
# Open Controller Ready4Linux, OPC UA via virtual interface

Open Controller, Ready4Linux, OPC UA

https://support.industry.siemens.com/cs/ww/en/view/109779253

Siemens Industry Online Support

# 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.

# Table of contents

# 1 Introduction

## 1.1 Introduction

The integration of PC functionality into the control 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 control program via defined interfaces or be integrated. A combination of both worlds – high-level language and PLC – is achieved with the CPU 1515SP PC2 Ready4Linux (referred to as Open Controller) and CPU 1505SP (F) (referred to as CPU).

## 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 can also execute functions (blocks) and applications that were programmed with C/C++.

The CPU gives you to option of having C/C++ code executed synchronously during the CPU cycle (via the CPU function library). Additionally, C/C++ applications can be executed on the Open Controller as separate applications parallel to the CPU.

It may be necessary to exchange data between both applications, the C/C++ application, and the CPU. 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 in the TIA Portal. The creation of the C/C++ application with Eclipse is then shown. OPC UA is selected here as the communication mode.

You can find additional application examples can be found in Industry Online Support:

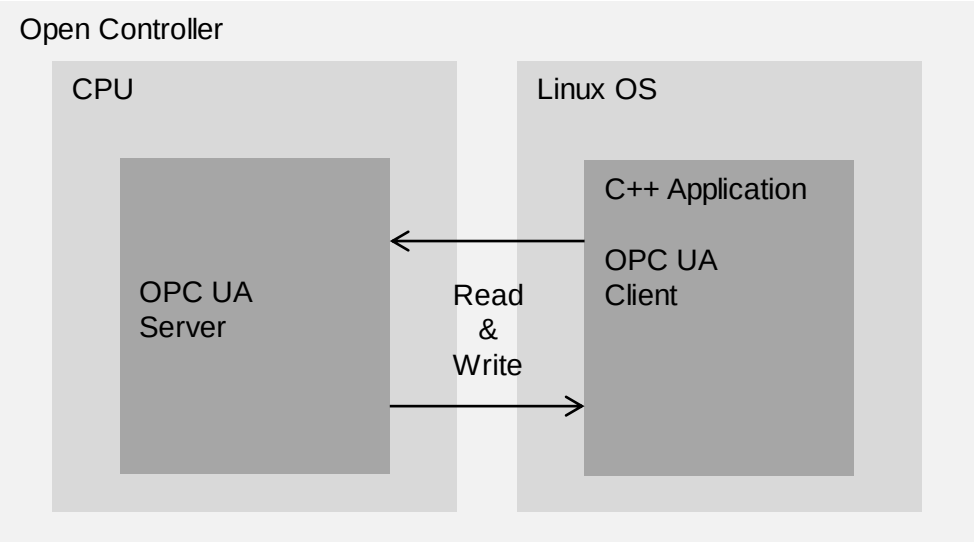
- Internal and external OPC-UA connection via the virtual Ethernet interface of the Software Controller from V2.5
  https://support.industry.siemens.com/cs/ww/en/view/109760541
- Setting up communication between CPU and C/C++ runtime for a multifunctional platform using OPC UA
  https://support.industry.siemens.com/cs/ww/en/view/109749176
- Establishment of Open User Communication between CPU runtime and C/C++ runtime of a multifunctional platform
  https://support.industry.siemens.com/cs/ww/en/view/109756757

## 1.3 Principle of operation

**Principle of communication between CPU and C/C++ application via OPC UA**

The following Figure shows the communication between CPU and C/C++ application on the Open Controller. This relies on the client-server principle. In the application example the C/C++ application 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 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.

Figure 1-1




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.

## 1.4 Components used

The Application Example has been created with the following hardware and software components:

Table 1-1: Hardware and software components

| Component | Quantity | Article number | Note |
|---|---|---|---|
| CPU 1515SP PC2 F (Ready4Linux) | 1 | 6ES7677-2SB40-0GB0 | The use of a CPU 1515SP PC2 (Ready4Linux) (6ES7677-2DB40-0GB0) is also possible |
| CPU 1505SP F | 1 | 6ES7672-5SC11-0YA0 | The use of a CPU 1505SP (6ES7672-5DC11-0YA0) is also possible |
| STEP7 Prof V16 | | 6ES7822-1A.06-.. | Enables the programming for the CPU |

This application example consists of the following components:

Table 1-2: Documents and projects included

| File name | Note |
|---|---|
| 109779253_CPU1515SP_Ready4LinuxOpcUa_CODE_V10.zip | TIA Portal project and C program |
| 109779253_CPU1515SP_Ready4LinuxOpcUa_DOC_V10_en.pdf | This document |

