SIEMENS
Ingenuity for life

TIA Add-Ins Getting Started - Startdrive

TIA Portal as of V16 / SINAMICS Startdrive as of V16

https://support.industry.siemens.com/cs/ww/en/view/109779415

Siemens Industry Online Support

Legal information

Using the application examples

The application examples illustrate the solution of automation tasks with the interaction of several components in the form of text, graphics and/or software blocks. The application examples are a free-of-charge service provided by Siemens AG and/or a subsidiary company of Siemens AG ("Siemens"). They are non-binding and do not claim to be complete and functional with regard to configuration and equipment. The application examples do not represent customer-specific solutions, but are designed to provide help for typical jobs. You are responsible for the proper and safe operation of products in compliance with the applicable regulations. You must check the function of the respective application example and adapt it specifically to your system.

Siemens grants you the non-exclusive, non-sub-licensable and non-transferable right of use of the application examples by professionally trained personnel. Any change to the application examples is made at your own risk. Transfer to third parties or duplication of the application examples or extracts thereof is only permitted in combination with your own products. The application examples do not necessarily undergo the usual tests and quality checks of a paid product, may contain functional and performance defects and may be subject to errors. You are responsible for ensuring that the application examples are used in such a way that any malfunctions do not lead to property damage or personal injury.

Disclaimer of liability

Siemens excludes any liability, irrespective of the legal ground, in particular for the usability, availability, completeness and freedom from defects of the application examples, as well as associated notes, configuration data and performance data and any damage caused by these. 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 death, injury or health impairment, non-compliance with a guarantee, fraudulent non-disclosure of a defect or culpable breach of material contractual obligations. The claims for compensation for the breach of essential contractual obligations are, however, limited to the foreseeable damage typical for the type of contract, except in the event of intent or gross negligence or death or injury or health impairment. The above provisions do not entail a change in the burden of proof to your disadvantage. You release Siemens from any existing or pending claims of third parties in this context, unless Siemens has mandatory liability.

By using the application examples, you acknowledge that Siemens cannot be made liable for any claims beyond the liability clause described.

Additional notes

Siemens reserves the right to make changes to these application examples at any time without prior notice. If there are any discrepancies between the recommendations provided in these application examples and other Siemens publications – e.g. catalogs – the contents of the other documents have priority.

In addition to this, the Siemens terms of use apply (https://support.industry.siemens.com).

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 represent only one component of such a concept.

Customers are solely responsible for preventing unauthorized access to their plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary, and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens recommendations relating to appropriate security measures should be taken into account. For additional information on industrial security, please visit: https://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens expressly recommends that updates are carried out as soon as they become available – and that only the current product version is ever used. The use of product versions that are obsolete or no longer supported may increase the risk of cyber threats.

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

Table of contents

1 Introduction

This document describes how the TIA Portal Openness API can be used in combination with Startdrive.

First the fundamentals of Startdrive Openness will be explained.

Based on the fundamentals, various preconfigured and tested methods that facilitate various functions on drive units are described:

- Read out parameters
- Change parameters
- Create BICO links
- Create and delete telegrams
- Search for specific drive axes
- Browse a TIA project for drive units

Emphasis was placed on the fact that these methods do not create exceptions in the code and provide a return value that provides basic information about success/errors.

All methods are made available in a class library, which can be integrated in your own Visual Studio projects.

A typical integration in one of the templates provided is described in detail and can be retraced step-by-step.

2 Fundamentals of Startdrive Openness

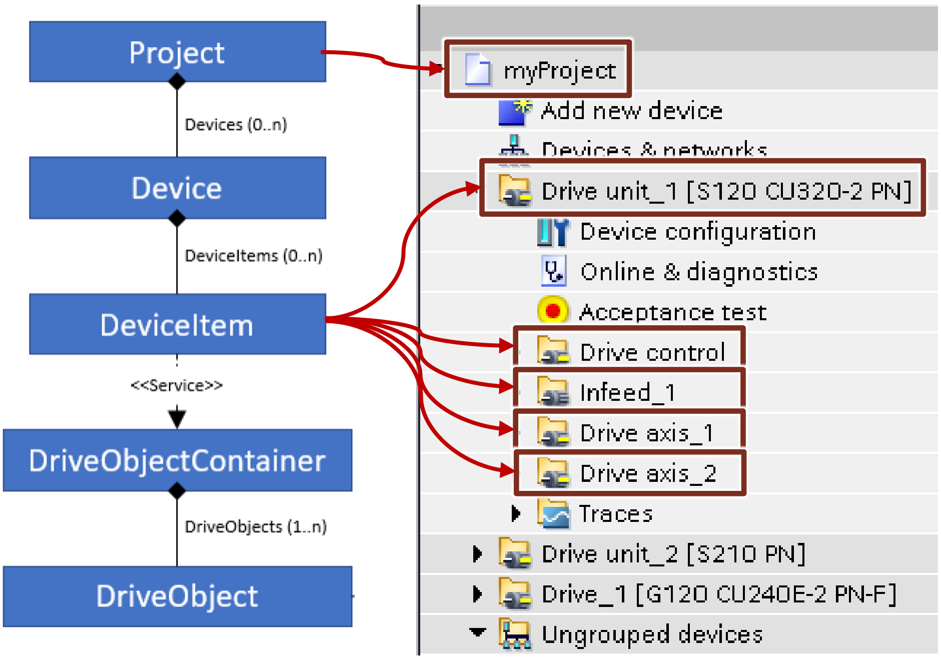
Different Openness operations on drive units are subsequently described in the following. In Chapter Class library StartdriveHelper these operations along with additional diagnostics are provided as encapsulated methods that can be further used.

2.1 Object model

Offline

The object model of Startdrive Openness is partially described, under the assumption that it is used to understand the described methods.

In the "StartdriveHelper" library, methods, which interact with objects of type "DriveObject", are mainly made available.



- The "DriveObjectContainer" service cannot be called on any object type "DeviceItem".
- For SINAMICS drives, only one object type "DriveObject" is in the "DriveObjectContainer"

Under the following conditions, the service "DriveObjectContainer" is available for SINAMICS drives and "DriveObject" can be accessed:

- S120/S210:

```
if (deviceItem.TypeIdentifier.ToString().Contains("System:Rack"))
{
   DriveObject selectedDriveObject =
GetService<DriveObjectContainer>().DriveObjects[0];
}
```

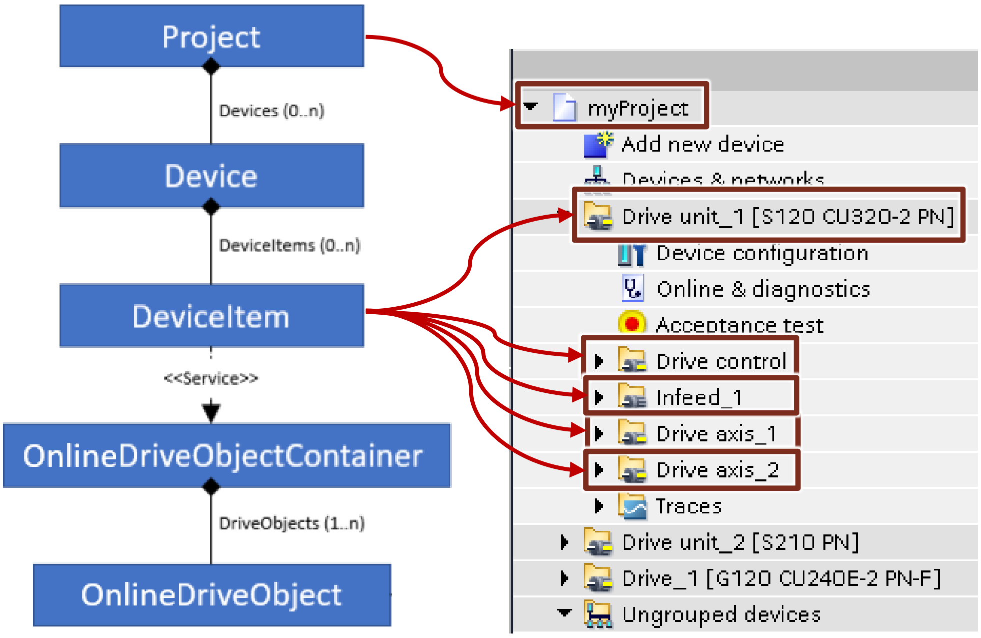
- G120:

```
if (deviceItem.Classification == DeviceItemClassifications.HM)
{
   DriveObject selectedDriveObject =
deviceItems.GetService<DriveObjectContainer>().DriveObjects[0];
}
```

As a multi-axis system, the SINAMICS S120 contains more than one "DriveObject". These objects can be differentiated either by parameter checks (e.g. p107) or by checking the name of the DeviceItem.

Online

The object model for online access to a drive unit differs slightly from the offline object model:



- On object type "DeviceItem", the service "OnlineDriveObjectContainer" is called for the online access.
- The "OnlineDriveObject" for parameter access can be found in "OnlineDriveObjectContainer"

Note

If a parameter is changed online, the offline value of a parameter does not change automatically. The value is only applied after an **upload**.

Similarly, the online value of a parameter does not change automatically when the offline value is changed. The value is only applied after a **download**.

The methods of library "StartdriveHelper" function both offline and online.

2.2 Fundamental information about interconnecting parameters

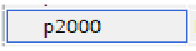
Parameters in SINAMICS have different structures. A parameter has to be addressed differently via TIA Portal Openness, depending on the specific structure. The parameters are accessed via TIA Portal Openness using a "DriveParameterComposition". In this section, this "DriveParameterComposition" is called "selectedDriveParameters".

Starting from an object "SelectedDriveObject" of type "DriveObject" or "OnlineDriveObject", DriveParameterComposition is accessed as follows:

```
DriveParameterComposition selectedDriveParameters =
selectedDriveObject. Parameters;
```

Parameters are interconnected as follows:

- Parameter without index, e.g. p2000

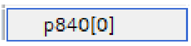


p2000

The parameter is written as follows, example: P2000 = 3500

```
selectedDriveParameters.Find(2000, -1).Value = "3500";
```

- Parameter with index, e.g. p840[0]

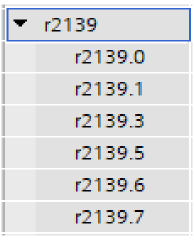


p840[0]

The parameter is written as follows, example: p840[0] = 0

```
selectedDriveParameters.Find(840, 0).Value = "0";
```

- Parameter without index, but with bit array, e.g. r2139



r2139
r2139.0
r2139.1
r2139.3
r2139.5
r2139.6
r2139.7

The parameter is found as follows, e.g.: Variable ack = r2139.3

```
var ack = selectedDriveParameters.Find(2139, -1).Bits[2];
```

or

```
var ack = selectedDriveParameters.Find(2139, -
1).Bits.Find("r2139.3");
```

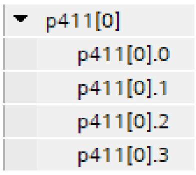
Note

Note regarding parameters with bit array

NOTICE: The bit array always starts at 0, and is incremented by +1 for each entry! Many bit parameters do not have contiguously ascending bit numbers! r2139 is an example of this, with r2139.0, r2139.1, r2139.3, …

As can be identified, r2139.2 does not exist! As a consequence, for the example listed above, r2139.3 is addressed using `selectedDriveParameters.Find(2139, -1).Bits[2];`.

