime (LTime)
- **Output:** OUT_String (String)
- **Temp:** Days_ULint, Hours_ULint, Minutes_ULint, Seconds_ULint, and milliSeconds_ULint (all ULint)

No default values or supervision settings are specified for these variables.

In the lower half of the screen, review the structured text code. Comments explain each step: converting LTime to days, hours, minutes, seconds, and milliseconds, then manipulating and concatenating strings to build the output string.

A warning at the bottom reads "Tag 'milliSeconds_String' not defined." It means the code references milliSeconds_String, which you haven't defined yet.



TIA Portal screen showing the LTime_To_String function block variable table, structured text code, and a warning about 'milliSeconds_String' not being defined

Step 15. Define any remaining undefined variables

1. In the code editor, locate each highlighted or underlined variable name (for example, milliSeconds_String).
2. Right-click the variable name and select **Define tag**.
3. In the dialog, set the section to **Local Temp**.
4. Choose the appropriate data type (for example, String for milliSeconds_String).
5. Click **Define** to add the variable to the **Temp** section.
6. Repeat for any other variable flagged as undefined.

Step 16. Verify that all variables are defined

Make sure every variable referenced in the code is listed in the appropriate section (**Input**, **Output**, or **Temp**). The code editor should no longer show warnings about undefined tags. The function is then ready for compilation and testing.

Create and configure the string conversion

Step 17. Reopen the function for time conversion

1. In the project tree, go to **Program blocks** under your PLC device (for example, PLC_1 [CPU 1516-3 PN/DP]).
2. Open the function LTime_To_String [FC1].

The block interface with the variable declarations is at the top of the workspace, and the structured text code is at the bottom. The variables Days_ULint, Hours_ULint, Minutes_ULint, Seconds_ULint, and MilliSeconds_ULint are declared as ULint. Corresponding string variables are being added.

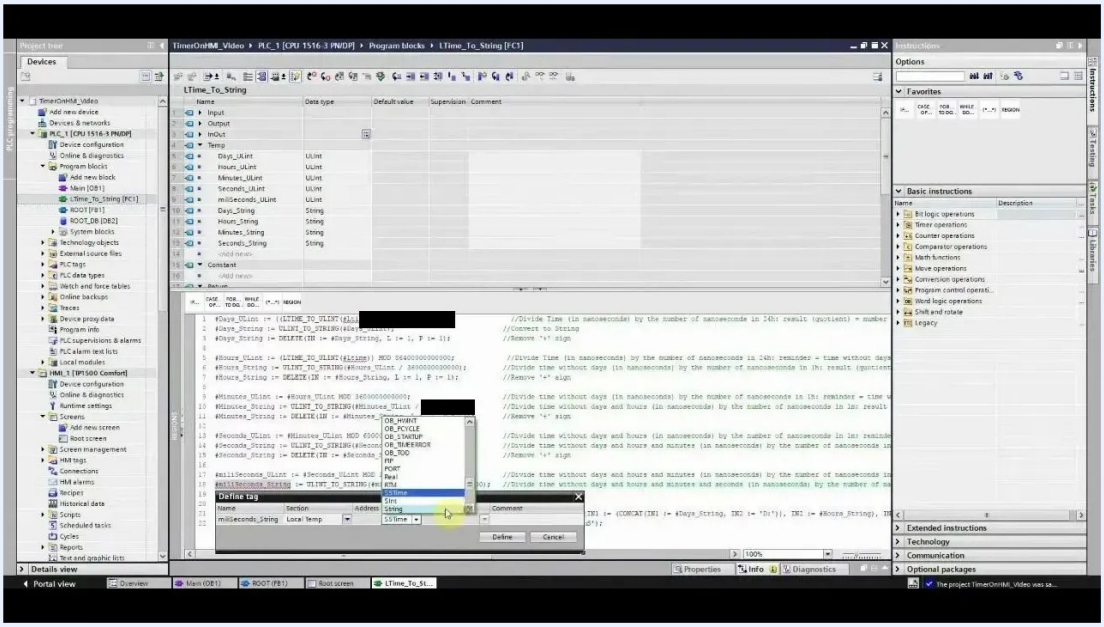
Step 18. Define the string variables

For each time component (days, hours, minutes, seconds, and milliseconds), add a new variable of type String in the block interface. Example names are Days_String, Hours_String, Minutes_String, Seconds_String, and MilliSeconds_String. In the code editor, make sure each string variable stores the converted string value of its time component.