# 2 Engineering

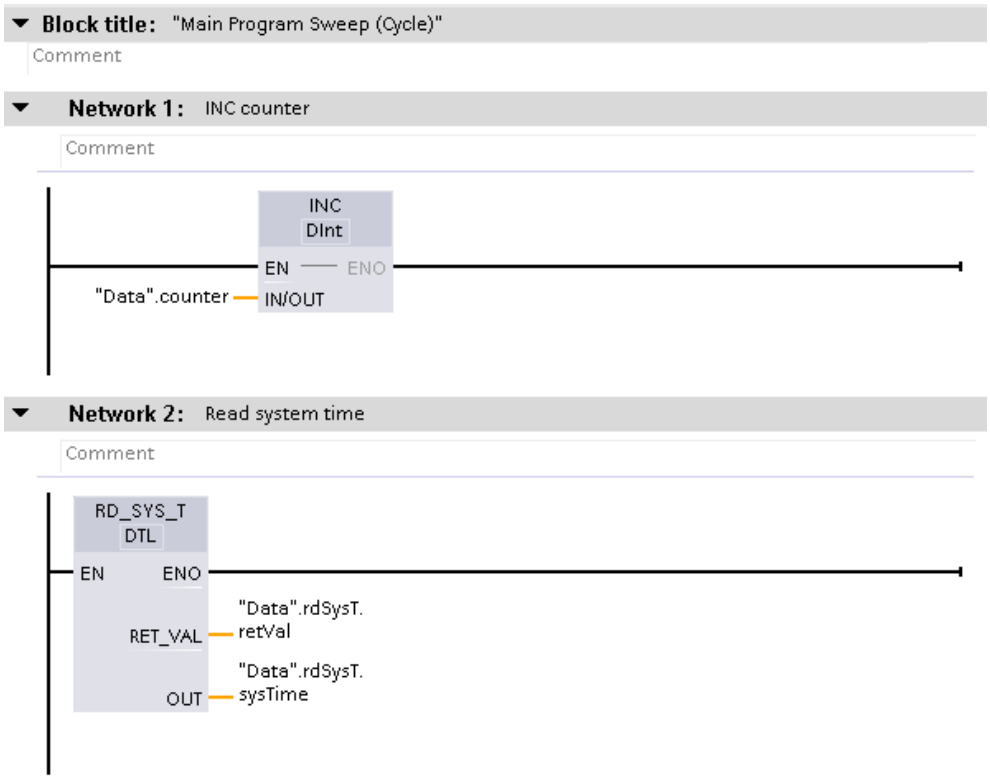
## 2.1 Engineering of the program for the CPU

This section describes the structure of the TIA Portal project for implementation of the application example for the CPU. 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 1” continuously increments a counter value.

“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” data block and can be read via the OPC UA client.

### 2.1.2 OPC UA server settings

**Note** The OPC UA Server is already activated in the supplied TIA Portal sample project.

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

To do this, 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”.
5. Under “Server > General”, set “Max session timeout” to “600000” s.

**Note** The parameter “Max session timeout” 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 procedure is recommended in case resources need to be released again.

## 2.2 Engineering of the program for the C/C++ application

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.

In this application example, the following is assumed:

- Linux OS is installed on the Open Controller (e.g. Debian 9)
- Virtual interface is set up in Linux OS (e.g. IP: 192.168.73.73)
- Eclipse IDE is installed (optional)

**Note** You can find additional information about the Open Controller Ready4Linux under https://support.industry.siemens.com/cs/ww/en/view/109769991

**Note** You can find a description of how to install Eclipse on a Linux system in this forum post, for example: https://support.industry.siemens.com/tf//ww/en/posts/how-to-use-iot2050-sdk-with-eclipse/239070/?page=0&pageSize=10

**Note** You can also develop the C program on another system and just run your application on the Open Controller. In this case, Eclipse does not need to be installed on the Open Controller.

How to run an application, such as the supplied “Ready4LinuxProj”-“binary”, on the Open Controller is described below.



### 2.2.1 Create a C/C++ project

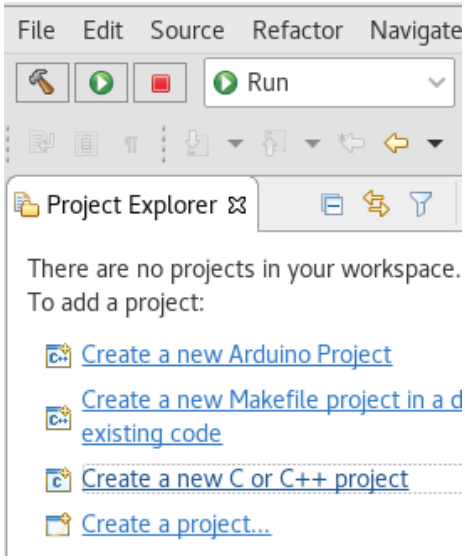
**Requirement**

- Linux OS is installed
- Eclipse is installed

**Procedure**

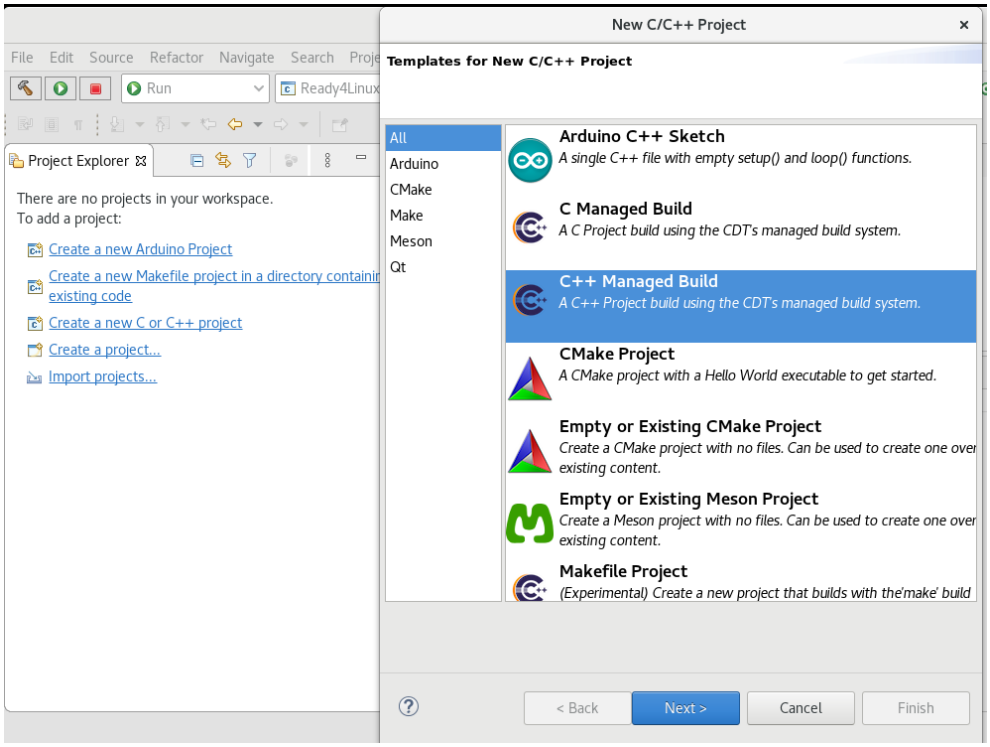
Start Eclipse and create a new C++ project e.g. by clicking “Create a new C or C++ project”.