- Parameter with index and with bit array, e.g. p411[0].0/ p411[0].1/…



p411[0]
p411[0].0
p411[0].1
p411[0].2
p411[0].3

The parameter is found as follows, example: p411[0].0 = 1

```
selectedDriveParameters.Find(411, 0).Bits[0].Value = "1";
```

Changing adjustable parameters

Values can be assigned to adjustable parameters. Adjustable parameter p2000 is an example, which represents the reference speed in revolutions per minute. In this case the Openness code is:

```
selectedDriveParameters.Find(2000, -1).Value = "3500";
```

Additional examples of other parameter structures are:

```
selectedDriveParameters.Find(2900, 0).Value = "5.0";
selectedDriveParameters.Find(212, -1).Bits[0].Value = "1";
selectedDriveParameters.Find(410, 0).Bits[0].Value = "1";
```

Changing a BICO interconnection (source and sink on the same drive axis)

BICO parameters always comprise a source and a sink, i.e. it is possible to interconnect one parameter with another parameter. p840[0] ON/OFF (OFF1) is an example of this. The following example shows how you can set p840[0] = r2090.0.

```
selectedDriveParameters.Find(840, 0).Value =
selectedDriveParameters.Find(2090, -1).Bits[0];
```

As both parameters are on the same drive axis, source and sink can be addressed via the same DriveParameterComposition.

Changing a BICO interconnection (source and sink on different drive axes)

BICO parameters always comprise a source and a sink, i.e. it is possible to interconnect one parameter with another parameter. p840[0] ON/OFF (OFF1) is an example of this. The following examples show you how you can set p840[0] = closed-loop drive control: r722.0.

To start, DriveParameterComposition myControlUnitParameter of the closed-loop drive control (Control Unit) must be found to interconnect the parameter.

```
selectedDriveParameters.Find(840, 0).Value =
myControlUnitParameter.Find(722, -1).Bits[0];
```

As the parameters are on different drive objects, they must be addressed with the associated DriveParameterComposition.

3 Class library StartdriveHelper

3.1 Methods for drive parameters

3.1.1 GetParameter

Function

Searches in a drive object for the specified parameter, represented by parameter number (and index, if available) or a representation as string.

If the parameter has been found in the drive object, the method returns the "DriveParameter" object of the parameter for further use.

Code

Method GetParameter is overloaded multiple times, some examples (not complete):

```
DriveParameter GetParameter(DriveObject actDriveObject, int
parameter, int index)

DriveParameter GetParameter(DriveObject actDriveObject, int
parameter)

DriveParameter GetParameter(DriveObject actDriveObject, string
parameter)

DriveParameter GetParameter(OnlineDriveObject actDriveObject, string
parameter)
```

Parameter

- actDriveObject: Drive object on which a search is made for the specified parameter
- (int) parameter: Number of the specified parameter
- Index: Index of the specified parameter
- (string) parameter: Parameter as string, e.g.: "2000", "840[0]", "2032.0". A leading "r" or "p" can be omitted

Return value

- DriveParameter: The parameter was found and is returned
- "zero": No parameter was found

Code example

```
DriveParameter myParameter = GetParameter(selectedDriveAxis, 2000);

DriveParameter myParameter = GetParameter(selectedDriveAxis, 840,
0);

DriveParameter myParameter = GetParameter(selectedDriveAxis,
"2000");

DriveParameter myParameter = GetParameter(selectedDriveAxis,
"p2000");

DriveParameter myParameter = GetParameter(selectedDriveAxis,
"840[0]");

DriveParameter myParameter = GetParameter(selectedDriveAxis,
"2032.0");
```

Private methods

```
DriveParameter GetParameterByString(DriveObject actDriveObject,
string parameter)
```

Method GetParameterByString is called by some of the overloads of method GetParameter.

Parameter "parameter" expects a string that starts with either a "p" or an "r".

3.1.2 ReadParameterValue

Function

Searches in a drive object for the specified parameter, represented by parameter number (and index, if available) or a representation as string.

If the parameter has been found in the drive object, the method returns the value of the parameter as String.

Code

Method ReadParameterValue is overloaded multiple times, some examples (not complete):

```
string ReadParameterValue(DriveObject actDriveObject, int parameter,
int index)
string ReadParameterValue(DriveObject actDriveObject, int parameter)
string ReadParameterValue(DriveObject actDriveObject, string
parameter)
string ReadParameterValue(OnlineDriveObject actDriveObject, string
parameter)
```

Parameter

- actDriveObject: Drive object on which a search is made for the specified parameter
- (int) parameter: Number of the specified parameter
- Index: Index of the specified parameter
- (string) parameter: Parameter as string, e.g.: "2000", "840[0]", "2032.0". A leading "r" or "p" can be omitted

Return value

- The parameter value is formatted as string
- "zero": No parameter was found

Code example

```
string myValue = ReadParameterValue(selectedDriveAxis, 2000);
string myValue = ReadParameterValue(selectedDriveAxis, 840, 0);
string myValue = ReadParameterValue(selectedDriveAxis, "2000");
string myValue = ReadParameterValue(selectedDriveAxis, "p2000");
string myValue = ReadParameterValue(selectedDriveAxis, "840[0]");
string myValue = ReadParameterValue(selectedDriveAxis, "2032.0");
```

3.1.3 SetParameter

Function

Searches in a drive object for the specified parameter and sets this to the required value, formatted as integer, double or string.

After writing, the parameter is read back and compared with the required value.

The method returns "true" if the parameter is set to the required value.

Code

Method SetParameter is overloaded multiple times, some examples (not complete):

```
bool SetParameter(DriveObject actDriveObject, int parameter, int
index, int value)
bool SetParameter(DriveObject actDriveObject, string parameter, int
value)
bool SetParameter(DriveObject actDriveObject, int parameter, int
index, double value)
bool SetParameter(DriveObject actDriveObject, int parameter, double
value)
bool SetParameter(DriveObject actDriveObject, int parameter, string
value)
bool SetParameter(DriveObject actDriveObject, string parameter,
string value)
bool SetParameter(OnlineDriveObject actDriveObject, string
parameter, string value)
```

Parameter

- actDriveObject: Drive object on which a search is made for the specified parameter
- (int) parameter: Number of the specified parameter
- Index: Index of the specified parameter
- (string) parameter: Parameter as string, e.g.: "2000", "840[0]", "2032.0". A leading "r" or "p" can be omitted
- (int) value: Value formatted as integer to which the parameter is to be set
- (double) value: Value formatted as double to which the parameter is to be set
- (string) value: Value formatted as a string to which the parameter is to be set

Return value

- "true": The parameter has been set to the required value
- "false": The parameter has not been set to the required value

Code example

```
SetParameter (selectedDriveAxis, 2000, 1500);
SetParameter (selectedDriveAxis, 2000, 123.45);
SetParameter (selectedDriveAxis, 2000, "500");
```

See Chapter GetParameter and ReadParameterValue for the various options of accessing parameters.

3.1.4 ConnectParameter

Function

Interconnects two BICO parameters (represented as string) with one another.

The signal sink is formed using parameters "actDriveObject" and "parameter".

Signal source is formed using parameters "actDriveObject" and "setToParameter" (source to the same DriveObject) or using parameters "connectedDriveObject" and "setToParameter" (source to another DriveObject).

The method returns "true" if the parameters have been successfully interconnected.

Code

The ConnectParameter method is overloaded.

```
bool ConnectParameter(DriveObject actDriveObject, string parameter,
string setToParameter)

bool ConnectParameter(DriveObject actDriveObject, string parameter,
DriveObject connectedDriveObject, string setToParameter)

bool ConnectParameter(OnlineDriveObject actDriveObject, string
parameter, OnlineDriveObject connectedDriveObject, string
setToParameter)
```

Parameter

- actDriveObject: Drive object of the signal sink (and source if the sink and source are located on the same drive object)
- parameters: Parameter as string, e.g.: "2000", "840[0]", "2032.0". A leading "r" or "p" can be omitted. This parameter is used as a signal sink of the BICO interconnection
- connectedDriveObject: Drive object of the signal source if the sink and the source are located on different drive objects
- setToParameter: Parameter as string, e.g.: "2000", "840[0]", "2032.0". A leading "r" or "p" can be omitted. This parameter is used as signal source of the BICO interconnection

Return value

- "true": The specified parameters have been interconnected
- "false": The specified parameters were not interconnected.

Code example

```
ConnectParameter(selectedDriveAxis, "840[0]", "2090.0");

ConnectParameter(selectedDriveAxis, "840[0]", selectedControlUnit,
"722.0");
```

3.1.5 GetParameterLimit

Function

Searches in a drive object for the given parameter, represented using a string.

If the parameter has been found in the drive object, the method returns the upper or lower parameter limit as a string.

Code

```
string GetParameterLimit(DriveObject actDriveObject, string
parameter, bool selectLimit)
string GetParameterLimit(OnlineDriveObject actDriveObject, string
parameter, bool selectLimit)
```

Parameter

- actDriveObject: Drive object on which a search is made for the specified parameter
- parameters: Parameter as string, e.g.: "2000", "840[0]", "2032.0". A leading "r" or "p" can be omitted
- selectLimit
  - "true": Upper parameter limit
  - "false": Lower parameter limit

Return value

- string: The parameter limit is returned
- "zero": No parameter limit found

Code example

```
string myLimit = GetParameterLimit(selectedDriveAxis, "2000",
false);
```

3.2 Methods for drive telegrams

Note

Drive telegrams can only be changed in the offline mode.

If an attempt is made to change drive telegrams in the online mode, the called methods return "false". "false" is also returned if the telegram is not supported by the drive object.

3.2.1 SetMainTelegramNumber

Function

Defines the main telegram on the specified axis (e.g. 1, 3, 5, 105, ...).

The method returns "true" if the telegram has been successfully defined.

Code

```
bool SetMainTelegramNumber(DriveObject actDriveObject, int
telegramNumber)
```

Parameter

- actDriveObject: Axis on which the main telegram is adapted
- telegramNumber: Number of the telegram

Return value

- "true": The telegram was successfully defined
- "false": The telegram was not set.

Code example

```
SetMainTelegramNumber(selectedDriveAxis, 105);
```

3.2.2 SetMainTelegramMCServo

Function

Specifies the process image of the main telegram on the specified axis on "MC Servo". OB "MC Servo" must be present on the PLC.

The method returns "true" if OB "MC Servo" was successfully interconnected.

Code

```
bool SetMainTelegramMCServo(DriveObject actDriveObject)
```

Parameter

- actDriveObject: Axis on which the main telegram is interconnected with "MC Servo"

Return value

- "true": "MC Servo" was defined successfully
- "false": "MC Servo" was not set.