When you add a new variable, the **Define tag** dialog appears:

1. Enter the variable name (for example, Days_String, Hours_String, MilliSeconds_String, or String_temp).
2. Set the section to Local Temp.
3. Select String from the data type dropdown list.
4. Click **Define** to confirm.

Repeat this for each required string variable.



Defining a new string variable using the Define tag dialog, with the data type dropdown open and String selected

Step 19. Verify the variable declarations

After you define all the string variables, check that the block interface lists them with the data type String. Confirm that Days_String, Hours_String, Minutes_String, Seconds_String, MilliSeconds_String, and String_temp are all present.

Step 20. Review and edit the structured text code

In the code editor, check that each time component is converted to a string with the appropriate conversion function (for example, ULINT_TO_STRING). Also check that the string variables are used in the concatenation logic that builds the final output string. Example snippets:

- Days_String := ULINT_TO_STRING(Days_ULint);
- Hours_String := ULINT_TO_STRING(Hours_ULint);
- Q_String := CONCAT(IN1 := CONCAT(IN1 := Days_String, IN2 := 'D'), IN2 := Hours_String);

Edit the code as needed so the logic converts and concatenates all time components correctly.

Integrate the function in the main program

Step 21. Add the function to the main program

1. Open the main program block (for example, ROOT [FB1]).
2. In the network view, insert the LTime_To_String function.
3. Connect the input (IN_LTime) to the timer's elapsed time output.
4. Connect the output (Q_String) to the destination where you want the string representation.

Step 22. Save and compile the project

1. Save your changes.
2. Compile the project to check for errors.
3. If errors appear (for example, undefined tags), return to the block interface and code editor to correct them.

Step 23. Optional: download to the PLC and test

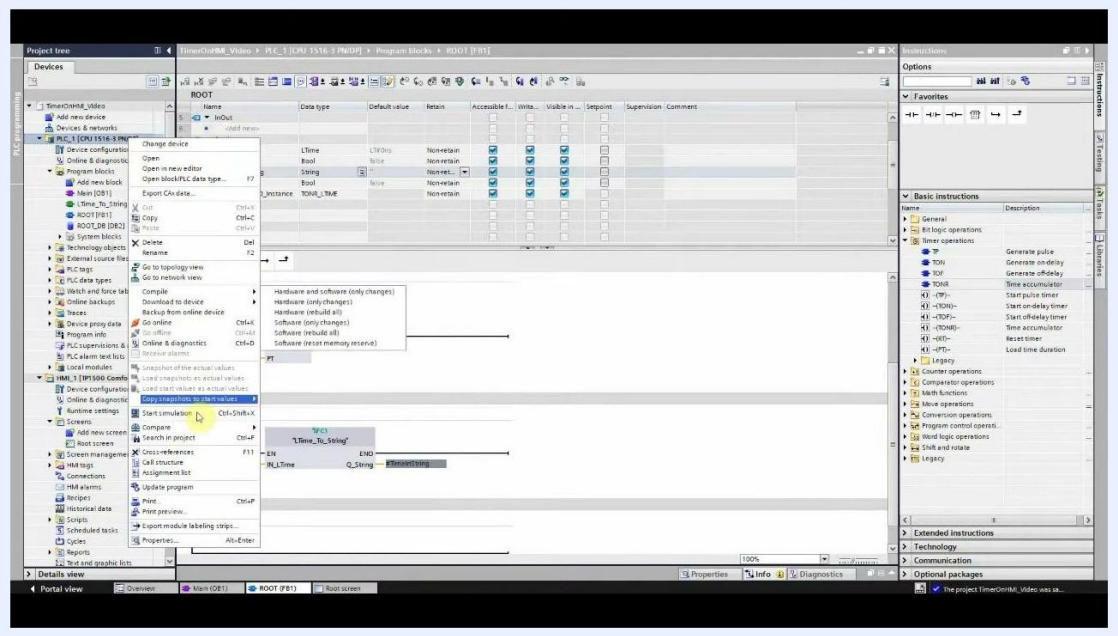
If your PLC is connected, download the updated project to the device. Then monitor the program while it runs to verify that the time conversion and string output work as expected.