Figure 2-2



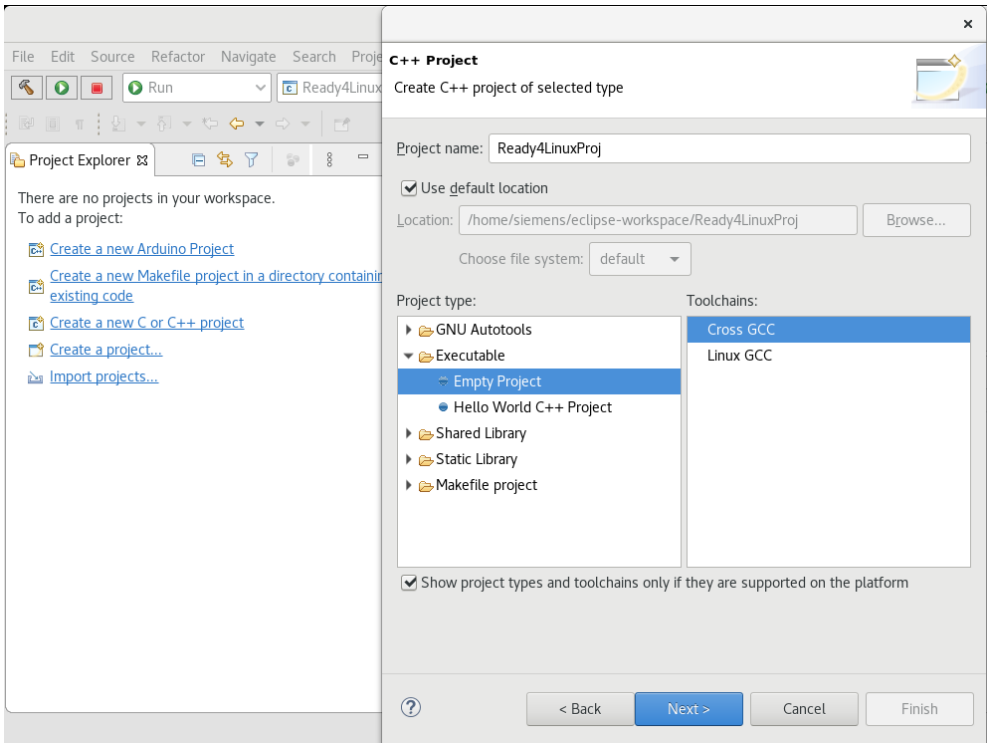

Select the “C++Managed Build” template and click “Next”.

Figure 2-3




Enter the project name and click “Next”.

Figure 2-4

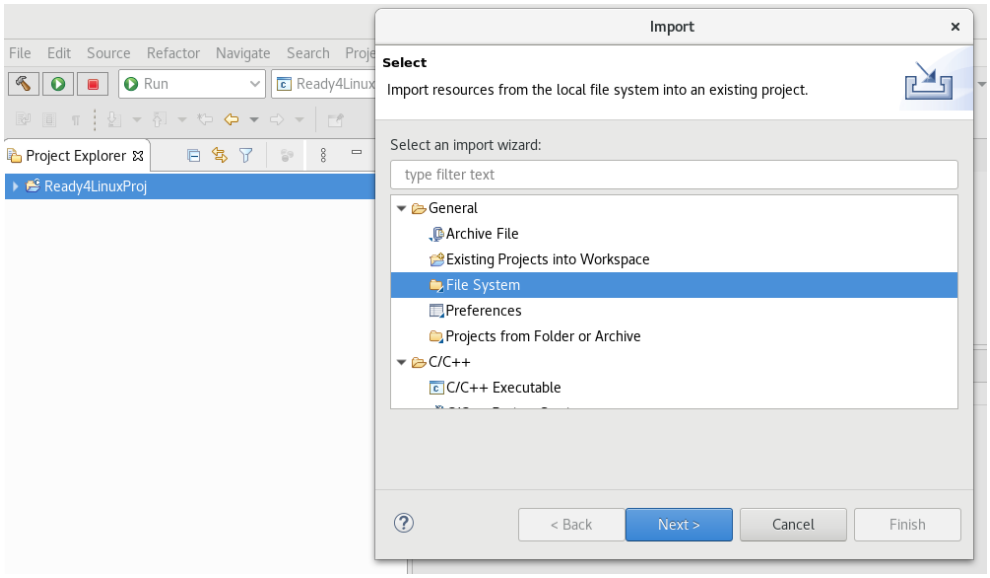



In the following dialogs, you can keep the default settings and use “Next” and “Finish” buttons to create your project.