Code example

```
SetMainTelegramMCServo(selectedDriveAxis);
```

3.2.3 GetMainTelegramAddressIn

Function

Returns the start address (receive direction) of the main telegram of the specified axis.

The method returns "-1" if the start address could not be read out.

Code

```
int GetMainTelegramAddressIn(DriveObject actDriveObject)
```

Parameter

- actDriveObject: Axis on which the start address of the main telegram is read

Return value

- Start address of the main telegram (receive direction)
- -1, if the start address could not be read out

Code example

```
int myAddressIn = GetMainTelegramAddressIn(selectedDriveAxis);
```

3.2.4 GetMainTelegramAddressOut

Function

Returns the start address (send direction) of the main telegram of the specified axis.

The method returns "-1" if the start address could not be read out.

Code

```
int GetMainTelegramAddressOut(DriveObject actDriveObject)
```

Parameter

- actDriveObject: Axis on which the start address of the main telegram is read

Return value

- Start address of the main telegram (send direction)
- -1, if the start address could not be read out

Code example

```
int myAddressOut = GetMainTelegramAddressOut(selectedDriveAxis);
```

3.2.5 AddAdditionalTelegram

Function

Specifies a free supplementary telegram with the specified send and receive length on the specified axis.

The method returns "true" if the free supplementary telegram has been successfully defined.

Code

```
bool AddAdditionalTelegram(DriveObject actDriveObject, int
sendLength, int receiveLength)
```

Parameter

- actDriveObject: Axis on which the free supplementary telegram is added
- sendLength: Send length in words
- receiveLength: Receive length in words

Return value

- "true": Free supplementary telegram has been successfully added
- "false": Free supplementary telegram was not added

Code example

```
AddAdditionalTelegram(selectedDriveAxis, 10, 5);
```

3.2.6 DeleteAdditionalTelegrams

Function

Deletes the free supplementary telegram on the specified axis.

Code

```
bool DeleteAdditionalTelegrams(DriveObject actDriveObject)
```

Parameter

- `actDriveObject:` Axis on which the free supplementary telegram is deleted

Return value

- "true": Free supplementary telegram has been successfully deleted
- "false": Free supplementary telegram was not deleted

Code example

```
DeleteAdditionalTelegrams(selectedDriveAxis);
```

3.2.7 AddTorqueTelegram

Function

Creates the torque supplementary telegram 750 on the specified axis.

The method returns "true" if the supplementary telegram 750 has been successfully defined.

Code

```
bool AddTorqueTelegram(DriveObject actDriveObject)
```

Parameter

- actDriveObject: Axis on which the supplementary telegram 750 is added

Return value

- "true": Supplementary telegram 750 has been successfully added
- "false": Supplementary telegram 750 was not added

Code example

```
AddTorqueTelegram(selectedDriveAxis);
```

3.2.8 DeleteTorqueTelegram

Function

Deletes the torque supplementary telegram 750 on the specified axis.

The method returns "true" if the supplementary telegram 750 has been successfully deleted.

Code

```
bool DeleteTorqueTelegram(DriveObject actDriveObject)
```

Parameter

- actDriveObject: Axis on which the supplementary telegram 750 is deleted

Return value

- "true": Supplementary telegram 750 has been successfully deleted
- "false": Supplementary telegram 750 was not deleted

Code example

```
DeleteTorqueTelegram(selectedDriveAxis);
```

3.2.9 AddSafetyTelegram

Function

On the specified axis, creates a PROFIsafe telegram with the specified telegram number (e.g. 30, 31, 901, ...).

The method returns "true" if the PROFIsafe telegram has been set successfully.

Code

```
bool AddSafetyTelegram(DriveObject actDriveObject, int
telegramNumber)
```

Parameter

- actDriveObject: Axis on which the PROFIsafe telegram is defined
- telegramNumber: Number of the PROFIsafe telegram

Return value

- "true": The PROFIsafe telegram has been defined successfully
- "false": The PROFIsafe telegram was not set.

Code example

```
AddSafetyTelegram(selectedDriveAxis, 30);
```

3.2.10 DeleteSafetyTelegram

Function

Deletes the PROFIsafe telegram on the specified axis.

The method returns "true" if the PROFIsafe telegram has been successfully deleted.

Code

```
bool DeleteSafetyTelegram(DriveObject actDriveObject)
```

Parameter

- actDriveObject: Axis on which the PROFIsafe telegram is deleted

Return value

- "true": The PROFIsafe telegram has been successfully deleted.
- "false": The PROFIsafe telegram was not deleted.

Code example

```
DeleteSafetyTelegram(selectedDriveAxis);
```

3.2.11 AddSafetyInfoControlChannel

Function

Creates on the specified axis the safety info/control channel with the specified telegram number (e.g. 700, 701).

The method returns "true" if the safety info/control channel has been successfully defined.

Code

```
bool AddSafetyInfoControlChannel(DriveObject actDriveObject, int
telegramNumber)
```

Parameter

- actDriveObject: Axis on which the safety info/control channel is defined
- telegramNumber: Number of the safety info/control channel

Return value

- "true": The safety info/control channel has been defined successfully
- "false": The safety info/control channel is not set.

Code example

```
AddSafetyInfoControlChannel(selectedDriveAxis, 701);
```

3.2.12 DeleteSafetyInfoControlChannel

Function

Deletes the safety info/control channel on the specified axis.

The method returns "true" if the safety info/control channel has been successfully deleted.

Code

```
bool DeleteSafetyInfoControlChannel(DriveObject actDriveObject)
```

Parameter

- actDriveObject: Axis on which the safety info/control channel is deleted

Return value

- "true": The safety info/control channel has been successfully deleted
- "false": The safety info/control channel is not deleted.

Code example

```
DeleteSafetyInfoControlChannel(selectedDriveAxis);
```

3.2.13 SetFreeTelegram

Function

Changes the main telegram to a free telegram with specified send and receive length on the axis.

The method returns "true" if the telegram has been successfully defined.

Code

```
bool SetFreeTelegram(DriveObject actDriveObject, int sendLength, int
receiveLength, bool keepAddr)
```

Parameter

- actDriveObject: Axis on which the main telegram is adapted
- sendLength: Send length in words
- receiveLength: Receive length in words
- keepAddr
  - "true": The start address of the main telegram is retained.
  - "false": The start address of the main telegram can change

Return value

- "true": The telegram has been successfully changed to a free telegram
- "false": The telegram was not changed

Code example

```
SetFreeTelegram(selectedDriveAxis, 10, 10, true);
```

3.2.14 AddMainTelegramExtension

Function

Extends the main telegram by the specified send and receive length.

The method returns "true" if the telegram has been successfully extended.

Code

```
bool AddMainTelegramExtension(DriveObject actDriveObject, int
sendExtension, int receiveExtension, bool keepAddr)
```

Parameter

- actDriveObject: Axis on which the main telegram is expanded
- sendExtension: Extension (send direction) in words

- receiveExtension: Extension (receive direction) in words
- keepAddr
  - "true": The start address of the telegram is retained.
  - "false": The start address of the telegram can change

Return value

- "true": The main telegram has been successfully extended
- "false": The main telegram was not extended

Code example

```
AddMainTelegramExtension(selectedDriveAxis, 2, 2, false);
```

3.3 Methods for drive axes

3.3.1 GetControlUnit

Function

Searches for the Control Unit in a drive unit with CU 320-2 (e.g. S120).

If the Control Unit was found in the drive unit, the method returns the "DriveObject" object of the Control Unit for further use.

Code

```
DriveObject GetControlUnit(DriveObject actDriveObject)
OnlineDriveObject GetControlUnit(OnlineDriveObject actDriveObject)
```

Parameter

- actDriveObject: Axis of the drive unit, starting from which the Control Unit is searched for

Return value

- DriveObject: The Control Unit was found and is returned
- "zero": A Control Unit was not found

Code example

```
DriveObject myControlUnit = GetControlUnit(selectedDriveAxis);
```

3.3.2 GetInfeedAxis

Function

Searches for an infeed in a drive unit with CU 320-2 (e.g. S120).

If an infeed has been found in the drive unit, the method returns the "DriveObject" object of the infeed for further use.

Code

```
DriveObject GetInfeedAxis (DriveObject actDriveObject)
OnlineDriveObject GetInfeedAxis(OnlineDriveObject actDriveObject)
```

Parameter

- actDriveObject: Axis of the drive unit, starting from which the infeed is searched for

Return value

- DriveObject: The infeed was found and is returned
- zero: No infeed was found

Code example

```
DriveObject myInfeed = GetInfeedAxis(selectedDriveAxis);
```

3.3.3 GetDriveAxisByName

Function

Searches for an axis with the given name in an S120 drive unit.

If an axis with the specified name was found in the drive unit, the method returns the "DriveObject" object of this axis for further use.

Code

```
DriveObject GetDriveAxisByName(DriveObject actDriveObject, String
nameOfS120Axis)
OnlineDriveObject GetDriveAxisByName(OnlineDriveObject
actDriveObject, String nameOfS120Axis)
```

Parameter

- actDriveObject: Axis of the drive unit, starting from which the axis is searched for with the specified name
- nameOfS120Axis: Name of the axis for which a search is performed in the drive unit

Return value

- DriveObject: The axis with the name that was being searched for was found and is returned
- "zero": No axis was found with the specified name

Code example

```
DriveObject myDriveAxis = (selectedDriveAxis, "myDriveAxisName");
```

3.4 Methods at the project level

3.4.1 OpenProject

Function

Opens a TIA project when the TIA Portal is open.

The method returns "true" if the project was successfully opened.

Code

```
bool OpenProject(string filePath, TiaPortal tiaPortal)
```

Parameter

- filePath: Absolute path to the TIA project that is to be opened
- tiaPortal: Opened instance of the TIA portal

Return value

- "true": The TIA project was successfully opened
- "false": The TIA project was not opened

Code example

```
OpenProject(@"D:\myTiaProject.ap16", myTiaPortal);
```

3.4.2 EnumerateDevicesInProject

Function

Searches a TIA project for devices. The search criteria are defined in the private method EnumerateDevices. As default, all SINAMICS drives are found. Groups and nested groups are also searched.

The method returns a list with all the found "Device" objects.

Code

```
IList<Device> EnumerateDevicesInProject(Project project)
```

Parameter

- project: TIA project that is searched for devices, which are added to the returned list

Return value

- IList<Device>: List of objects of the type Device that were found in the project

Code example

```
List<Device> myDeviceList = EnumerateDevicesInProject(myProject) as
List<Device>;
```

Private methods

```
void EnumerateDevicesInGroups(IList<Device> deviceList, Project
project)
```

The EnumerateDevicesInGroups method searches through all groups for devices.

```
void EnumerateDeviceUserGroup(IList<Device> deviceList,
DeviceUserGroup deviceUserGroup)
```

The EnumerateDeviceUserGroup method recursively searches through all nested groups for devices.

```
void EnumerateDevices(IList<Device> deviceList, DeviceComposition
deviceComposition)
```

The EnumerateDevices method inserts all devices that correspond to the criteria in the method to "deviceList" list. As default, all SINAMICS drives are considered.