Download to the PLC and monitor live data values

Step 24. Start the download to the PLC

1. In the project tree, right-click the PLC device (for example, PLC_1 [CPU 1516-3 PN/DP]).
2. Select **Download to device > Software (only changes)**. You can also press Ctrl+L.

The workspace shows the main program block (ROOT [FB1]) with the timer and string conversion logic in the ladder diagram. The **Basic instructions** toolbox is on the right, and the project tree is on the left.

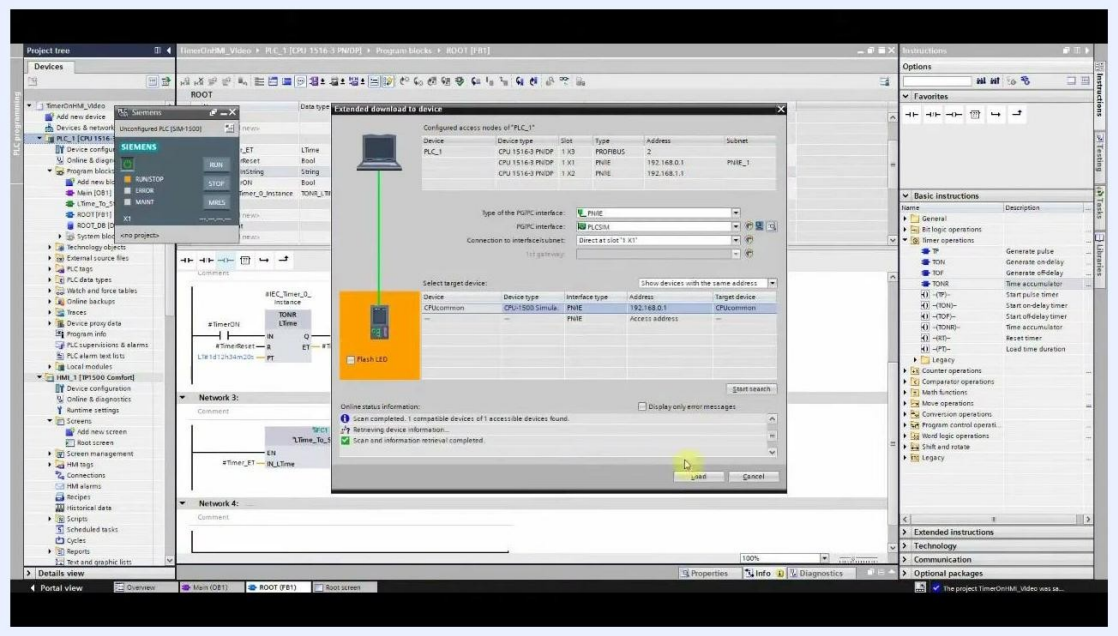


Right-click context menu on the PLC device with Download to device > Software (only changes) highlighted, and the ladder diagram with timer and string conversion logic in the workspace

Step 25. Select the target device and confirm the download

1. In the **Extended download to device** dialog, confirm that the correct PLC is selected (for example, CPU 1516-3 PN/DP with address 192.168.0.1).
2. Make sure the connection interface is set (for example, PN/IE).
3. Click **Load** to download the compiled program to the PLC.