### Use included sources

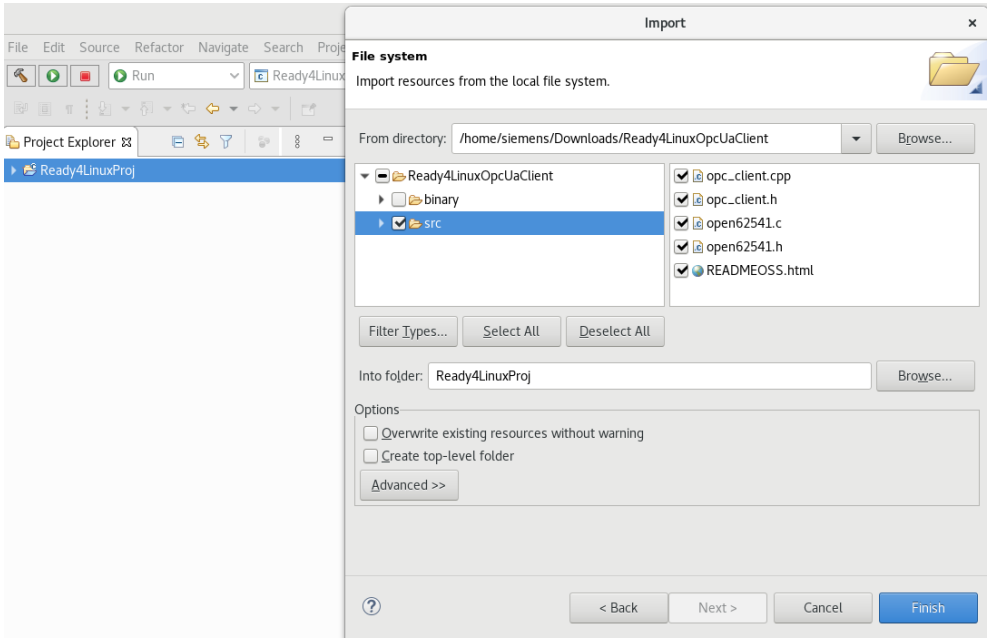
Import the supplied files into your C++ project, e.g. via the menu “File > Import”. Select “File System” and click “Next”.

Figure 2-5



Select the “src” directory and click “Finish”.

Figure 2-6




The files are displayed in the Project Explorer.

**Note**

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

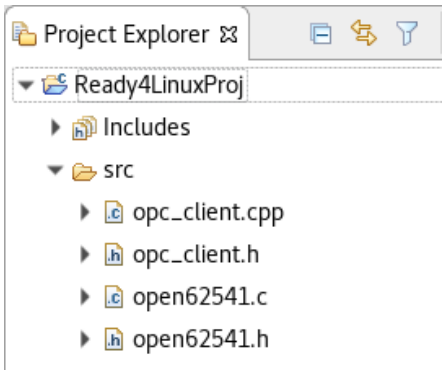
The customer is responsible for the C/C++ application and its know-how protection.

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

### Sources for the C/C++ application

The following Figure presents an overview of the central components of the C/C++ application for implementing an OPC UA client. E.g. the used sources are listed under “src”.

Figure 2-7



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

Table 2-1: Components contained in “src”

| File | Meaning |
|---|---|
| opc_client.cpp | Contains the primary elements of the C program and is further specified in the next section. |
| opc_client.h | Header file for “opc_client.cpp” |
| open62541.c | Open Source components “open62541”. Contains the implementation of the OPC UA client “open61541” in C. |
| open62541.h | Open Source components “open62541”. Header file for “open62541.c” |

### Basic framework of the C/C++ application

In the C program, the OPC UA client is implemented using the Open Source components “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 can find the basic program flow under “opc_client.cpp”. Refer to the following table for the meaning of the various methods:



Table 2-2: Central methods of the program execution

| Method | Meaning |
|---|---|
| main() | Forms the starting point of the C program. The start() method is called from here. |
| start() | The method checks the status of the OPC UA client instantiated and brings up the command prompt for the user. Then the run() method is called. |
| main_menu()<br>run_Menu() | Both methods serve to display the command prompt to the user. All available options in the application example are shown here. |
| run() | The run() method is run in the form of a loop until the end of the program. The input entered previously by the user is processed at this point and further methods are invoked as appropriate. |
| read_continuous_counter() | This method reads the “counter” variable. The string “Data”.counter is used here as OPC UA Node. |
| reset_continuous_counter() | The method resets the “counter” variable to 0. The string from the “data.counter” variable is used as the OPC UA node here. |
| write_continuous_counter() | The method sets the “counter” variable to a value entered by the user. The string from the “data.counter” variable is used as the OPC UA node here. |
| read_tod() | The method reads the system time. The constant is used here as the OPC UA node: UA_NS0ID_SERVER_SERVERSTATUS_CURRENTTIME. |

## 2.3 Integration into the user project

This section explains how to download the respective program components to the CPU.

The following IP addresses are used in the application example:

- CPU:
  - PROFINET: 192.168.1.130
  - virtual interface
    “Runtime communication interface”: 192.168.73.1
- Linux OS:
  - virtual interface
    “Siemens AG Ethernet”: e.g. 192.168.73.73

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

Deploy the TIA Portal project onto the CPU using the specified IP address via the PROFINET interface.

### 2.3.2 C/C++ Application

**Note** The following describes how to run the supplied “Ready4LinuxProj”-“binary” on the Open Controller. In this case, it is not necessary to create the application with Eclipse.



#### Creating a C/C++ application

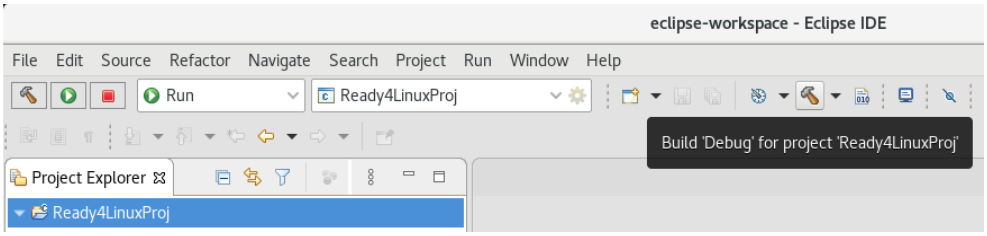
#### Requirement

- Eclipse is open

#### Procedure

For example, you can create your application by clicking on “Build ‘Debug’ for project ……”

Figure 2-8



# 3 Operation

To use the application example, we will start with the TIA Portal. Then the C/C++ application will be executed and all functions tested.

