Setting up communication between CPU and C/C++ runtime for a multifunctional platform using OPC UA

SIMATIC CPU 1518-4 PN/DP MFP

https://support.industry.siemens.com/cs/ww/de/view/109749176
Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (https://support.industry.siemens.com) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens’ products and solutions constitute one element of such a concept. Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the Internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.

Siemens’ products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer’s exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: https://www.siemens.com/industrialsecurity.

OPC UA für CPU 1518-4 PN/DP MFP
Entry ID: 109749176, V1.0, 06/2018
# Table of contents

Legal information .................................................................................................................. 2

1 Introduction .......................................................................................................................... 5

1.1 Introduction ......................................................................................................................... 5
1.2 Overview of the automation task ......................................................................................... 5
1.3 Mode of operation .................................................................................................................. 5
1.3.1 Principle of communication between CPU runtime and C/C++ runtime using OPC UA ........................................................................................................................................ 5
1.3.2 Call structure and program structure .............................................................................. 7
1.4 Components used .................................................................................................................. 8

2 Engineering ............................................................................................................................. 9

2.1 Engineering of the program for the CPU runtime ................................................................. 9
2.1.1 Functional description OB main ..................................................................................... 9
2.1.2 Functional description FB "increaser" .............................................................................. 10
2.1.3 OPC UA server settings ............................................................................................... 10
2.2 Engineering of the program for the C/C++ runtime ............................................................ 11
2.2.1 Creating the C/C++ project and explanation of the folder structure ................................ 11
2.2.2 Development of the C/C++ runtime application ............................................................ 12
2.2.3 Sources for the C/C++ Runtime Application ................................................................. 12
2.2.4 Basic structure of the C/C++ runtime application ......................................................... 13
2.3 Integration into the user project .......................................................................................... 15
2.3.1 Loading the TIA Portal project into the target system ................................................... 15
2.3.2 Downloading the C/C++ runtime application to the target system................................. 16
2.3.3 Configuring PuTTY ......................................................................................................... 16
2.3.4 Commissioning C/C++ runtime ..................................................................................... 18
2.3.5 Preparation and generation of the C program ............................................................... 20
2.3.6 Setting up a new connection to the target system in Eclipse ........................................ 20
2.3.7 Downloading the C/C++ runtime application to the target system, executing and terminating it .................................................................................................................. 22
2.4 Operation ............................................................................................................................ 24
2.4.1 Execute TIA portal project ............................................................................................ 24
2.4.2 Executing the C/C++ runtime application .................................................................... 25
2.4.3 Executing the application using Eclipse ........................................................................ 25
2.4.4 Executing the application using Secure Shell .............................................................. 25
2.4.5 Reading the system time ............................................................................................... 27
2.4.6 Setting up the counter value ......................................................................................... 28
2.4.7 Writing the counter value .............................................................................................. 29
2.4.8 Closing the OPC UA Client ......................................................................................... 31
2.5 Error handling .................................................................................................................... 32

3 Useful information .................................................................................................................. 34

3.1 Basics of the hardware and software components ............................................................... 34
3.1.1 SIMATIC ODK 1500S ..................................................................................................... 34
3.1.2 CPU 1518-4 PN/DP MFP .............................................................................................. 35
3.1.3 Interface assignment of the CPU 1518-4 PN/DP MFP ................................................... 35
3.2 Alternative solutions .......................................................................................................... 37
3.2.1 Downloading to the target system C/C++ runtime ....................................................... 37
3.2.2 Alternative communication via Open User Communication ........................................... 37

4 Appendix ............................................................................................................................... 38

4.1 Service and Support ............................................................................................................ 38
1 Introduction

4.2 Links and Literature ................................................................. 39
4.3 Change documentation ......................................................... 39
1 Introduction

1.1 Introduction

The integration of PC functionality into the controller program is often advantageous to the overall automation task. For example, compact automation solutions are implemented on a single device. In addition, complex functions and solutions for automation tasks are already available in high-level languages or are created in high-level languages.

For this purpose, for example, high-level language programs must communicate with the controller program via defined interfaces or be integrated. A connection of both worlds – high-level language and PLC – is achieved within the multifunctional platform CPU 1518(F)-4 PN/DP MFP.

1.2 Overview of the automation task

The task is to integrate high-level language applications in the SIMATIC PLC. Besides the STEP 7 blocks of the customary user program, the CPU 1518-4 PN/DP MFP can also execute functions (blocks) and applications that were programmed with C/C++.

The multifunctional platform gives you the option of having C/C++ code executed synchronously during the CPU cycle (via the CPU function library). In addition, the multifunctional platform can run C/C++ applications as separate applications in parallel with the CPU runtime.