3.4.3 EnumerateDriveObjectsInDeviceList

Function

Searches a list of devices for drive objects.

The method returns a list with all the "DriveObject" objects found.

Code

```
IList<DriveObject> EnumerateDriveObjectsInDeviceList(IList<Device>
deviceList)
```

Parameter

- deviceList: List of devices searched for objects of type DriveObject.

Return value

- IList<DriveObject>: List of objects of type DriveObject that were found in the DeviceList

Code example

```
List<DriveObject> myDriveObjectList =
EnumerateDriveObjectsInDeviceList(myDevices) as List<DriveObject>;
```

Private methods

```
void EnumerateDriveObjectsInDevice(IList<DriveObject>
driveObjectList, Device device)
```

The EnumerateDriveObjectsInDevice method searches the transferred device and adds all DriveObjects that match the criteria in the method to the "DriveObjectList" list. As default, all SINAMICS drive objects are considered.

3.4.4 EnumerateOnlineDriveObjectsInDeviceList

Function

Searches a list of devices for drive objects.

The method returns a list with all the objects "OnlineDriveObjekt" that were found.

Code

```
IList<OnlineDriveObject>
EnumerateOnlineDriveObjectsInDeviceList(IList<Device> deviceList)
```

Parameter

- deviceList: List of devices searched for objects of type OnlineDriveObject.

Return value

- IList<OnlineDriveObject>: List of objects of type OnlineDriveObject that were found in the DeviceList

Code example

```
List<OnlineDriveObject> myDriveObjectList =
EnumerateOnlineDriveObjectsInDeviceList(myDevices) as
List<OnlineDriveObject>;
```

Private methods

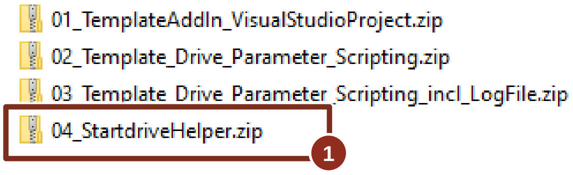
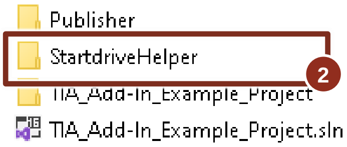
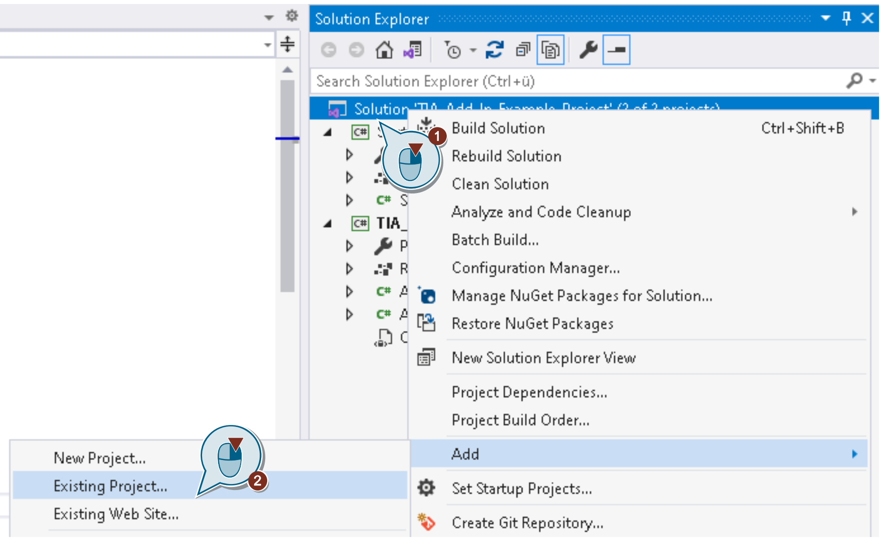
```
void EnumerateOnlineDriveObjectsInDevice(IList<OnlineDriveObject>
driveObjectList, Device device)
```

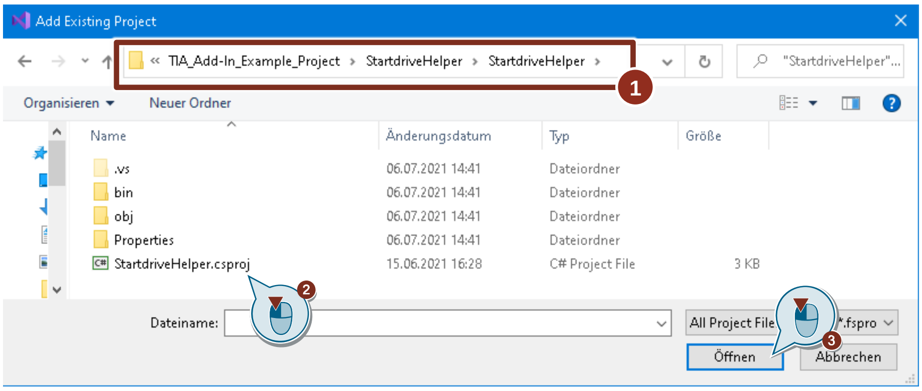
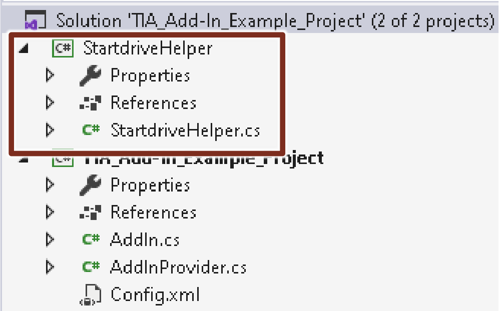
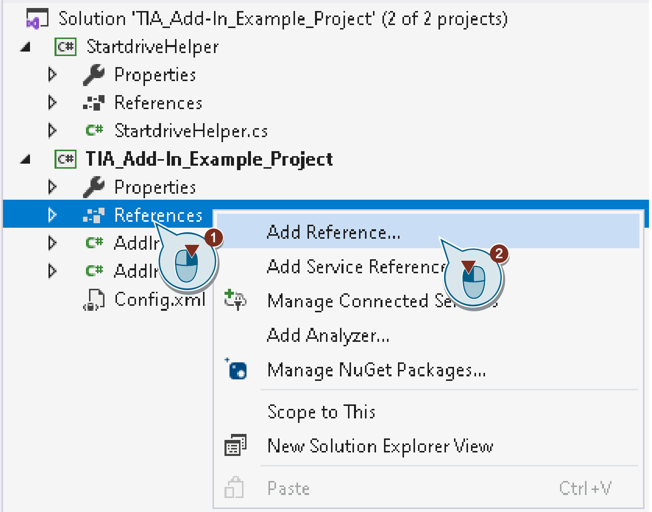
The EnumerateOnlineDriveObjectsInDevice method searches the transferred device and adds all OnlineDriveObjects that match the criteria in the method to the "DriveObjectList" list. As default, all SINAMICS drive objects are considered.

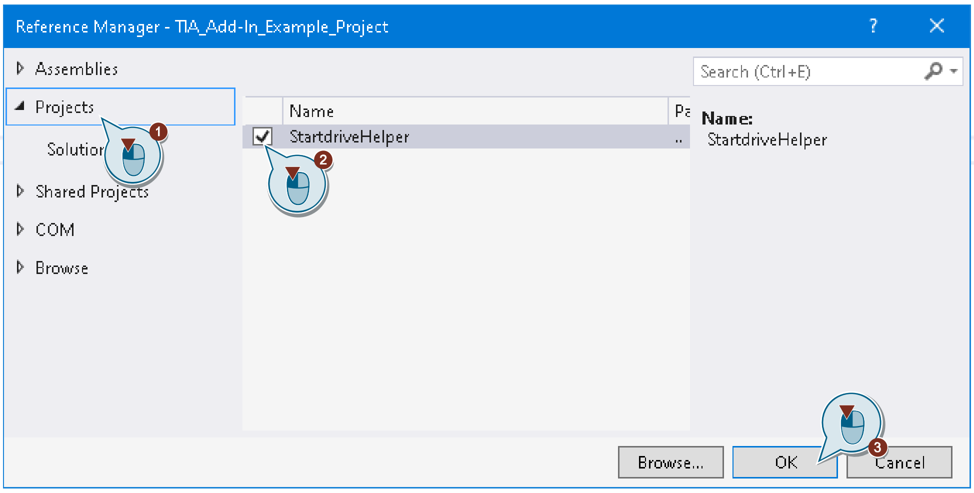
3.5 Integrating the library ...

3.5.1 ... via the Visual Studio project of the library

Table 3-1: Integrating the library via the library project

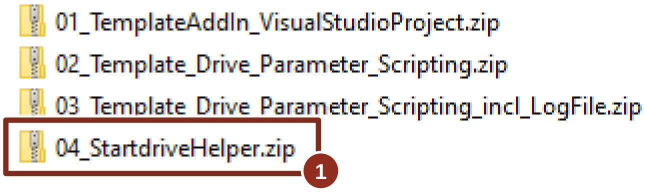
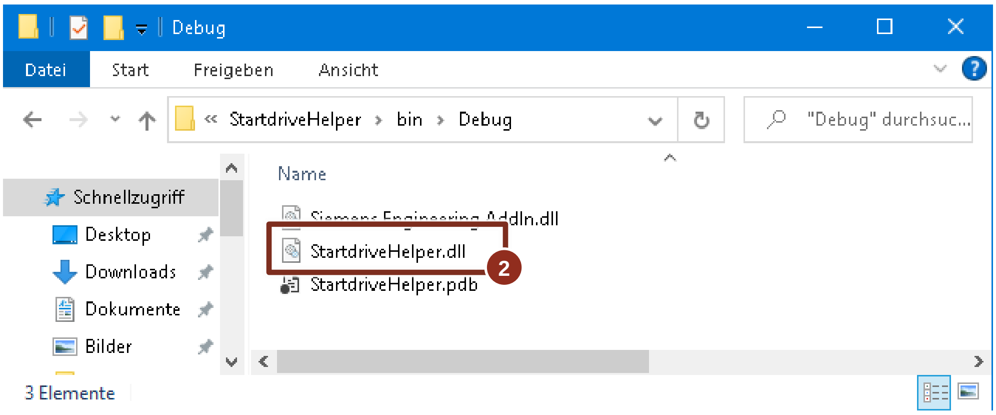
No.	Action
1.	Unzip the Visual Studio project "StartdriveHelper" that was provided **(1)**.  01_TemplateAddIn_VisualStudioProject.zip 02_Template_Drive_Parameter_Scripting.zip 03_Template_Drive_Parameter_Scripting_incl_LogFile.zip 04_StartdriveHelper.zip 1 Insert the unzipped project in the storage location of your solution **(2)**.  Publisher StartdriveHelper TIA_Add-In_Example_Project TIA_Add-In_Example_Project.sln 2
2.	Add the project to your solution by right-clicking on your solution **(1)** and selecting "Add > Existing Project..." **(2)**.  Solution Explorer Search Solution Explorer (Ctrl+ü) Solution 'TIA_Add-In_Example_Project' (2 of 2 projects) TIA_ Build Solution Ctrl+Shift+B Rebuild Solution Clean Solution Analyze and Code Cleanup Batch Build... Configuration Manager... Manage NuGet Packages for Solution... Restore NuGet Packages New Solution Explorer View Project Dependencies... Project Build Order... Add Set Startup Projects... Create Git Repository... New Project... Existing Project... Existing Web Site... 1 2