## 3.1 Execute TIA Portal project

### Requirement

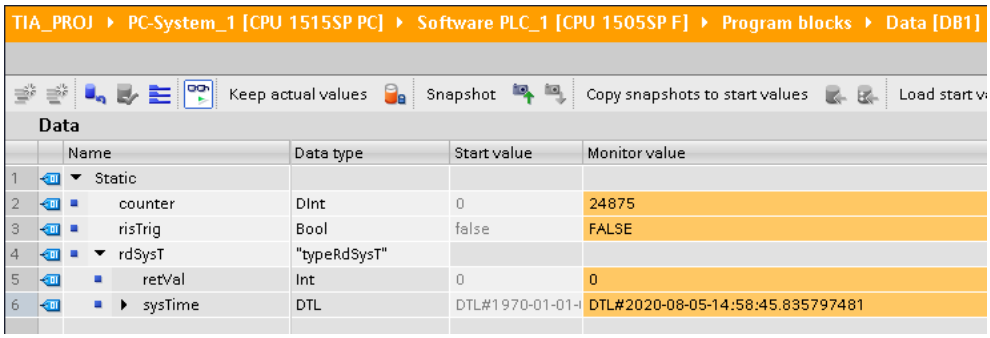
- TIA Portal project was loaded onto the CPU.

### Procedure

You can monitor the variables in the TIA Portal: Proceed as follows:

1. Go online on the CPU by selecting the CPU and clicking “GoOnline”.
2. Navigate to the “data” data block and click “Monitor all”. Alternatively, you can also perform the observation using the Observation Table.
3. Observe how the “counter” variable is constantly incremented.
4. Observe how the “sysTime” variable is updated with each cycle.

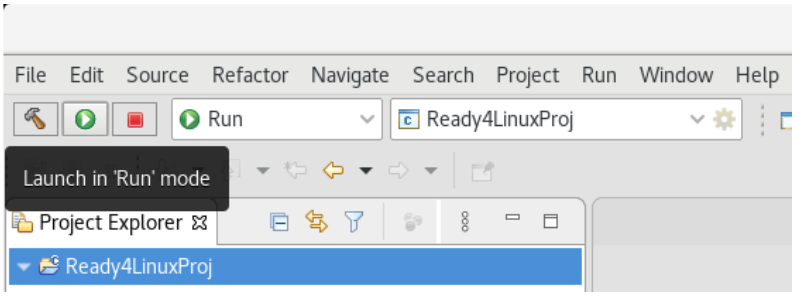
Figure 3-1




## 3.2 Executing the C/C++ application

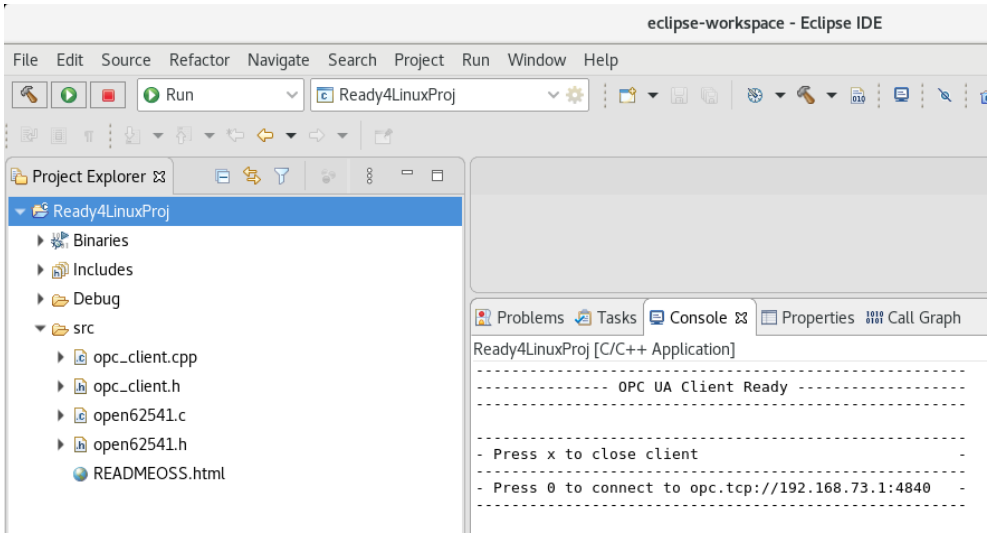
Start the C/C++ application in Eclipse e.g. by clicking “Launch in ‘Run’ mode”.

Figure 3-2



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.