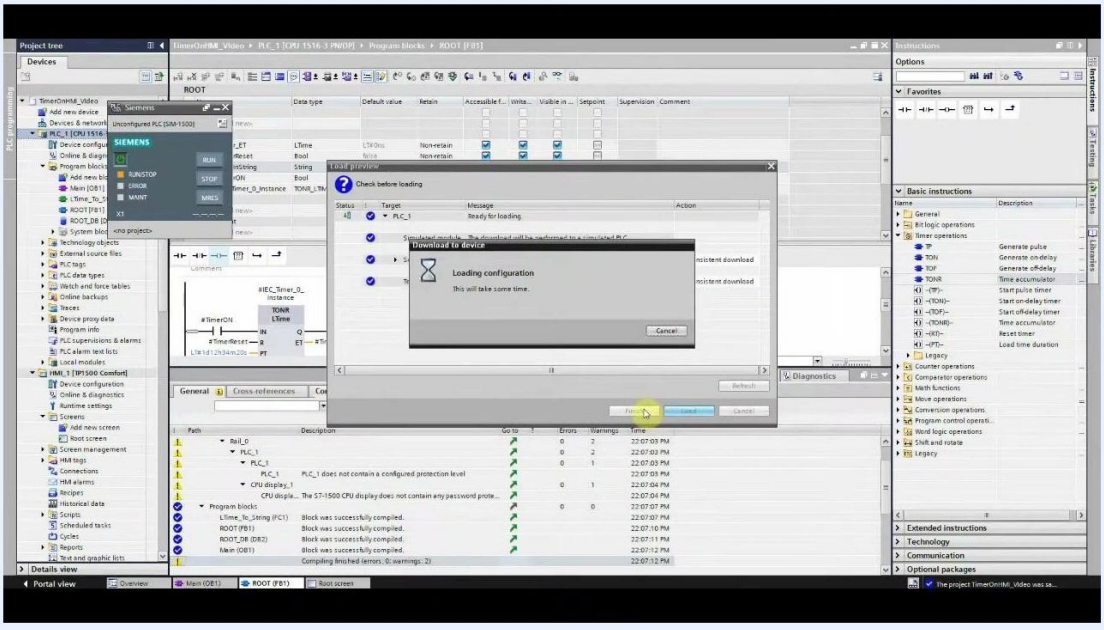
The dialog shows a visual representation of the PC-to-PLC connection and the device status.



Extended download dialog with device selection, connection interface, and Load button highlighted

Step 26. Monitor the download progress

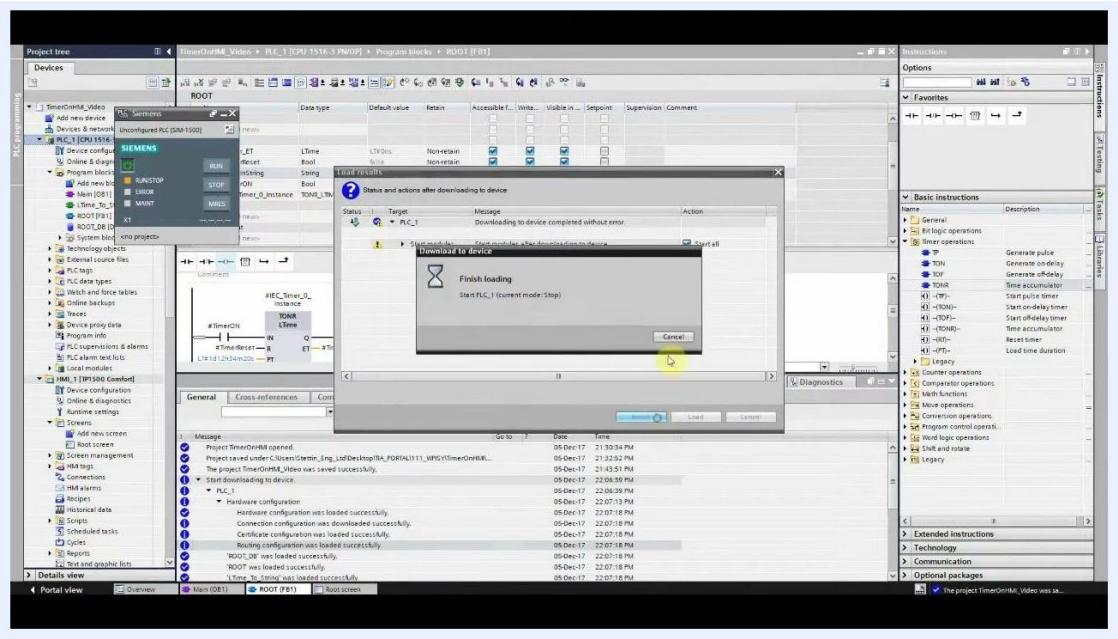
A **Download to device** progress window appears with the message "Loading configuration", a progress bar, and an hourglass icon. Wait for the configuration to finish loading. The status area shows messages such as "Ready for loading" and "Loading configuration. This will take some time."