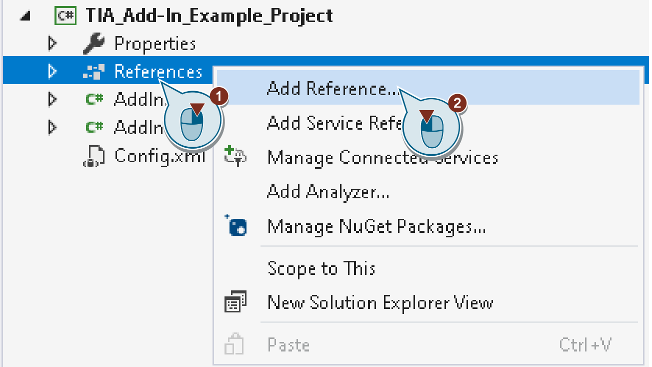
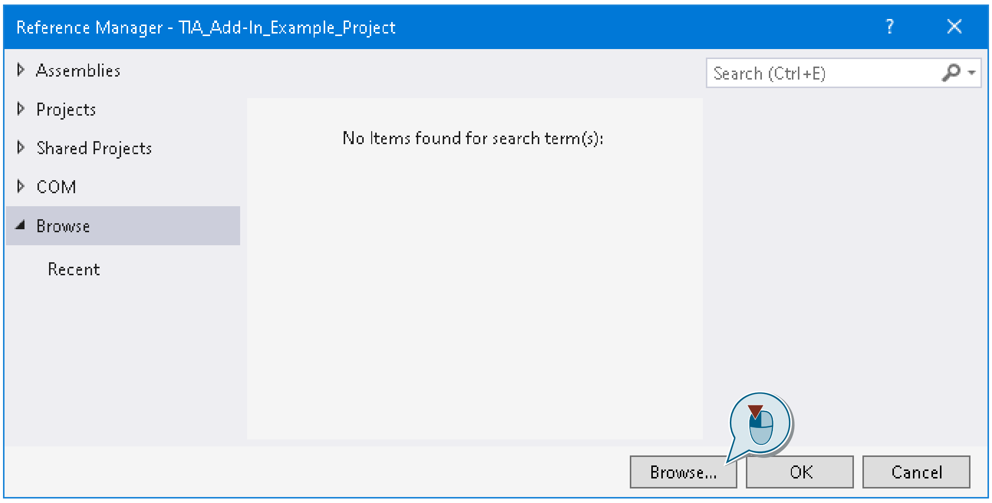
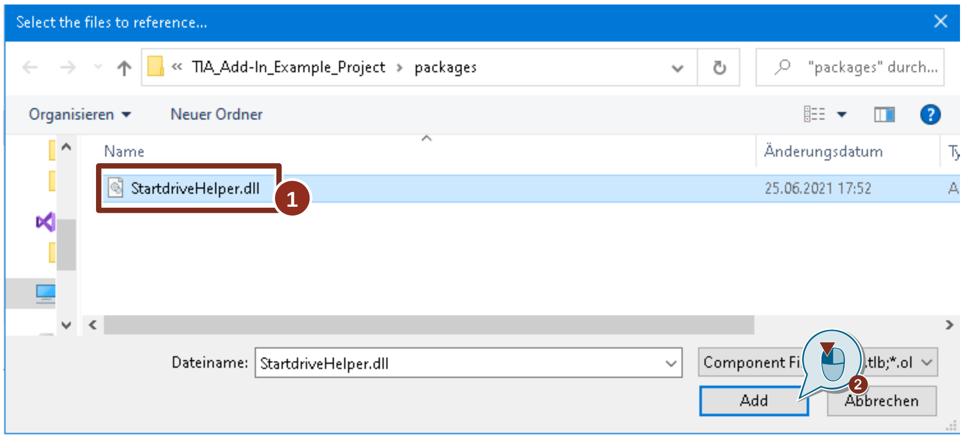
No.	Action
3.	Navigate to the "StartdriveHelper" project **(1)**, select the "StartdriveHelper.csproj" project file **(2)** and confirm by clicking on "Open" **(3)**.  Add Existing Project « TIA_Add-In_Example_Project › StartdriveHelper › StartdriveHelper › Name — Änderungsdatum — Typ — Größe .vs — 06.07.2021 14:41 — Dateiordner bin — 06.07.2021 14:41 — Dateiordner obj — 06.07.2021 14:41 — Dateiordner Properties — 06.07.2021 14:41 — Dateiordner StartdriveHelper.csproj — 15.06.2021 16:28 — C# Project File — 3 KB Dateiname: All Project File *.fspro "StartdriveHelper"... Organisieren Neuer Ordner 1 2 3 Öffnen Abbrechen
4.	The Visual Studio project "StartdriveHelper" is now part of the solution.  Solution 'TIA_Add-In_Example_Project' (2 of 2 projects) StartdriveHelper Properties References StartdriveHelper.cs TIA_Add-In_Example_Project Properties References AddIn.cs AddInProvider.cs Config.xml
5.	In order to be able to access the methods, project "StartdriveHelper" must be referenced in your Visual Studio project. In your Visual Studio project, right-click on "References" **(1)** and on the left-hand side on "Add Reference" **(2)**  Solution 'TIA_Add-In_Example_Project' (2 of 2 projects) StartdriveHelper Properties References StartdriveHelper.cs TIA_Add-In_Example_Project Properties References AddIr AddIr Config.xml Add Reference... Add Service Referenc Manage Connected Se Add Analyzer... Manage NuGet Packages... Scope to This New Solution Explorer View Paste Ctrl+V 1 2

No.	Action
6.	Under "Projects" **(1)** select project "StartdriveHelper" **(2)** and confirm with "OK" **(3)**. The "StartdriveHelper" project is now referenced in your Visual Studio project.  Reference Manager - TIA_Add-In_Example_Project ? Assemblies Projects 1 Solution Shared Projects COM Browse Search (Ctrl+E) Name Pa ☑ StartdriveHelper 2 Name: StartdriveHelper Browse... OK 3 Cancel

3.5.2 ... via the library DLL

Table 3-2: Integrating the library via the library DLL

No.	Action
1.	Unzip the Visual Studio project "StartdriveHelper" that was provided **(1)**.  01_TemplateAddIn_VisualStudioProject.zip 02_Template_Drive_Parameter_Scripting.zip 03_Template_Drive_Parameter_Scripting_incl_LogFile.zip 04_StartdriveHelper.zip The .dll is located in folder "StartdriveHelper > bin > Debug" **(2)**.  Debug Datei Start Freigeben Ansicht « StartdriveHelper › bin › Debug "Debug" durchsuc... Schnellzugriff Desktop Downloads Dokumente Bilder Name Siemens.Engineering.AddIn.dll StartdriveHelper.dll StartdriveHelper.pdb 2 3 Elemente

No.	Action
2.	In order to be able to access the methods, the .dll of the "StartdriveHelper" project must be referenced in your Visual Studio project. In your Visual Studio project, right-click on "References" **(1)** and on the left-hand side on "Add Reference" **(2)**  TIA_Add-In_Example_Project Properties References AddIn AddIn Config.xml 1 Add Reference... 2 Add Service Refe Manage Connected Services Add Analyzer... Manage NuGet Packages... Scope to This New Solution Explorer View Paste Ctrl+V
3.	Browse the file system by clicking on "Browse..."  Reference Manager - TIA_Add-In_Example_Project ? Assemblies Projects Shared Projects COM Browse Recent Search (Ctrl+E) No Items found for search term(s): Browse... OK Cancel
4.	Search for the "StartdriveHelper.dll" data in the file system **(1)** and add it to your Visual Studio project as reference by clicking on "Add" **(2)**. The .dll can also be copied to a different storage location, e.g. in the current Visual Studio project.  Select the files to reference... « TIA_Add-In_Example_Project › packages "packages" durch... Organisieren Neuer Ordner Name Änderungsdatum Ty StartdriveHelper.dll 25.06.2021 17:52 A 1 Dateiname: StartdriveHelper.dll Component Fi tlb;*.ol Add Abbrechen 2
5.	The .dll of the "StartdriveHelper" project is now referenced in your Visual Studio project.

3.5.3 Registering a library in the Visual Studio project

Table 3-3: Registering a library in the Visual Studio project

No.	Action
1.	The library is integrated in the Visual Studio project using the following code line. `using static SDRhelper.StartdriveHelper;`

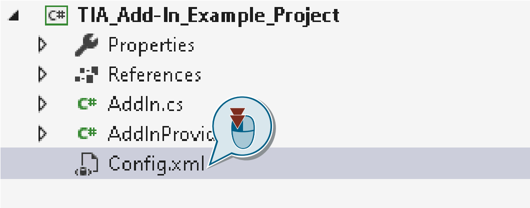
3.5.4 Using the library in an add-in

Table 3-4: Using the library in an add-in

<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>1.</td><td>

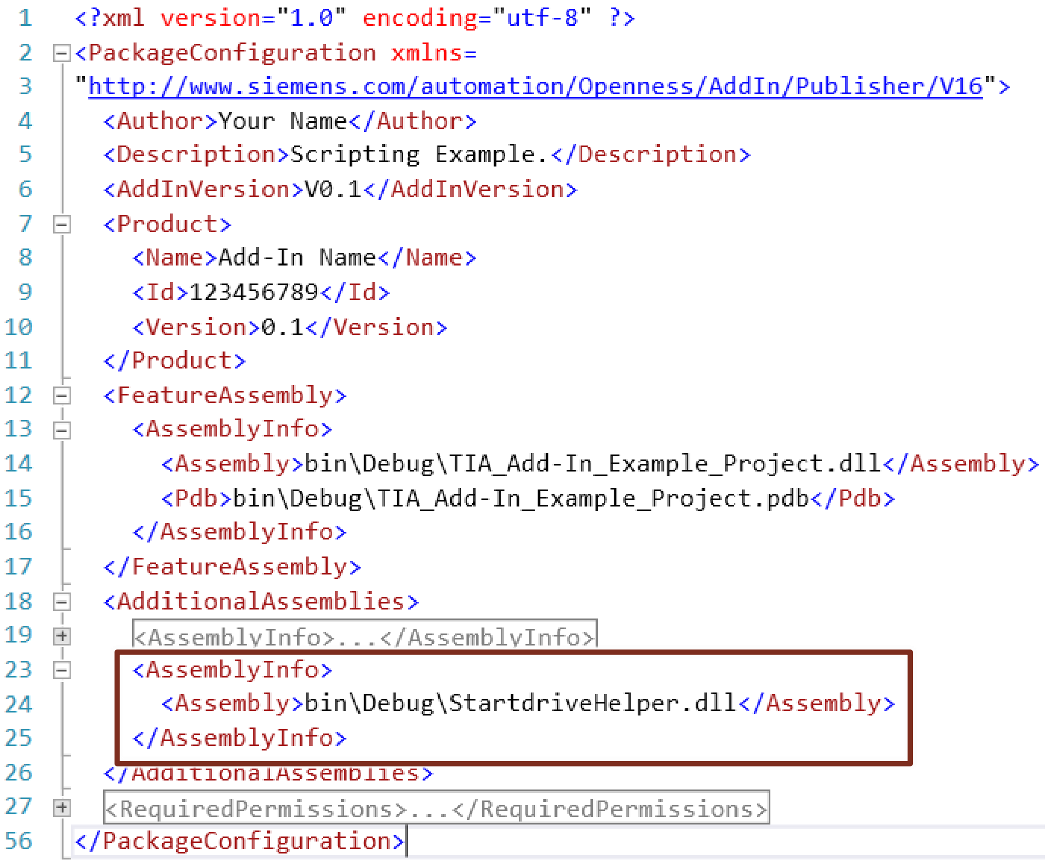
You can read how to create an .addin file in the second document of this SIOS entry.