Figure 3-3




## 3.3 OPC UA client and connection to OPC UA server

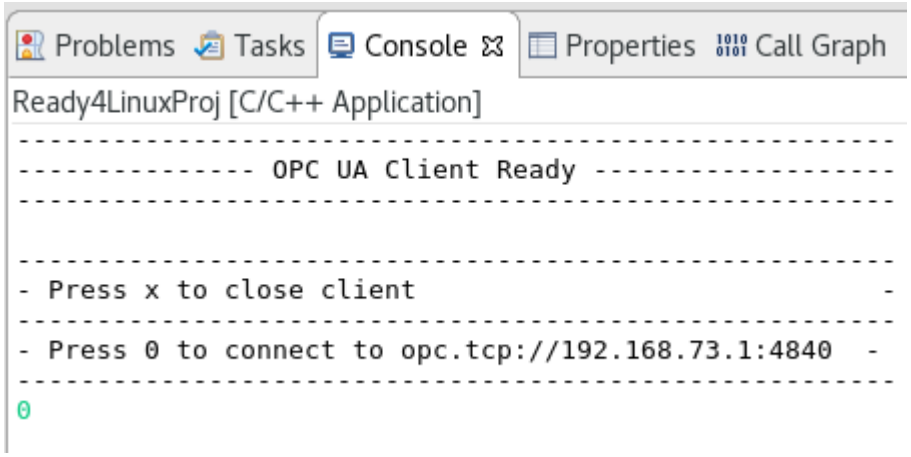
**Requirement**

- The C/C++ application has been executed

**Procedure**

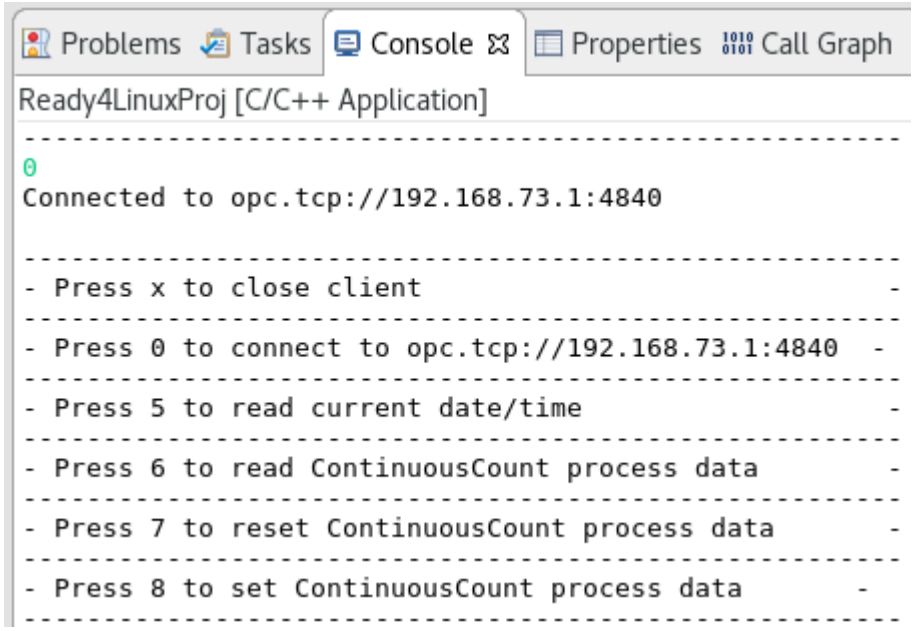
1. After executing the C/C++ application enter “0” in the console.

Figure 3-4



2. Observe how a submenu for subsequent program continuation opens up.

Figure 3-5




**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.

## 3.4 Reading the system time

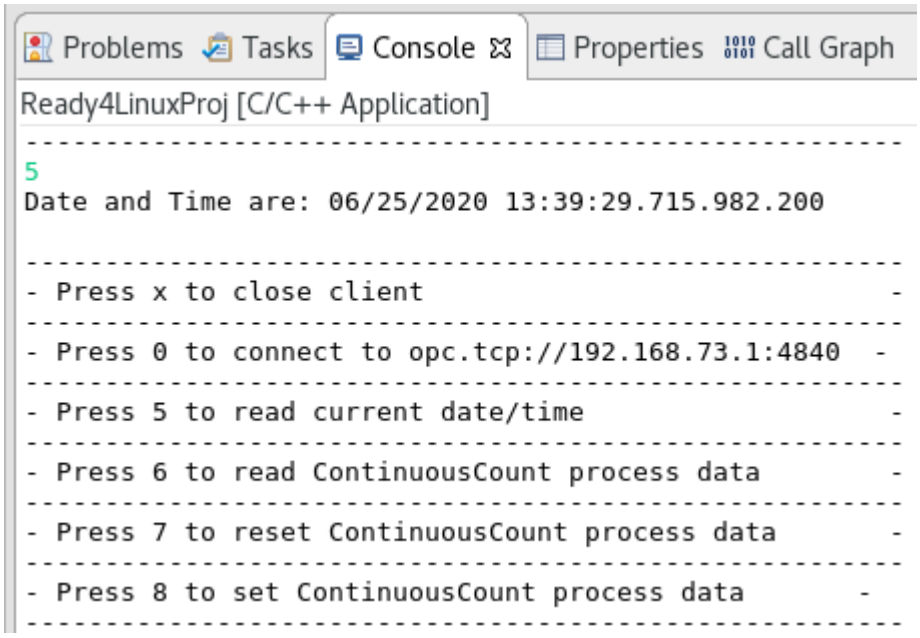
**Requirement**

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

**Procedure**

1. Enter “5” at the console, which issues the command to read the system time.
2. View the system time via the 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 3-6




**Note** Since only a snapshot of the system time is displayed via the 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.

## 3.5 Reading the counter value

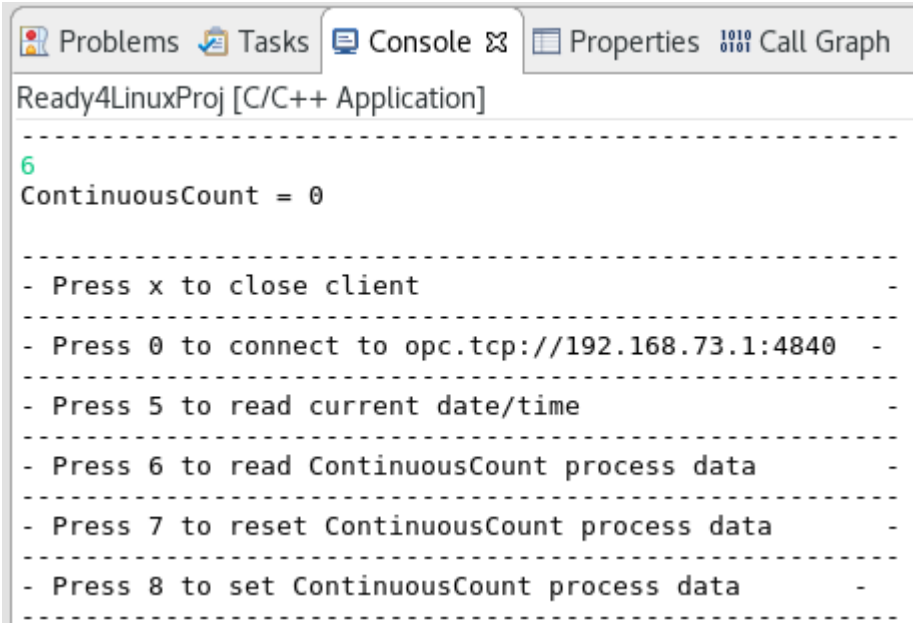
**Requirement**

- The C/C++ 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 console, which issues the command to read the counter value.
2. View the counter value via the console “ContinuousCount”.
3. View the counter value in the TIA Portal in the “data” block under the variable “counter” in parallel to the previous step.
4. Observe how the submenu is redisplayed and further user input is awaited.