Download progress window with Loading configuration message and progress bar

Step 27. Complete the download and start the PLC

When loading is complete, a window appears with the message "Finish loading." The status area confirms "Downloading to device completed without error." If prompted, click the button to start the PLC (for example, **Start PLC_1**).



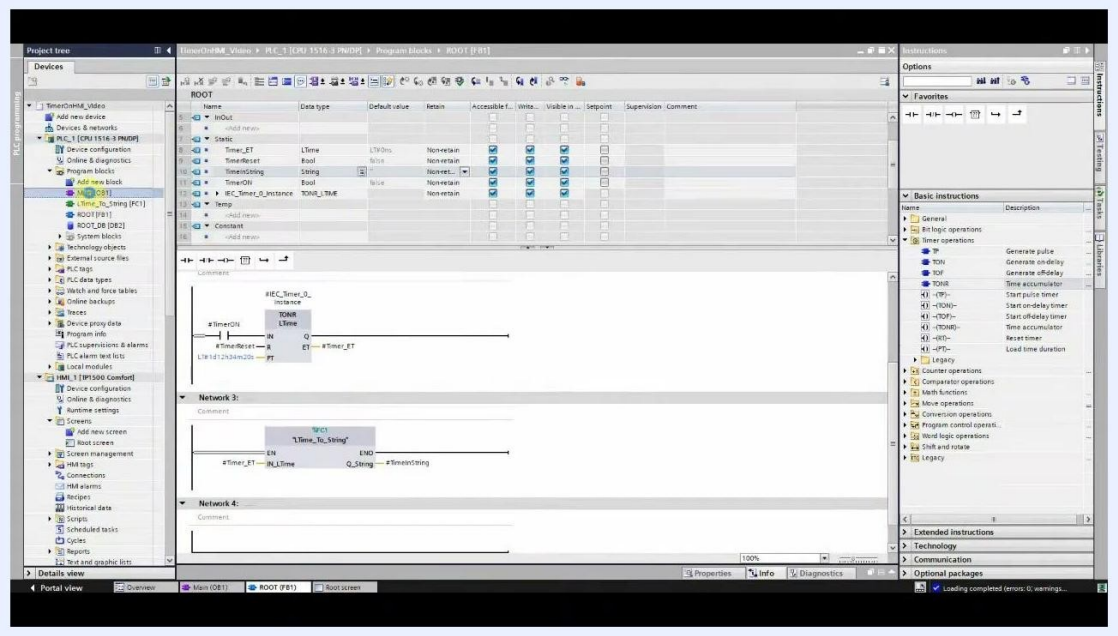
Finish loading window confirming the download completed without error, with the option to start the PLC

Step 28. Monitor live data values in the data block

Open the data block associated with the main program (for example, ROOT_DB [DB2]). The table shows variable names, data types, start values, and live monitor values. The monitor values update in real time as the PLC runs. Example values:

- LTime = T#335.835MS
- TimerReset = FALSE
- TimeInString = '00:00:00M335.835MS'
- TimerON = TRUE

The **Retain** column and other access settings are also visible. The right panel provides options for **Find and replace** and language settings.