Open the Config.xml of your Visual Studio project with a double-click.



</td></tr>
<tr><td>2.</td><td>

Under "AdditionalAssemblies", insert the selected part so that the "StartdriveHelper.dll" is considered when the .addin file is created.

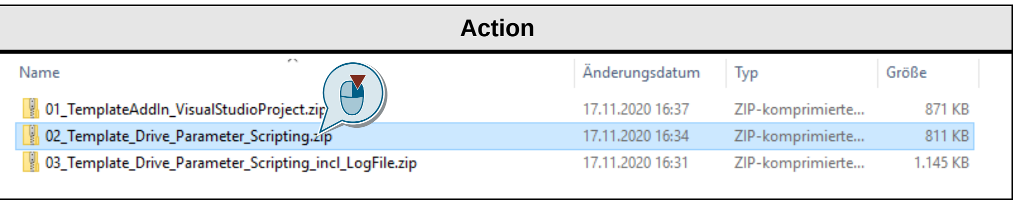
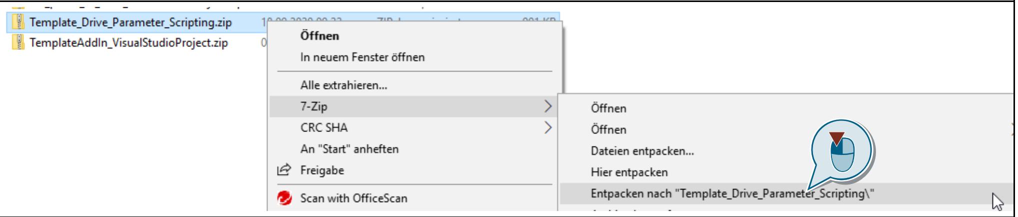
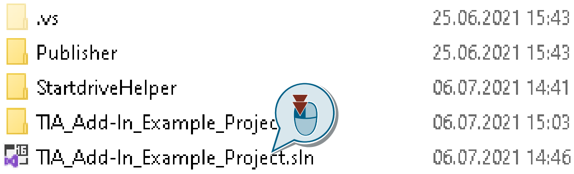
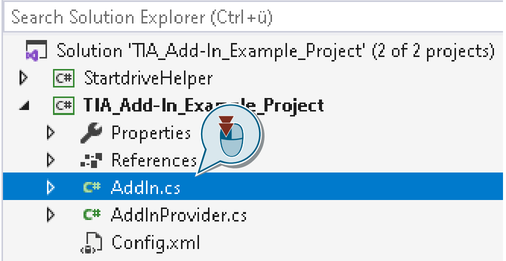
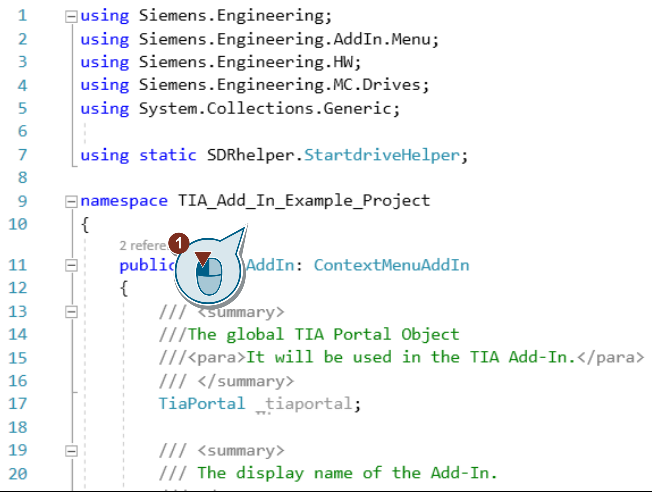
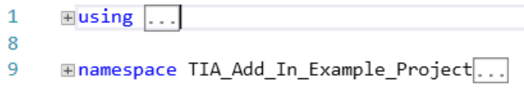


```
<?xml version="1.0" encoding="utf-8" ?>
<PackageConfiguration xmlns=
"http://www.siemens.com/automation/Openness/AddIn/Publisher/V16">
  <Author>Your Name</Author>
  <Description>Scripting Example.</Description>
  <AddInVersion>V0.1</AddInVersion>
  <Product>
    <Name>Add-In Name</Name>
    <Id>123456789</Id>
    <Version>0.1</Version>
  </Product>
  <FeatureAssembly>
    <AssemblyInfo>
      <Assembly>bin\Debug\TIA_Add-In_Example_Project.dll</Assembly>
      <Pdb>bin\Debug\TIA_Add-In_Example_Project.pdb</Pdb>
    </AssemblyInfo>
  </FeatureAssembly>
  <AdditionalAssemblies>
    <AssemblyInfo>...</AssemblyInfo>
    <AssemblyInfo>
      <Assembly>bin\Debug\StartdriveHelper.dll</Assembly>
    </AssemblyInfo>
  </AdditionalAssemblies>
  <RequiredPermissions>...</RequiredPermissions>
</PackageConfiguration>
```

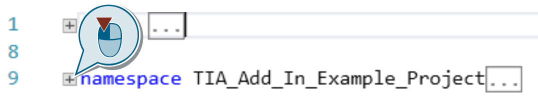
</td></tr>
</table>

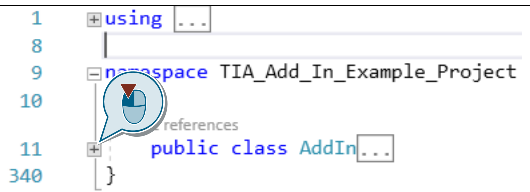
3.6 Integration in the existing template

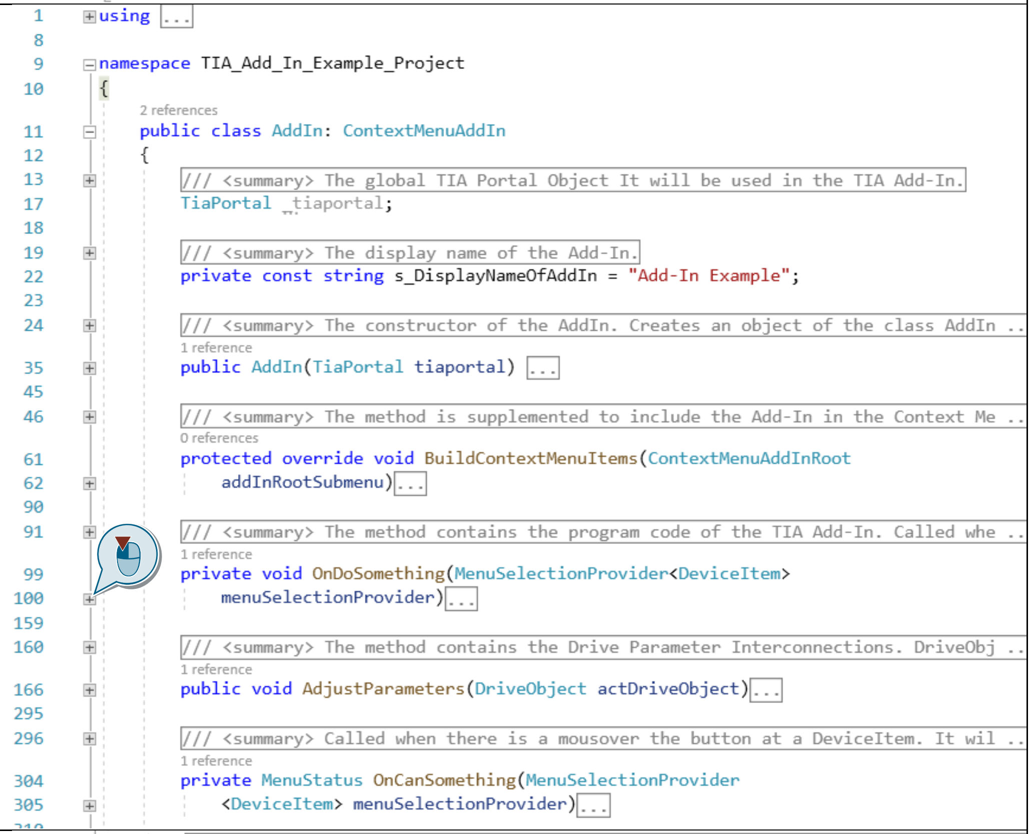
The starting point for this chapter is Visual Studio project "02_Template_Drive_Parameter_Scripting". To do this, the Visual Studio project provided must be unzipped.