In so doing, it may be necessary to arrange for a data exchange between the C/C++ runtime and the CPU runtime. This data exchange may be used, for example, to trigger responses on the other side or to provide necessary information. An initial step in the application example shows the configuration for the CPU runtime in the TIA Portal. The creation of the C/C++ Runtime application with Eclipse is then shown.

OPC UA is selected here as the communication mode.

1.3 Mode of operation

1.3.1 Principle of communication between CPU runtime and C/C++ runtime using OPC UA

The following figure depicts the communication between the CPU runtime and C/C++ runtime in the CPU 1518-4 PN/DP MFP. This relies on the client-server principle. In the application example the C/C++ runtime starts actively with the connection request and thus responds to the client. The Open Source OPC UA-Client ("open62541") used in included in the C-program. The CPU runtime provides the server and can in this case respond to requests from multiple clients. The OPC UA server is parameterized accordingly on the CPU through the TIA portal.
If a connection to the server was successfully established, values can be read and written. This is done in the C program using the appropriate “read” and “write” functions. Then the connection is terminated again by the client, whereby the “session” is released to be re-used. For simple handling in the application, it is also recommended to integrate error-handling in the C program.
1.3.2 Call structure and program structure

The figure below shows the previously described methods within the program flow. The cyclically running program in the CPU runtime starts by incrementing a “counter” variable in steps of 10 Hz. Then the CPU system time is saved in an instance of a specially created data type. This serves for subsequent monitoring. The OPC UA server for the CPU is ready when the CPU is switched on and with prior parameter setting.

While the program for the CPU runtime is cyclical, this is not the case for the C program. Here, only the "Run()" method is run through in the form of a continuous loop and is not ended until the connection is closed ("exit_Program()"). After establishing a connection to the server via the client with "connect()", the user has the choice between "read()", "write()" and "reset()" for certain variables. This is controlled through input at the console. To end the connection request "exit_Program()" is triggered, which causes the C program to terminate simultaneously.

Figure 1-2: Program execution between the CPU-runtime and C/C++ runtime

<table>
<thead>
<tr>
<th>CPU-Runtime (TIA-Project)</th>
<th>Flow</th>
<th>C/C++ Runtime (C-Programm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase counter()</td>
<td>Connect</td>
<td>connect()</td>
</tr>
<tr>
<td>Read time-of-day()</td>
<td>Read &amp; Write</td>
<td>read()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>run()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>main()</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Disconnect</td>
<td>exit_program()</td>
</tr>
</tbody>
</table>
1.4 Components used

This application example was created with the following hardware and software components:

Table 1-1: Hardware and software components

<table>
<thead>
<tr>
<th>Components</th>
<th>Number</th>
<th>Article number</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU 1518-4 PN/DP MFP</td>
<td>1</td>
<td>6ES7518-4AX00-1AB0</td>
<td>A CPU 1518F-4 PN/DP MFP can also be used</td>
</tr>
<tr>
<td>Open Development Kit 1500S V2.5</td>
<td>1</td>
<td>6ES7806-2CD03-0YA0</td>
<td>Provides the templates for developing the C/C++ runtime application, plus the Eclipse development environment.</td>
</tr>
<tr>
<td>PuTTY</td>
<td>1</td>
<td>Free software</td>
<td>Enables access to the C/C++ runtime.</td>
</tr>
<tr>
<td>STEP 7 Professional V15</td>
<td></td>
<td>6ES7822-1..05..</td>
<td>Enables the programming for the CPU runtime</td>
</tr>
</tbody>
</table>

This Application Example consists of the following components:

Table 1-2: Documents and projects included

<table>
<thead>
<tr>
<th>Components</th>
<th>File name</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executable application and projects</td>
<td>109749176_CPU1518MFP-OPCUA-CCpp_CODE_V10.zip</td>
<td>Folder includes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• TIA Portal project.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The project provides the OPC UA server and generates values</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Executable C/C++ runtime application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The program implements the OPC UA client reads and writes values.</td>
</tr>
<tr>
<td>Establishing communication between CPU</td>
<td>109749176_CPU1518Mfp_OpcUa_DOC_V1_en.pdf</td>
<td>This document</td>
</tr>
<tr>
<td>and C/C++ runtime for a multifunctional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>platform using OPC UA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2 Engineering

2.1 Engineering of the program for the CPU runtime

This section describes the structure of the TIA Portal project for implementation of the application example for the CPU runtime. More complex components will be specified in greater detail here.

2.1.1 Functional description OB main

OB main consists of 2 networks which clarify the two central functions of reading and writing for two variables in the application example.

Figure 2-1: Network overview of OB main

“Network 1” continuously increments a counter value. The “Increaser” component will be explained in due course.

“Network 2” reads the CPU system time and saves this in an instance of a UDT "typeRdSysT" created for this purpose.

Both variables are saved in the “Data” component and can be read via the OPC UA client. Write operations are also possible for the “counter” variable.
2.1.2 Functional description FB “increaser”

The FB “increaser” increments the integer counter value of the “counter” variable after a defined frequency.