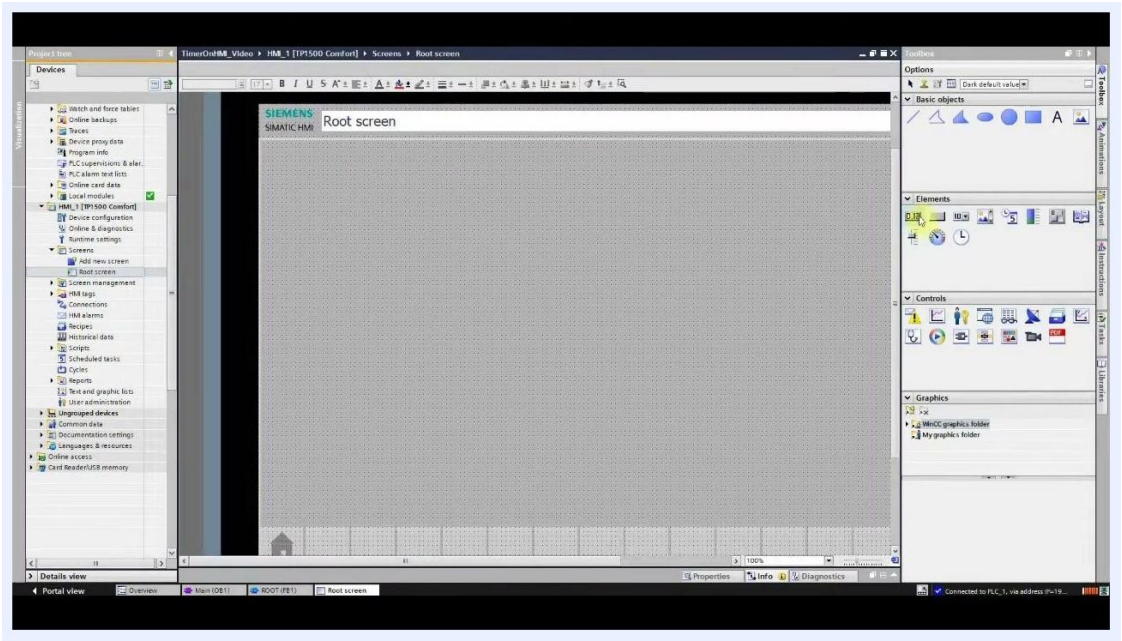
Data block view showing live monitor values for LTime, TimerReset, TimeInString, and TimerON, with Retain and access columns visible

Display the running timer on the HMI screen

Step 29. Open the HMI root screen

1. In the project tree, switch to the HMI device (for example, HMI_1 [TP1500 Comfort]).
2. Open the **Screens** section and select **Root screen**.

The screen editor shows a blank grid labeled "Root screen". The **Toolbox** on the right provides basic objects (shapes, text, images), elements (buttons, sliders, clocks), controls, and graphics folders. You can now add elements that display PLC data.