Figure 3-7




**Result**

A snapshot of the counter value is displayed.

## 3.6 Resetting the counter value

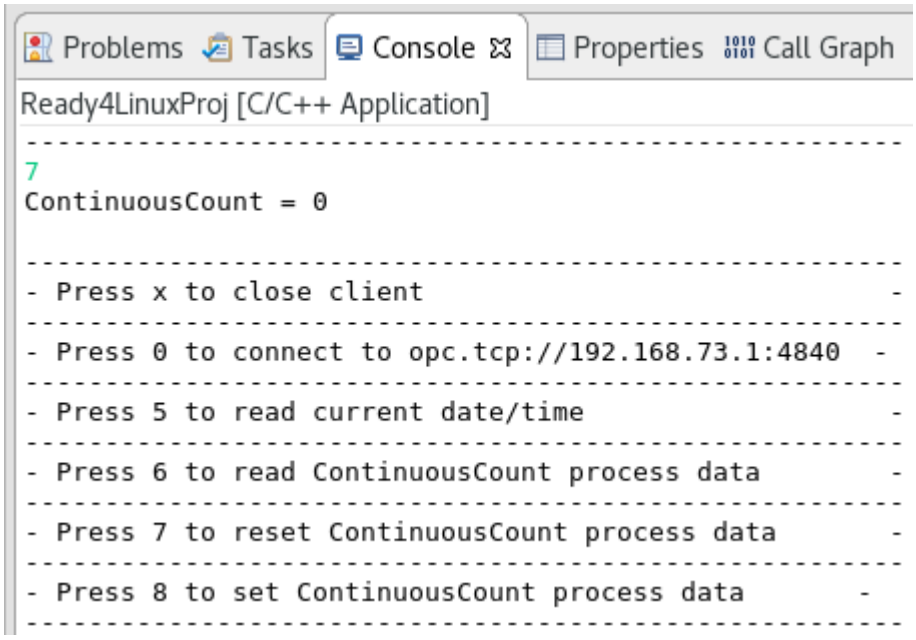
**Requirement**

- The C/C++ 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 console, which issues the command to reset the counter value.
2. View the reset counter value via the console “ContinuousCount”.
3. View the counter value in the TIA Portal in the “data” block under the variable “counter” in parallel to the previous step.
4. Observe how the submenu is redisplayed and further user input is awaited.

Figure 3-8




**Result**

A snapshot of the reset counter value is displayed.