The "LGF_Frequency" component from the LGF is used here, which emits a Boolean signal at a specified frequency, here 10 Hz. On evaluation of a positive edge, the counter is incremented.

NOTE

LGF is a library of general functions (LGF) for STEP 7 (TIA Portal) and S7-1200 / S7-1500. It's available at the following post: https://support.industry.siemens.com/cs/ww/en/view/109479728

Figure 2.2: Implementation of the FB “increaser”

2.1.3 OPC UA server settings

The CPU runtime provides the OPC UA server, this is parameterized in advance via the TIA portal.

Proceed as follows:
1. Select the CPU in “Device configuration”.
2. In the “Properties” tab, click on “OPC US” in the area navigation.
3. Under “Server > General”, activate the OPC UA server by checking the box for “Activate OPC UA server”.
4. Under “Server > Options > General”, set the port to “4840”.

Note

The parameter "Max session timeouts" can also be reduced, but this will cause the session to terminate more quickly if no request is sent via the OPC UA client. This approach is recommended in the event that resources are to be released.
2 Engineering of the program for the C/C++ runtime

The preparation and the handling of the development environment are explained in this section. Building on this, the core elements of the C program are described.

2.2.1 Creating the C/C++ project and explanation of the folder structure

The ODK 1500S installation is coming with a template for C/C++ runtime application. It enables you to create the project (in the documentation "C Program" or C/C++ runtime application) for the C/C++ runtime.

Make certain that the following plugins have been installed under Eclipse:
- C/C++ Remote Launch
- TCF Target Explorer
- TCF Remote System Explorer

Note: You can find information on installing plugins in the "STEP 7 (TIA Portal) Options Open Development Kit 1500S V2.5" programming and operating manual

Requirement

Open Development Kit (ODK) is installed.

Procedure

Engineering using the "Open Development Kit 1500S" ("ODK 1500S") in Eclipse provides you with templates that facilitate your work with the CPU 1518-4 PN/DP MFP. You can find these in the "ODK 1500S Templates" folder.

Figure 2-3: Selection of templates for MFP Linux application
When the template is used to create a new project, the folder structure shown in the figure is created, including the files under "Includes".

Figure 2-4: Folder structure after use of the template for MFP Linux application

```
<table>
<thead>
<tr>
<th>Project Explorer</th>
</tr>
</thead>
<tbody>
<tr>
<td>109749176_CPU1518MFP-OPCUA-CCpp_CODE_V1</td>
</tr>
<tr>
<td>- Binaries</td>
</tr>
<tr>
<td>- Includes</td>
</tr>
<tr>
<td>- C:\Program Files (x86)\Siemens\Automation\ODK1500S\V2.5\eclipse\build_tools\mf</td>
</tr>
<tr>
<td>- C:\Program Files (x86)\Siemens\Automation\ODK1500S\V2.5\eclipse\build_tools\mf</td>
</tr>
<tr>
<td>- C:\Program Files (x86)\Siemens\Automation\ODK1500S\V2.5\eclipse\build_tools\mf</td>
</tr>
<tr>
<td>- Archiv</td>
</tr>
<tr>
<td>- launch</td>
</tr>
<tr>
<td>- src</td>
</tr>
</tbody>
</table>
```

**Note**

The default user and the C/C++ application must not have root rights. Create a new user for executing the C/C++ application.

Depending on the type of programming in the C/C++ application, the CPU performance may be affected by jitter.

The customer is responsible for the C/C++ application and its know-how protection.

### 2.2.2 Development of the C/C++ runtime application

**Sources for the C/C++ Runtime Application**

The following figure presents an overview of the central components of the C/C++ runtime application for implementing an OPC UA client. The sources used are listed under "src".

Figure 2-5: Sources

```
<table>
<thead>
<tr>
<th>Package Explorer</th>
</tr>
</thead>
<tbody>
<tr>
<td>109749176_CPU1518MFP-OPCUA-CCpp_CODE_V1</td>
</tr>
<tr>
<td>- launches</td>
</tr>
<tr>
<td>- mfp1518_release</td>
</tr>
<tr>
<td>- src</td>
</tr>
<tr>
<td>109749176_CPU1518MFP-OPCUA-CCpp_CODE_V1.cpp</td>
</tr>
<tr>
<td>opc_client.cpp</td>
</tr>
<tr>
<td>opc_client.h</td>
</tr>
<tr>
<td>open62541.c</td>
</tr>
<tr>
<td>open62541.h</td>
</tr>
</tbody>
</table>
```
Note

To directly use the components of this application example, you must perform an import.

Ensure that only a "Main" method is present after importing the files.