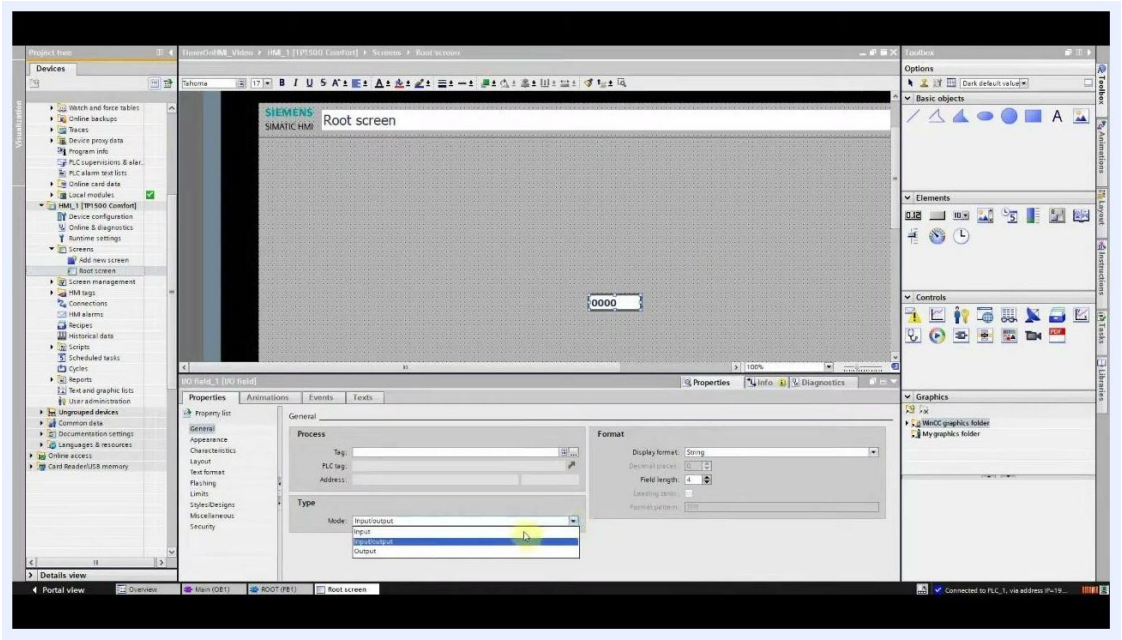


HMI screen editor with a blank Root screen and the toolbox for adding elements and controls, with the project tree showing the HMI device and screens

Step 30. Add an I/O field to show the running timer

1. With the **Root screen** open, drag an I/O field from the **Toolbox** (under **Basic objects**) to the center of the grid.
2. Select the I/O field, which displays "0000".
3. In the **Properties** panel at the bottom, find the **Type** dropdown menu.
4. Set **Type** to **Input/Output**, **Input**, or **Output** as your application requires. This example selects **Input/Output**.

The **Process** section is where you bind the I/O field to a PLC tag. You haven't configured it yet.



HMI screen editor with the I/O field selected, the Type dropdown open, and Input/Output highlighted in the Properties panel

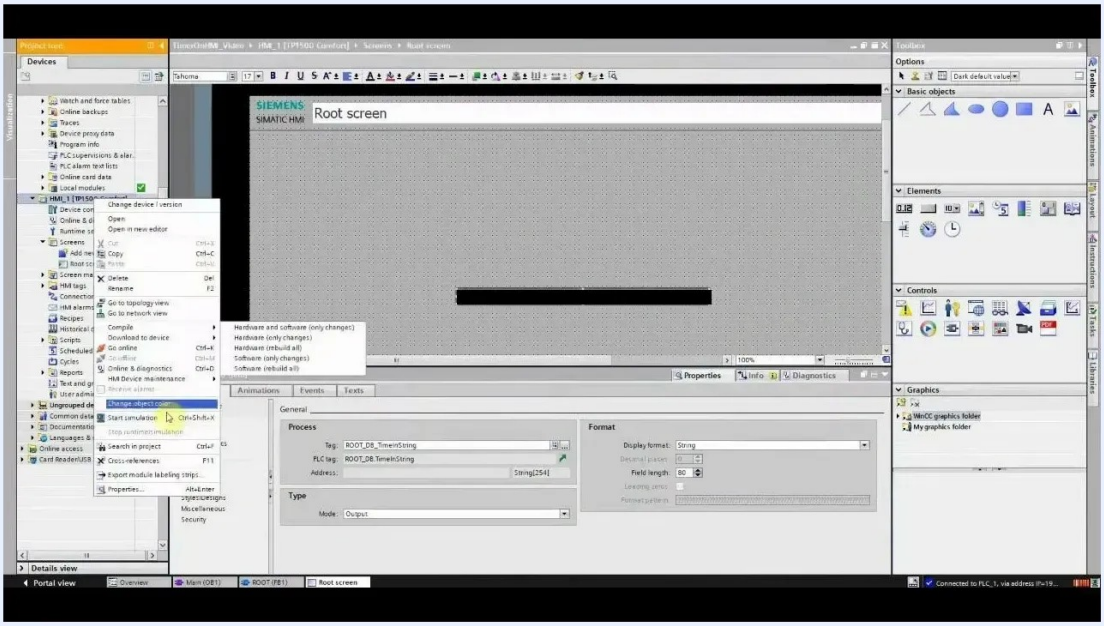
Step 31. Bind the I/O field to the PLC string tag and start the simulation

1. With the I/O field still selected, expand the **Process** section in the **Properties** panel.
2. Set **Tag** to the PLC data block variable (for example, ROOT_DB_TimeInString).
3. Set **Type** to **Output** so the field displays data from the PLC.
4. Set **Display format** to **String**.
5. Set **Field length** to 80, matching the expected string length.