## 3.7 Writing the counter value

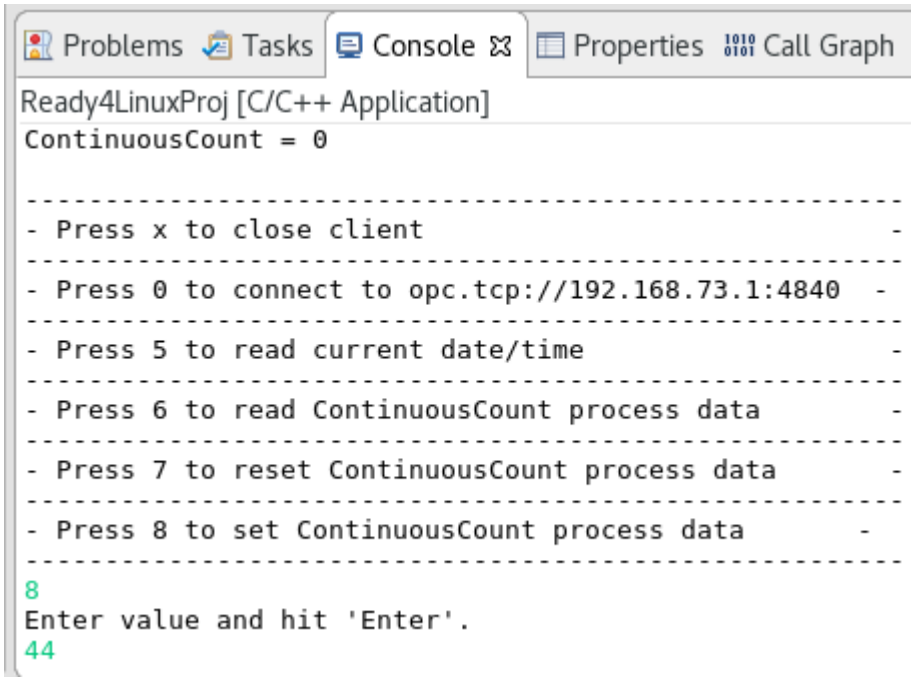
**Requirement**

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

**Procedure**

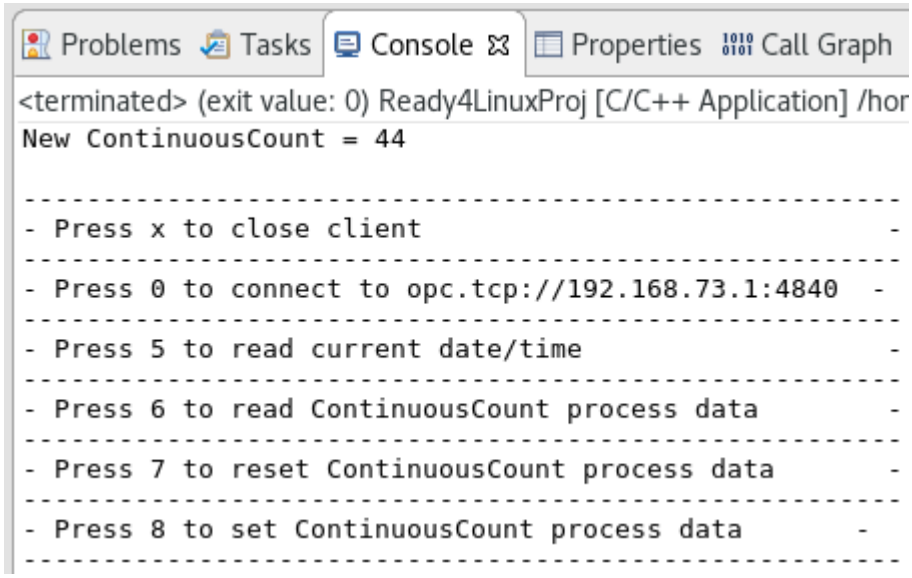
1. Enter “8” in the console, which issues the command to write the counter value.
2. Enter any integer counter value, in the example this is “44”, and confirm by selecting “Enter”.

Figure 3-9




3. View the reset counter value via the console “New ContinuousCount” in comparison to the old counter value of “Previous ContinuousCount”.
4. View the written counter value in the TIA Portal in the “data” block under the variable “counter” in parallel to the previous step.
5. Observe how the submenu is redisplayed and further user input is awaited.

Figure 3-10




**Result**

A snapshot is displayed of the entered, old and written counter values.

## 3.8 Closing the OPC UA Client

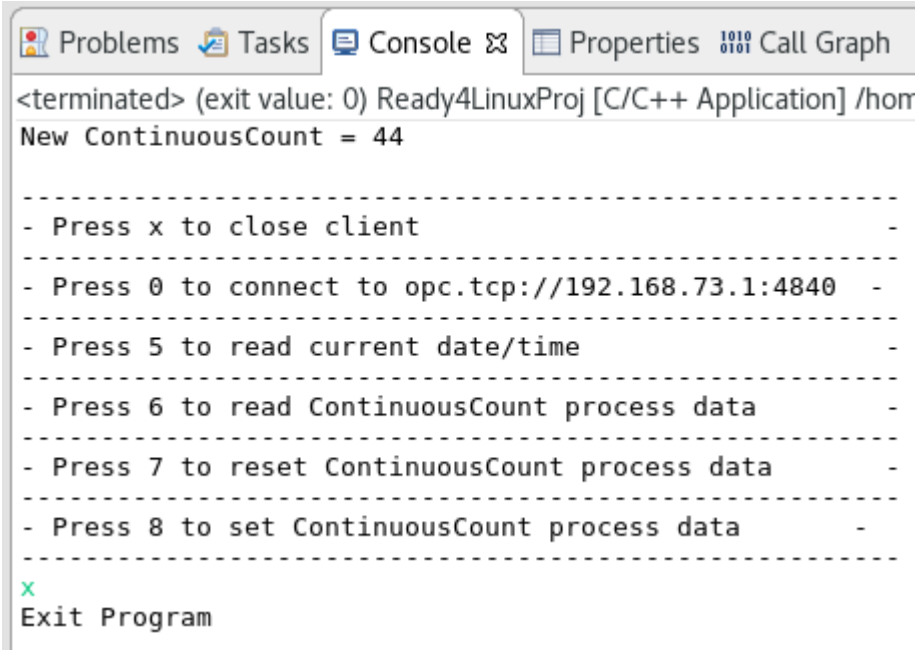
**Requirement**

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

**Procedure**

1. Enter “x” in the 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++ application is terminated.

Figure 3-11



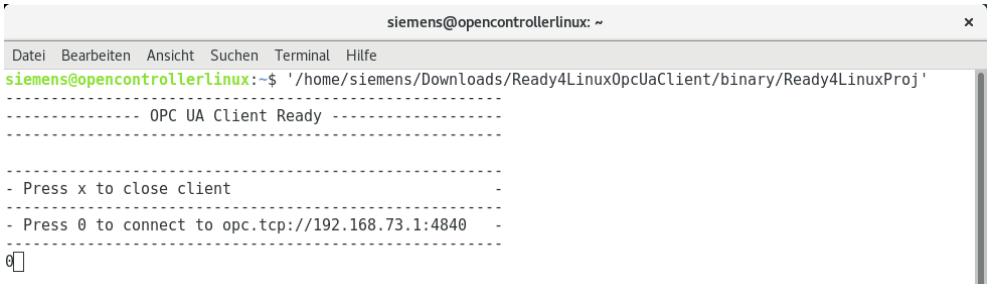

**Result**

The C/C++ application has been ended.

## 3.9 Use “Ready4LinuxProj”-“binary”

You can run the supplied “binary” without Eclipse on the Open Controller. E.g. drag & drop the file “Ready4LinuxProj” into the terminal window and confirm with “Enter”.

Figure 3-12




The operation and calling of the functions are as described above.

# 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:

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:

support.industry.siemens.com/cs/my/src

**SITRAIN – Digital Industry Academy**

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:

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:

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 iOS and Android:

support.industry.siemens.com/cs/ww/en/sc/2067

## 4.2 Links and literature

Table 4-1

| No. | Subject |
|---|---|
| \1\ | Siemens Industry Online Support<br>https://support.industry.siemens.com |
| \2\ | Link to the article page of the application example<br>https://support.industry.siemens.com/cs/ww/en/view/109779253 |
| \3\ | |

## 4.3 Change documentation

Table 4-2

| Version | Date | Change |
|---|---|---|
| V1.0 | 10/2020 | First edition |
| | | |