You can find a description of the import here:


An additional option for using the program components provided consists of switching the workspace. An explanation of this can be found in Section 2.3.2, subsection "Preparation and generation of the C program"

The following table gives you an overview of the individual source files:

<table>
<thead>
<tr>
<th>File</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>opc_client.cpp</td>
<td>Contains the primary elements of the C program and is further specified in the next section.</td>
</tr>
<tr>
<td>opc_client.h</td>
<td>Header file for &quot;opc_client.cpp&quot;</td>
</tr>
<tr>
<td>open62541.c</td>
<td>Contains the implementation of the OPC UA client &quot;open61541&quot; in C.</td>
</tr>
<tr>
<td>open62541.h</td>
<td>Header file for &quot;open62541.c&quot;</td>
</tr>
<tr>
<td>109749176_CPU1518MFP-OPCUA-CCpp_CODE_V1</td>
<td>Source created through project generation. By default this includes the main method.</td>
</tr>
</tbody>
</table>

Basic structure of the C/C++ runtime application

After creating a new project, here named "109749176_CPU1518MFP-OPCUA-CCpp_CODE_V1", a corresponding file is created in "src". This can be used to implement your "main" method.

Due to the compact structure of the project, however, this is integrated directly in the source file "opc_client.cpp". The "main" method contained in the source file "109749176_CPU1518MFP-OPCUA-CCpp_CODE_V1.cpp" must, therefore, be removed or commented out.

In the C program, the OPC UA client is implemented using the open source component "open62541". Methods which are required accordingly, such as access to certain nodes, reading, writing, enabling and disabling connections, are made available in this way.

You will find the basic program flow under "opc_client.cpp".
Refer to the following table for the meaning of the various methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>main()</td>
<td>Forms the starting point of the C program. The start() method is called from here.</td>
</tr>
<tr>
<td>start()</td>
<td>The method checks the status of the OPC UA client instantiated and brings up the command prompt for the user. Then the</td>
</tr>
</tbody>
</table>
## 2.3 Integration into the user project

This section explains how to download the respective program components to the CPU 1518-4 PN/DP MFP. Along with the TIA Portal project, the C program must also be loaded and started with the help of an SSH client tool (e.g., PuTTY). This section also explains how to configure these tools and transfer the program.

### 2.3.1 Loading the TIA Portal project into the target system

The PROFINET interface X3 P1 is subdivided within the CPU 1518-4 PN/DP MFP for the CPU runtime and the C/C++ runtime. For this reason, separate IP addresses exist for the C/C++ runtime and CPU runtime.

- Enter the IP address of the CPU in STEP 7. You can find additional information in the STEP 7 online help.
- Set the IP address of the C/C++ runtime using the "network.sh" script in the "/etc/mfp/etc" directory of the C/C++ runtime.

Note the following limitations for the configuration of the PROFINET interface X3 P1 with STEP 7:

- In STEP 7, if you disable the option "Properties > General > PROFINET interface GBIT [X3] > Advanced options > Port options > Activate > Activate this port for use", then the PROFINET interface X3 P1 is disabled for the CPU and internal communication with the C/C++ runtime.
- The configuration of the "Transmission rate / duplex" has no effect on the connection to the PROFINET interface X3 P1 and the C/C++ runtime.
- The "Monitor" option is not supported.
2 Engineering

- The topology configuration is not supported.

Figure 2-7: Properties in the configuration of the connection

The following IP addresses are used in the application example:
- CPU runtime: 192.168.15.10
- C/C++ runtime: 192.168.15.18

Follow the usual procedure to download the TIA Portal project to the CPU 1518-4 PN/DP MFP using the specified IP addresses.

2.3.2 Downloading the C/C++ runtime application to the target system

Configuring PuTTY

Requirement
- You have successfully installed PuTTY.
- To establish a secure connection between Eclipse and the C/C++ runtime for the CPU 1518-4 PN/DP MFP, you need a configured SSH client (e.g. PuTTY).

Procedure
Procedure using PuTTY as an example
1. Start PuTTY.
2. Enter the destination address (default address: 192.168.15.18) in the entry field "Host Name (or IP address)".
3. Ensure that the following default settings are kept:
   - Port: 22
   - Connection type: SSH
4. To identify the PuTTY window and to associate the connection to the CPU in Eclipse, enter the title "CPU 1518MFP Linux Secure Connection" in the "Window title" entry field in the "Windows > Behavior" category.

5. Enter the following values in the "Connection > SSH > Tunnels" category:
   - Under "Source port": "1534" or "2345".
   - Under "Destination": "localhost:1534" or "localhost: 2345".

   Confirm each entry with "Add".
6. Enter "CPU-1518MFP-Linux-Secure-Connection" under "Saved Sessions" in the "Session" category and confirm with "Save".

7. To log onto the CPU 1518-4 PN/DP MFP, select the connection created and click "Open".

Commissioning C/C++ runtime

Requirement
- You have started the CPU 1518-4 PN/DP MFP.