The I/O field now shows a long string of zeros, which means it is ready to display the PLC string value.

6. In the project tree, right-click the HMI device (for example, HMI_1 [TP1500 Comfort]).

7. Select **Start simulation**, or press Ctrl+Shift+X, to launch the HMI runtime simulation.



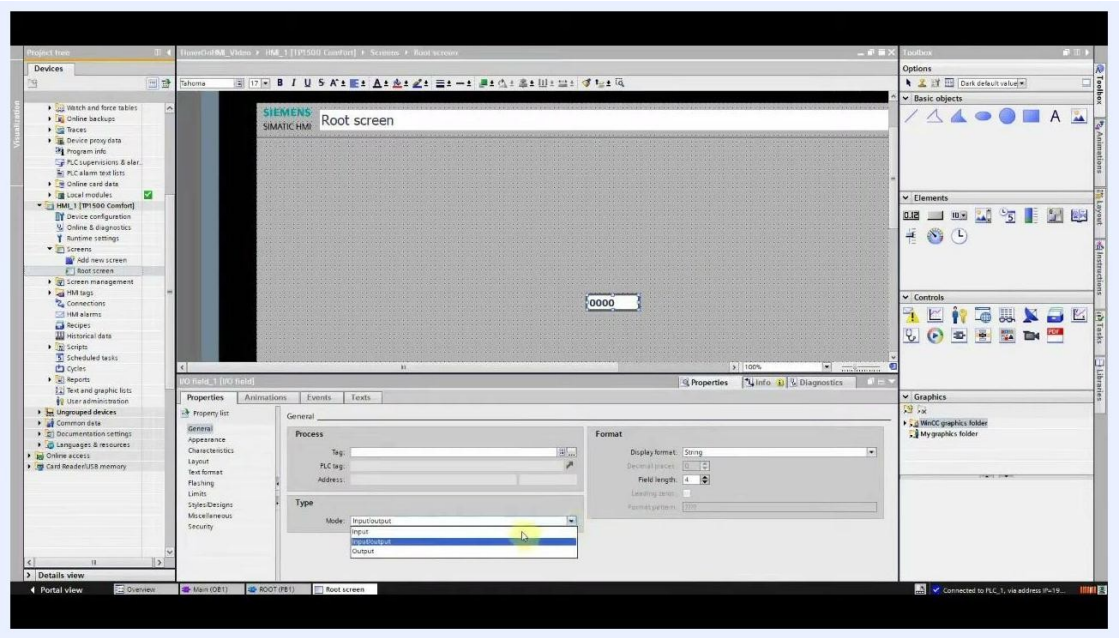
HMI screen editor with the I/O field bound to the PLC tag ROOT_DB_TimeInString, field length set to 80, and the simulation being started from the context menu

Step 32. Run the simulation and watch the running timer

The HMI runtime simulation window opens and displays the **Root screen**. The I/O field shows the live value from the PLC, such as 0D:0H:2M:8S:360mS. The simulation window includes:

- The "Root screen" label at the top.
- The current date and time in the upper right corner (for example, "05-Dec-17" and "22:09:30").
- The I/O field, centered on the screen, showing the formatted timer string.
- Navigation buttons at the bottom, including a home button.

Verify that the displayed value updates in real time and reflects the string value in the PLC data block.



HMI runtime simulation showing the Root screen with the I/O field displaying the live timer string from the PLC

Summary

You now know how to display a running timer on an HMI in TIA Portal. You added a TONR timer and its control variables, created an SCL function that converts LTime to a string, and defined its input, output, and temporary variables. You then integrated the function into the main program, downloaded the project to the PLC, and bound the data block string to an I/O field on the HMI root screen. The HMI simulation shows the timer string updating live.

What's next

- Compile and test the LTime_To_String function with your own preset times.
- Extend the main program with additional logic or blocks as your application requires.
- Add further visualization elements from the HMI **Toolbox** to interact with your PLC project.