Procedure
1. Start the Secure Shell Client (e.g. PuTTY).
2. Connect the Secure Shell Client to the CPU 1518-4 PN/DP MFP with the help of the PuTTY configuration "CPU 1518MFP Linux Secure Connection" using the destination address.
3. Enter the user name and password and establish a Secure Shell connection. The default user is "root". The default password is shown in the display under "Overview > MFP > Default Password:" ("Overview > MFP > Default > Password:").

4. After the initial startup of the CPU, change the default password.

5. Start the TCF Agent in the C/C++ runtime with the following command:
   `/usr/sbin/tcf-agent -d -l -t0 -sTCP:localhost`

6. Create a folder on the CPU 1518-4 PN/DP MFP, in directory "/home/<user>", in which you want to download the application. In the application example the directory structure "/home/root/OPCUA" was selected.

**Note**

To automatically start the TCF Agent (Target Communication Framework) on startup, enter the named command in the "autostart.sh" script file in the "/home/<user>" directory.

You can find information on creating an autostart.sh, changing the default address, and activating the DHCP (Dynamic Host Configuration Protocol) as well as other information in the "STEP 7 (TIA Portal) Options Open Development Kit 1500S V2.5" programming and operating manual (https://support.industry.siemens.com/cs/ww/en/view/109752683)
Preparation and generation of the C program

Requirement
- Eclipse is open

Procedure
In order to use the program components supplied with the application example, you have two options:
- Import the supplied sources: This option is more extensive and requires the creation of a new project, plus use of the template for CPU1518MFP.
- Switch the workspace to the desired project: This method directly uses all the data supplied and doesn’t require its own creation. It is described below and is recommended for use.

Switching the workspace:
1. In Eclipse, click on the “File” tab and select “Switch Workspace > Other”.
2. In the dialog which opens, navigate via “Browse” to the storage location for the project folder for the C/C++ runtime application "109749176_CPU1518MFP-OPCUA-CCpp_CODE_V1".
3. Select this folder, so that all sub-folders such as ".metadata" are included, and confirm by clicking OK.

Result
The workspace has been redefined.

Next you can generate the project as follows:
1. Select the project for the C/C++ runtime application.
2. Select the "Build Project" command in the "Project" menu on the system bar.

Note
The project data will only be generated if you have changed the files.

Setting up a new connection to the target system in Eclipse

Requirement
- An MFP project is created in Eclipse.
- An MFP project is generated in Eclipse.
- The TCF agent is started up.

Procedure
1. Select the "Run Configurations..." command in the "Run" menu on the system bar to open the dialog.
2. Configure your connection.
3. To set up a new connection, click the "New" button under "Connection" on the "Main" tab. The "New Connection" dialog opens.

4. Select "TCF" and confirm with "Next".

5. Fill in the dialog as shown in the following figure and confirm with "Finish".
6. Select the "localhost" connection under "Connections" in the "Run Configurations" dialog.
7. Apply the configuration settings with "Apply".

**Downloading the C/C++ runtime application to the target system, executing and terminating it**

**Procedure**

Transfer the C/C++ runtime application to the target system as follows:

1. Select the "Run Configurations..." command in the "Run" menu on the system bar of Eclipse.
   The "Run Configurations" dialog opens.
2. Select the desired configuration under "C/C++ Remote Application".
3. Execute the download operation with "Run".
4. Enter your password or the CPUs default password. Here the "User ID" is irrelevant.

**Figure 2-15: Password entry**
Result

The C/C++ runtime application has been downloaded to the CPU 1518-4 PN/DP MFP and executed.

Using Eclipse you can directly execute the application example via the “Console” without using PuTTY. However, to avoid possible errors by the TCF agent here with repeated execution, this approach is not recommended. So terminate the C/C++ runtime application as follows:

1. In the console in Eclipse, click on the end of the C/C++ runtime application.
2. Terminate the C/C++ runtime application by entering “x”.

Figure 2-16: Terminating the C/C++ runtime application via the console in Eclipse

Note

Additional information on the behavior of the TCF agent can be found in the “Interesting facts” section.
2.4 Operation

To operate the application example the execution sequence is irrelevant. However, both programs (TIA Portal project and C/C++ runtime application) must be loaded. The OPC UA server is active with CPU in “Run” mode.

In this case the TIA Portal is started first. Then the C/C++ runtime application is executed and all functions tested.

2.4.1 Execute TIA portal project

Requirement
- The TIA Portal project has been downloaded to the CPU 1518-4 PN/DP MFP.
- The CPU is set to “run” mode.

Procedure

Operating the application example in the TIA portal can be reduced to monitoring relevant variables:

1. Go online on the CPU runtime by selecting the CPU and clicking “GoOnline”.
2. Navigate to the “data” component and click “Monitor all”. (2) Alternatively you can use the watchtable to observe.
3. Observe how the “counter” variable is constantly incremented. (3)
4. Observe how the “sysTime” variable is updated with each cycle. (4)

Figure 2-17: Monitoring the variables

Note
Make sure that you have ticked the box “Accessible from HMI/OPC UA” for all the variables to be monitored.
2.4.2 Executing the C/C++ runtime application

There are two options for executing your C/C++ runtime application. These are presented in the following sections. Only use one of the two options for the execution.

The recommend option is to use PuTTY, "Execute application via secure shell", as already described in Section 2.3.2, subsection "Load, execute and terminate the C/C++ runtime application on the target system via Eclipse".

Executing the application using Eclipse

Requirement
- The "autostart.sh" file has been run. Alternatively TCF agent has been started manually.

Procedure
Regardless of whether the C/C++ runtime application is on the target system, you can start it directly using Eclipse.
To do so, follow the procedure described in section "Downloading the C/C++ runtime application to the target system and executing it using Eclipse".

Result
The application is executed on the target system.

Executing the application using Secure Shell

Requirement
- The C/C++ runtime application is located on the target system.
- The "autostart.sh" file has been run.

Procedure
1. Connect to the target system using PuTTY. To do so, use the PuTTY configuration "CPU 1518MFP Linux Secure Connection" defined in section 2.3.2, and confirm with "Open"
2. Enter the user name and password, as described in section 2.
3. Ensure that the C/C++ runtime application is on the C/C++ runtime. To do so, navigate to the folder in which it is stored and read out the files contained there using the "ls" command.
   In the application example, the C/C++ runtime application is stored under "/home/root/OPCUA".
4. To execute the C/C++ runtime application, enter the following command: 
   
<table>
<thead>
<tr>
<th>absolute directory path</th>
<th>[file name without file ending]</th>
</tr>
</thead>
<tbody>
<tr>
<td>/home/root/OPCUA/109749176_CPU1518MFP-OPCUA-CCpp_CODE_V10</td>
<td></td>
</tr>
</tbody>
</table>


2 Engineering

Figure 2-18: Executing the C/C++ runtime application

Result

The C/C++ runtime application starts, reports the status of the OPC UA client and gives options for subsequent program continuation. Then it waits for user input.

2.4.3 OPC UA client and connection to OPC UA server

Requirement

- The C/C++ runtime application has been executed

Procedure

1. After the C/C++ runtime application has been executed, enter “0” in the PuTTY console.
2. Observe how a submenu for subsequent program continuation opens up.

Figure 2-19: Menu and submenu after connection

Result

The OPC UA client connects with the OPC UA server. The submenu remains open
Note  After executing the command to connect the OPC UA client and OPC UA server, the execution sequence for subsequent commands is irrelevant. The same command may also be executed several times in turn.

2.4.4  Reading the system time

Requirement
- The C/C++ runtime application has been executed
- The OPC UA client status is “ready”
- The OPC UA client is connected

Procedure
1. Enter “5” at the PuTTY console, which issues the command to read the system time.
2. View the system time via the PuTTY console "Date and Time are"
3. View the system time at the TIA portal in the “data” component under the variable “sysTime” in parallel to the previous step.
4. Observe how the submenu is redisplayed and further user input is awaited.

Figure 2-20: Reading the system time

Note  Since only a snapshot of the system time is displayed via the PuTTY console, this only shows an identical value to the “sysTime” variable at the TIA portal for a brief period.

Result
A snapshot of the system time is displayed.
2.4.5 Reading the counter value

Requirement

- The C/C++ runtime application has been executed
- The status of the OPC UA client is “ready”
- The OPC UA client is connected

Procedure

1. Enter “6” in the PuTTY console, which issues the command to read the counter value.
2. View the counter value via the PuTTY console “ContinuousCount”.
3. View the counter value at the TIA portal in the “data” component under the variable “counter” in parallel to the previous step.
4. Observe how the submenu is redisplayed and further user input is awaited.

Figure 2-21: Reading the counter value

Result

A snapshot of the counter value is displayed.
2.4.6 Resetting the counter value

Requirement
- The C/C++ runtime application has been executed
- The status of the OPC UA client is “ready”
- The OPC UA client is connected

Procedure
1. Enter “7” in the PuTTY console, which issues the command to reset the counter value.
2. View the reset counter value via the PuTTY console “ContinuousCount”.
3. View the counter value at the TIA portal in the “data” component under the variable “counter” in parallel to the previous step.
4. Observe how the submenu is redisplayed and further user input is awaited.

Result
A snapshot of the reset counter value is displayed.
2.4.7 Writing the counter value

Requirement

- The C/C++ runtime application has been executed
- The OPC UA client status is “ready”
- The OPC UA client is connected

Procedure

1. Enter “8” in the PuTTY console, which issues the command to write the counter value.
2. Enter any integer counter value, in the example this is “99999”, and confirm by selecting “Enter”.
3. View the reset counter value via the PuTTY console “New ContinuousCount” in comparison to the old counter value of “Previous ContinuousCount”.
4. View the written counter value at the TIA portal in the “data” component under the variable “counter” in parallel to the previous step.
5. Observe how the submenu is redisplayed and further user input is awaited.

Result

A snapshot is displayed of the entered, old and written counter values.
2.4.8 Closing the OPC UA Client

**Requirement**

- The C/C++ runtime application has been executed
- The OPC UA client status is “ready”

**Procedure**

1. Enter “x” in the PuTTY console, which issues the command to end the OPC UA client.
2. Observe how the connection to the OPC UA server and the C/C++ runtime application is terminated.

**Result**

The C/C++ runtime application has been ended.
2.5 Error handling

As described in the previous sections, the C/C++ runtime application can be executed in two different ways:

- Execution directly via the console in Eclipse
- Execution via the PuTTY console

Error description

In the event that the first option is used (this is not recommended) there can be errors, if the C/C++ runtime application is not terminated again via the console in Eclipse.

The TCF agent is required for executing the application example. This is started up using the file “autostart.sh” or via a separate command.

The C/C++ runtime application executed via Eclipse causes the TCF agent to be blocked for the duration of the execution and it is only enabled again once this C/C++ runtime application has been terminated in Eclipse. So if the program is not properly terminated, the TCF agent remains permanently blocked.

If the C/C++ runtime application is executed again via Eclipse, the following error message will be shown by Eclipse: "Connection refused".

It doesn't matter if the intention was simply to load or to execute via Eclipse.

Figure 2-24: Error message Connection refused

<table>
<thead>
<tr>
<th>Problem Occurred</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Error Message" /></td>
</tr>
<tr>
<td>Error during file upload.</td>
</tr>
<tr>
<td>Connection refused: connect</td>
</tr>
</tbody>
</table>

Troubleshooting

1. Start PuTTY and connect as usual with the C/C++ runtime.
2. Enter the command “Top”, which displays all currently running processes.
3. Check that the TCF agent process is active (in the example, this is the process with PID 1766). This is blocked through the C/C++ runtime application which was started via Eclipse.
4. Terminate this process manually by entering the command “kill -1 [your PID]” (in the example: "kill -1 1766")
5. Restart the TCF agent by executing the autostart file again (in the example: "home/root/autostart.sh"). Alternatively, you can enter the command to execute the TCF agent directly:

```
/usr/sbin/tcf-agent -d -L -l0 -sTCP:localhost
```

Figure 2-26: Re-execution of the autostart.sh

6. Now execute the C/C++ runtime application again via PuTTY.
3 Useful information

3.1 Basics of the hardware and software components

3.1.1 SIMATIC ODK 1500S

With the "Open Development Kit ODK 1500S" (ODK) you create the C/C++ blocks (CPU function library for CPU runtime) and C/C++ runtime applications.

With the ODK you use the mechanisms of higher-level programming languages (e.g. object orientation) within a modern programming environment.

You program the following with the ODK:

- Blocks in C/C++, which are executed synchronously during the execution cycle of the CPU (CPU function library for CPU runtime)
- C/C++ runtime applications, which run on the C/C++ runtime in the CPU 1518-4 PN/DP MFP independent of the STEP 7 user program.

With C/C++ runtime applications you implement processes in parallel with the STEP 7 user program, such as for preprocessing or sending of data over Industrial Ethernet. A CPU can undertake several tasks simultaneously. This reduces the complexity of functions and the time needed for implementation.

You can reuse existing C/C++ algorithms. To continue using the existing technological know-how, you can integrate the existing C/C++ code using the Open Development Kit as follows:

- In the runtime environment of the CPU
- As C/C++ runtime applications in the CPU 1518-4 PN/DP MFP C/C++ runtime

After integration of the C/C++ sources you execute these on the CPU.

You can find a more detailed description of the Open Development Kit in the S7-1500 Open Development Kit 1500S programming and operating manual. The sections that describe the CPU function library for CPU runtime and the C/C++ runtime applications are relevant for the CPU 1518-4 PN/DP MFP.

Note

You can find the programming and operating manual of the S7-1500 Open Development Kit 1500S at:


For details on hardware and software components, refer to section 2 of the programming and operating manual.
3.1.2 CPU 1518-4 PN/DP MFP

The following figure shows that the CPU 1518-4 PN/DP MFP has a C/C++ runtime in addition to the CPU runtime.

Figure 3-1: Structure of CPU 1518-4 PN/DP MFP

The CPU 1518-4 PN/DP MFP has additional memory for C/C++ code and data:

Work memory for:
- STEP 7 user program
- The CPU function library for CPU runtime
- The C/C++ runtime applications

Load memory for:
- The STEP 7 user program, including CPU function library for CPU runtime
- The C/C++ runtime applications

The possible connections between the two runtimes are presented in the following sections.

3.1.3 Interface assignment of the CPU 1518-4 PN/DP MFP

To gain access to the respective runtime, corresponding interfaces must be used. Of particular relevance to the application example is interface X3 P1, from which both the CPU runtime and the C/C++ runtime can be controlled. See the illustrations below for clarification.
The CPU 1518-4 PN/DP MFP has the following interfaces:

Table 3-1: Interface properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROFINET IO</td>
<td>The interface has two ports. It supports basic PROFINET functionality as well as PROFINET IO RT (Real-Time) and IRT (Isochronous Real-Time).</td>
<td>PROFINET function manual (<a href="https://support.industry.siemens.com/cs/ww/en/view/49948856">https://support.industry.siemens.com/cs/ww/en/view/49948856</a>)</td>
</tr>
<tr>
<td>PROFINET interface (X2 P1)</td>
<td>The interface has one port. It supports basic PROFINET functionality as well as PROFINET IO RT (Real-Time).</td>
<td>S7-1500 ODK 1500S manual (<a href="https://support.industry.siemens.com/cs/ww/en/view/109752683">https://support.industry.siemens.com/cs/ww/en/view/109752683</a>)</td>
</tr>
</tbody>
</table>
| PROFINET interface (X3 P1) | The interface is used for:  
- Connection of development tools for C/C++ applications  
- Connection of the TIA Portal for development of the STEP 7 applications  
- Communication of the C/C++ runtime with the "outside world"  
- Internal communication between C/C++ and CPU runtime (via a virtual network) | |
| PROFIBUS DP | This interface is used for connection to a PROFIBUS network. | PROFIBUS function manual (https://support.industry.siemens.com/cs/ww/en/view/59193579) |
| Operation of the CPU as DP master | In the role of DP master, the CPU addresses the connected DP slaves. The CPU cannot assume the role of a DP slave. | |
3 Useful information

3.2 Alternative solutions

3.2.1 Downloading to the target system C/C++ runtime

In this application example, the C/C++ runtime application is downloaded to the target system directly using Eclipse. Alternatively, other solutions such as use of WinSCP (Windows Secure Copy) are conceivable here. WinSCP is a free SFTP and FTP client software for Windows.

3.2.2 Alternative communication via Open User Communication

In the application example, the communication mode between both runtimes uses a client-server principle via OPC UA. This gives OPC UA the benefit of being “cross-platform” and means it is not tied to either the operating system or the programming language. However, other forms of communication are possible.

Another option is to use open user communication via TCP/IP. This establishes the connection partner on the CPU runtime through the "TCON", "TSEND", "TRCV", "TDISCON" components. Corresponding connection partners on the C/C++ runtime are implemented directly in the C/C++ runtime application.

Figure 3-3: Principle of OUC between the CPU runtime and C/C++ runtime

Note

Additional information on setting up open user communication for CPU1518MFP can be found at:

4 Appendix

4.1 Service and Support

Industry Online Support
Do you have any questions or need assistance?
Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.
The Industry Online Support is the central address for information about our products, solutions and services.
Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks:
https://support.industry.siemens.com

Technical Support
The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts. Please send queries to Technical Support via Web form:
www.siemens.com/industry/supportrequest

SITRAIN – Training for Industry
We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that’s tailored to the customer’s specific needs.
For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page:
www.siemens.com/sitrain

Service offer
Our range of services includes the following:
- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts
You can find detailed information on our range of services in the service catalog web page:
https://support.industry.siemens.com/cs/sc

Industry Online Support app
You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for Apple iOS, Android and Windows Phone:
https://support.industry.siemens.com/cs/ww/en/sc/2067
4 Appendix

4.2 Links and Literature

Table 4-1

<table>
<thead>
<tr>
<th>No.</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Siemens Industry Online Support</td>
</tr>
<tr>
<td></td>
<td><a href="https://support.industry.siemens.com">https://support.industry.siemens.com</a></td>
</tr>
<tr>
<td>2</td>
<td>Link to the entry page for the Application Example</td>
</tr>
<tr>
<td></td>
<td><a href="https://support.industry.siemens.com/cs/ww/de/view/109749176">https://support.industry.siemens.com/cs/ww/de/view/109749176</a></td>
</tr>
<tr>
<td>3</td>
<td>Programming and Operating Manual SIMATIC S7-1500 ODK 1500S</td>
</tr>
<tr>
<td>4</td>
<td>Device Manual SIMATIC S7-1500 CPU 1518-4 PN/DP MFP</td>
</tr>
</tbody>
</table>

4.3 Change documentation

Table 4-2

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>06/2018</td>
<td>First version</td>
</tr>
</tbody>
</table>