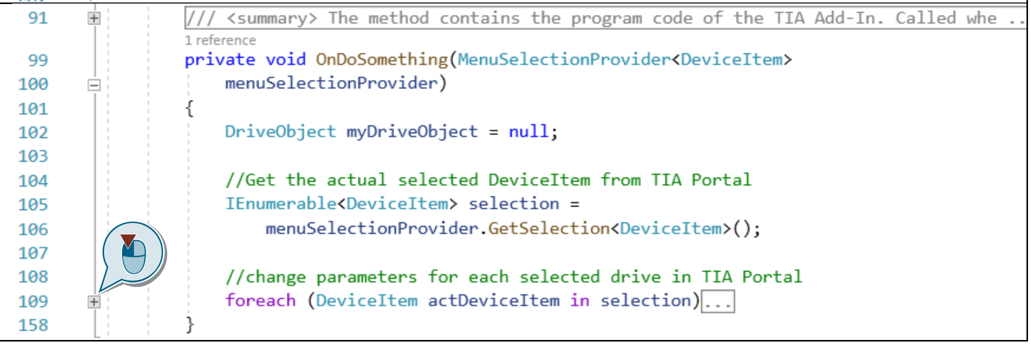
No.	Action
1.	 Name | Änderungsdatum | Typ | Größe 01_TemplateAddIn_VisualStudioProject.zip | 17.11.2020 16:37 | ZIP-komprimierte... | 871 KB 02_Template_Drive_Parameter_Scripting.zip | 17.11.2020 16:34 | ZIP-komprimierte... | 811 KB 03_Template_Drive_Parameter_Scripting_incl_LogFile.zip | 17.11.2020 16:31 | ZIP-komprimierte... | 1.145 KB
2.	 Template_Drive_Parameter_Scripting.zip TemplateAddIn_VisualStudioProject.zip Öffnen In neuem Fenster öffnen Alle extrahieren... 7-Zip CRC SHA An "Start" anheften Freigabe Scan with OfficeScan Öffnen Öffnen Dateien entpacken... Hier entpacken Entpacken nach "Template_Drive_Parameter_Scripting\"
3.	 .vs | 25.06.2021 15:43 Publisher | 25.06.2021 15:43 StartdriveHelper | 06.07.2021 14:41 TIA_Add-In_Example_Projec | 06.07.2021 15:03 TIA_Add-In_Example_Project.sln | 06.07.2021 14:46
4.	 Search Solution Explorer (Ctrl+ü) Solution 'TIA_Add-In_Example_Project' (2 of 2 projects) StartdriveHelper TIA_Add-In_Exa...le_Project Properties References AddIn.cs AddInProvider.cs Config.xml
5.	Left click in the program code.  using Siemens.Engineering; using Siemens.Engineering.AddIn.Menu; using Siemens.Engineering.HW; using Siemens.Engineering.MC.Drives; using System.Collections.Generic; using static SDRhelper.StartdriveHelper; namespace TIA_Add_In_Example_Project { 2 refere public AddIn: ContextMenuAddIn { /// <summary> ///The global TIA Portal Object ///<para>It will be used in the TIA Add-In.</para> /// </summary> TiaPortal _tiaportal; /// <summary> /// The display name of the Add-In.
6.	CTRL + A followed by CTRL + M + M must be pressed, and then the complete code collapses. During the course of the description, the relevant code positions are discussed in more detail.  using ... namespace TIA_Add_In_Example_Project...

No.	Action

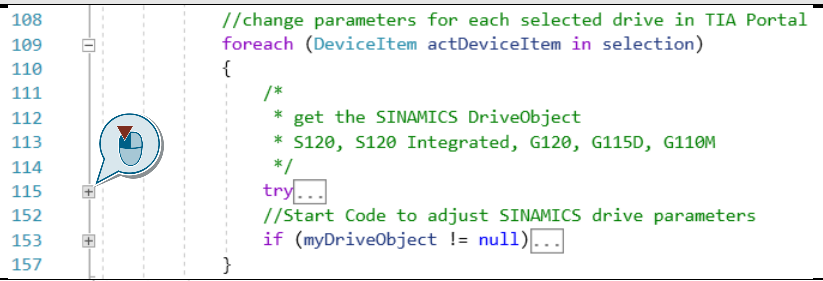
7.	Then, only the relevant areas are expanded again:
	
	`[...]`
	`namespace TIA_Add_In_Example_Project[...]`

8.	
	`using [...]`
	`namespace TIA_Add_In_Example_Project`
	`references`
	`    public class AddIn[...]`
	`}`

9.	
	`using [...]`
	`namespace TIA_Add_In_Example_Project`
	`{`
	`    2 references`
	`    public class AddIn: ContextMenuAddIn`
	`    {`
	`        /// <summary> The global TIA Portal Object It will be used in the TIA Add-In.`
	`        TiaPortal _tiaportal;`
	`        /// <summary> The display name of the Add-In.`
	`        private const string s_DisplayNameOfAddIn = "Add-In Example";`
	`        /// <summary> The constructor of the AddIn. Creates an object of the class AddIn ...`
	`        1 reference`
	`        public AddIn(TiaPortal tiaportal) [...]`
	`        /// <summary> The method is supplemented to include the Add-In in the Context Me ...`
	`        0 references`
	`        protected override void BuildContextMenuItems(ContextMenuAddInRoot`
	`            addInRootSubmenu)[...]`
	`        /// <summary> The method contains the program code of the TIA Add-In. Called whe ...`
	`        1 reference`
	`        private void OnDoSomething(MenuSelectionProvider<DeviceItem>`
	`            menuSelectionProvider)[...]`
	`        /// <summary> The method contains the Drive Parameter Interconnections. DriveObj ...`
	`        1 reference`
	`        public void AdjustParameters(DriveObject actDriveObject)[...]`
	`        /// <summary> Called when there is a mousover the button at a DeviceItem. It wil ...`
	`        1 reference`
	`        private MenuStatus OnCanSomething(MenuSelectionProvider`
	`            <DeviceItem> menuSelectionProvider)[...]`

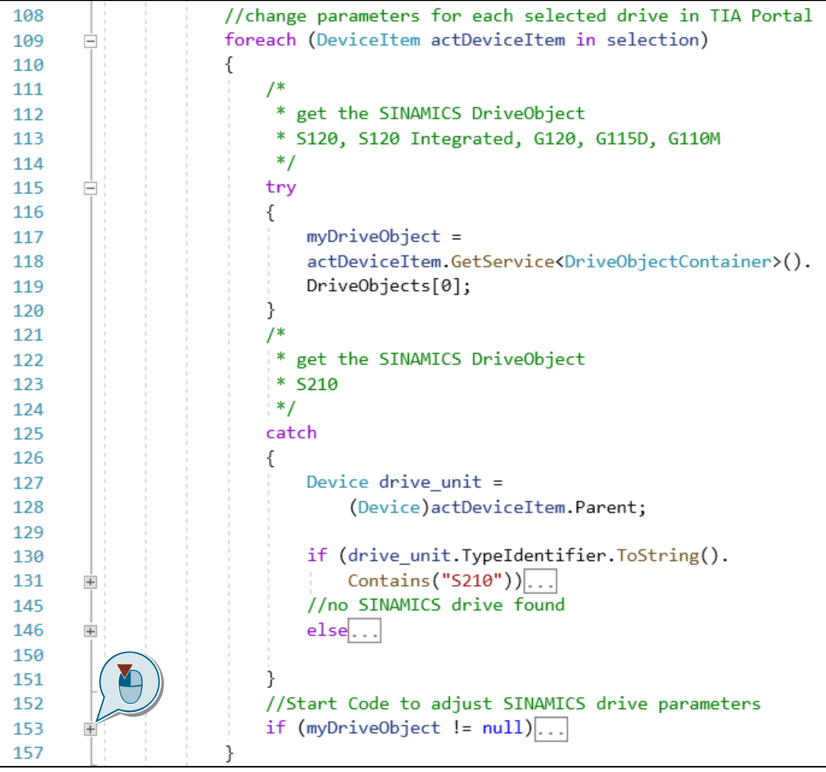
10.	
	`/// <summary> The method contains the program code of the TIA Add-In. Called whe ...`
	`1 reference`
	`private void OnDoSomething(MenuSelectionProvider<DeviceItem>`
	`    menuSelectionProvider)`
	`{`
	`    DriveObject myDriveObject = null;`
	`    //Get the actual selected DeviceItem from TIA Portal`
	`    IEnumerable<DeviceItem> selection =`
	`        menuSelectionProvider.GetSelection<DeviceItem>();`
	`    //change parameters for each selected drive in TIA Portal`
	`    foreach (DeviceItem actDeviceItem in selection)[...]`
	`}`

<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>11.</td><td>



```
//change parameters for each selected drive in TIA Portal
foreach (DeviceItem actDeviceItem in selection)
{
    /*
     * get the SINAMICS DriveObject
     * S120, S120 Integrated, G120, G115D, G110M
     */
    try...
    //Start Code to adjust SINAMICS drive parameters
    if (myDriveObject != null)...
}
```

</td></tr>
<tr><td>12.</td><td>

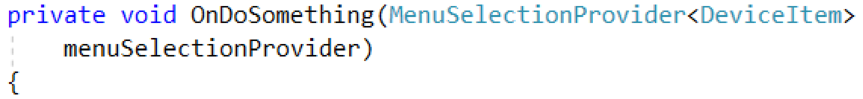


```
//change parameters for each selected drive in TIA Portal
foreach (DeviceItem actDeviceItem in selection)
{
    /*
     * get the SINAMICS DriveObject
     * S120, S120 Integrated, G120, G115D, G110M
     */
    try
    {
        myDriveObject =
        actDeviceItem.GetService<DriveObjectContainer>().
        DriveObjects[0];
    }
    /*
     * get the SINAMICS DriveObject
     * S210
     */
    catch
    {
        Device drive_unit =
            (Device)actDeviceItem.Parent;

        if (drive_unit.TypeIdentifier.ToString().
            Contains("S210"))...
        //no SINAMICS drive found
        else...

    }
    //Start Code to adjust SINAMICS drive parameters
    if (myDriveObject != null)...
}
```

</td></tr>
<tr><td>13.</td><td>

A brief description is now given as to how this template functions.

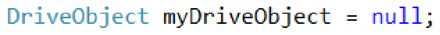
The "OnDoSomething" method of the "AddIn" class is executed after the start of the TIA add-in.



```
private void OnDoSomething(MenuSelectionProvider<DeviceItem>
    menuSelectionProvider)
{
```

</td></tr>
<tr><td>14.</td><td>

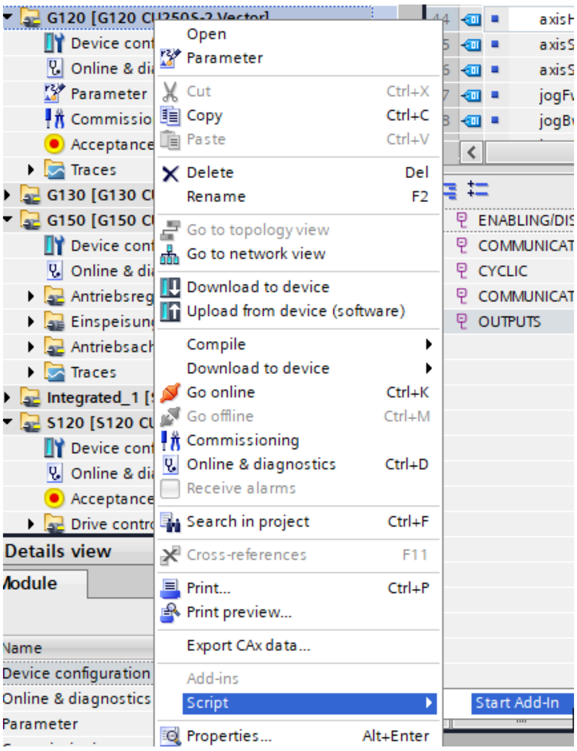
This means that the starting point of the program code to interconnect the drive parameters is method "OnDoSomething".
A variable, type "DriveObject" must now be created.



```
DriveObject myDriveObject = null;
```

</td></tr>
</table>

<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>15.</td><td>
In the TIA Portal or Startdrive, the TIA add-in is started on a drive axis or on a drive unit by right clicking on this object. For example, this then looks like this:

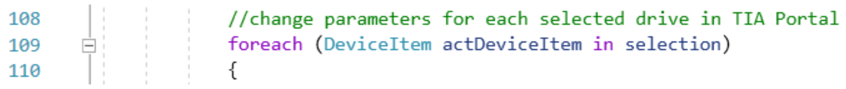
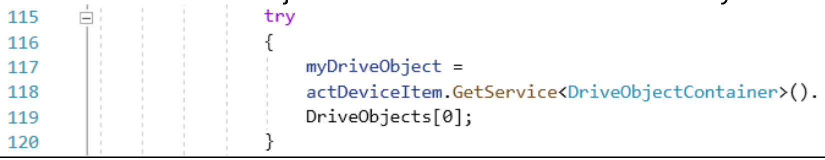


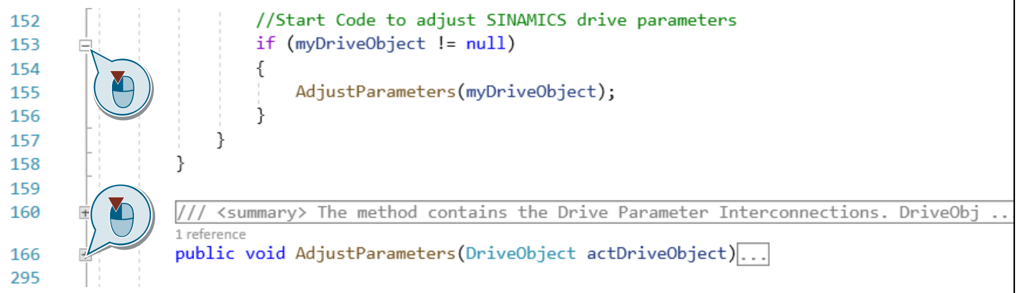
This means that the drive axis represents the entry point of the TIA. As a consequence, this selected drive axis must be saved in the program code.

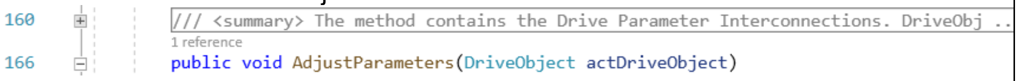
To do this, the selected drive axis is saved in the variable called "selection". The selected drive axis is found using function "menuSelectionProvider.GetSelection<DeviceItem>()".

```
//Get the actual selected DeviceItem from TIA Portal
IEnumerable<DeviceItem> selection =
    menuSelectionProvider.GetSelection<DeviceItem>();
```
</td></tr>
</table>

No.	Action

16.	The selected drive axis can then be accessed using variable "selection". However, it is also possible that the user had simultaneously selected several axes, and then the TIA add-in was started. Under certain circumstances, several drive axes can be edited simultaneously using TIA Portal Openness.
	Therefore it is possible that variable selection not only contains just one drive axis, but several.
	So the next step in the code is that a "foreach" loop is used to search all axes for "selection".
	
	`//change parameters for each selected drive in TIA Portal`
	`foreach (DeviceItem actDeviceItem in selection)`
	`{`
	However, to interconnect the drive parameters, "selection" is not used, but the associated "DriveObject" of the selected axes. To obtain the associated drive object, the service called "DriveObjectContainer" must be called using method "GetService". This is how you obtain the drive object of the selected drive axis in the TIA Portal. The drive object is saved in the variable called "myDriveObject".
	
	`try`
	`{`
	`    myDriveObject =`
	`    actDeviceItem.GetService<DriveObjectContainer>().`
	`    DriveObjects[0];`
	`}`

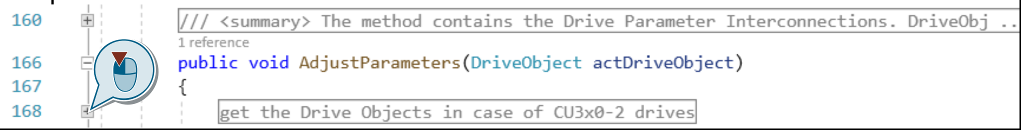
17.	To ensure that the template is user-friendly, the drive object is transferred to the method called "AdjustParameters". This means that users can specify all parameter interconnections in "AdjustParameters". To do this, the program code must be expanded again:
	
	`        //Start Code to adjust SINAMICS drive parameters`
	`        if (myDriveObject != null)`
	`        {`
	`            AdjustParameters(myDriveObject);`
	`        }`
	`    }`
	`}`
	`/// <summary> The method contains the Drive Parameter Interconnections. DriveObj ...`
	`1 reference`
	`public void AdjustParameters(DriveObject actDriveObject)...`
	Here, it can be seen that the drive object is transferred to method "AdjustParameters".

18.	The function of method "AdjustParameters" is now described in more detail.
	
	`/// <summary> The method contains the Drive Parameter Interconnections. DriveObj ...`
	`1 reference`
	`public void AdjustParameters(DriveObject actDriveObject)`
	The starting point for the additional operations is the drive object, shown here as "actDriveObject".
	The drive object is transferred to the functions of the StartdriveHelper library, which then e.g. write, read or interconnect parameters.

<table>
<thead>
<tr><th>No.</th><th>Action</th></tr>
</thead>
<tbody>
<tr><td>19.</td><td>

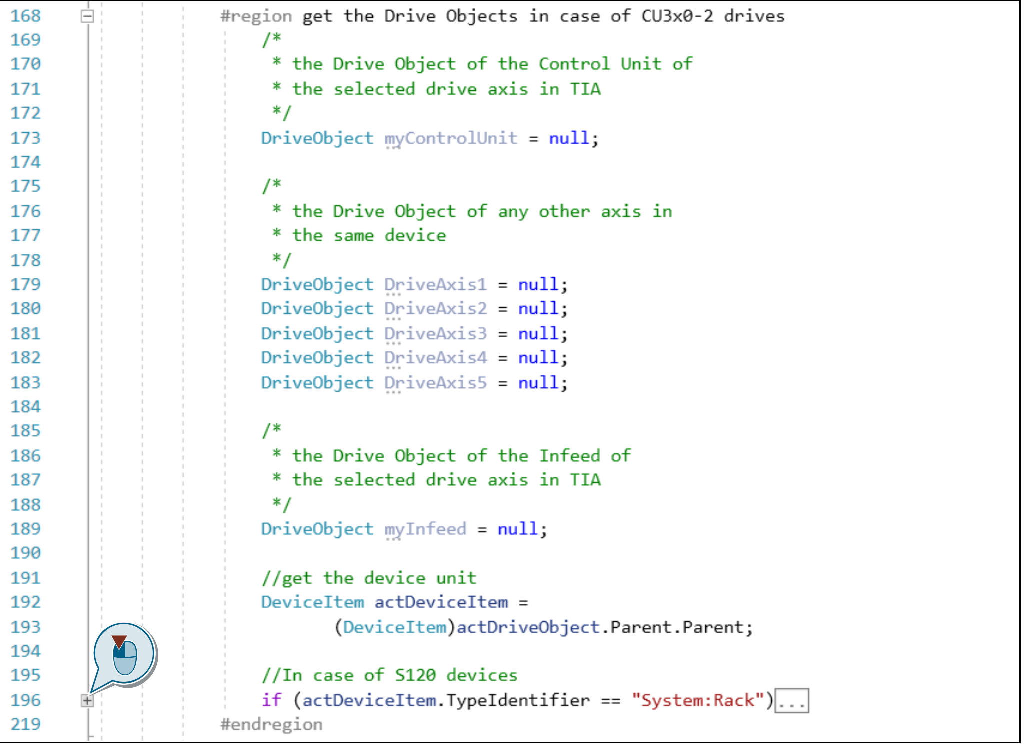
For S120 – CU320-2, the following applies:
The following must be noted, if interconnections have to be created between various drive axes, or interconnections to an axis other than the axis selected in Startdrive.

Another axis can be addressed on the CU320-2 using the drive object called DriveAxis1/DriveAxis2/ DriveAxis3/ DriveAxis4/ DriveAxis5. To link "DriveParameterComposition" - called "DriveAxis1/DriveAxis2/ DriveAxis3/ DriveAxis4/ DriveAxis5" - with a drive axis, the following has to be done in the template:



```
/// <summary> The method contains the Drive Parameter Interconnections. DriveObj ..
1 reference
public void AdjustParameters(DriveObject actDriveObject)
{
    get the Drive Objects in case of CU3x0-2 drives
```

</td></tr>
<tr><td>20.</td><td>

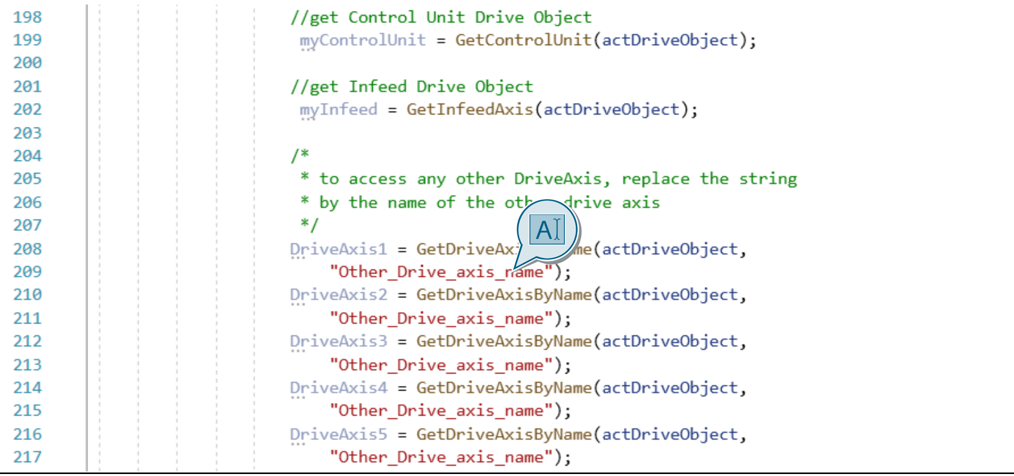


```
    #region get the Drive Objects in case of CU3x0-2 drives
        /*
         * the Drive Object of the Control Unit of
         * the selected drive axis in TIA
         */
        DriveObject myControlUnit = null;

        /*
         * the Drive Object of any other axis in
         * the same device
         */
        DriveObject DriveAxis1 = null;
        DriveObject DriveAxis2 = null;
        DriveObject DriveAxis3 = null;
        DriveObject DriveAxis4 = null;
        DriveObject DriveAxis5 = null;

        /*
         * the Drive Object of the Infeed of
         * the selected drive axis in TIA
         */
        DriveObject myInfeed = null;

        //get the device unit
        DeviceItem actDeviceItem =
                (DeviceItem)actDriveObject.Parent.Parent;

        //In case of S120 devices
        if (actDeviceItem.TypeIdentifier == "System:Rack")...
    #endregion
```

</td></tr>
</tbody>
</table>

<table>
<tr><th>No.</th><th>Action</th></tr>
<tr><td>21.</td><td>

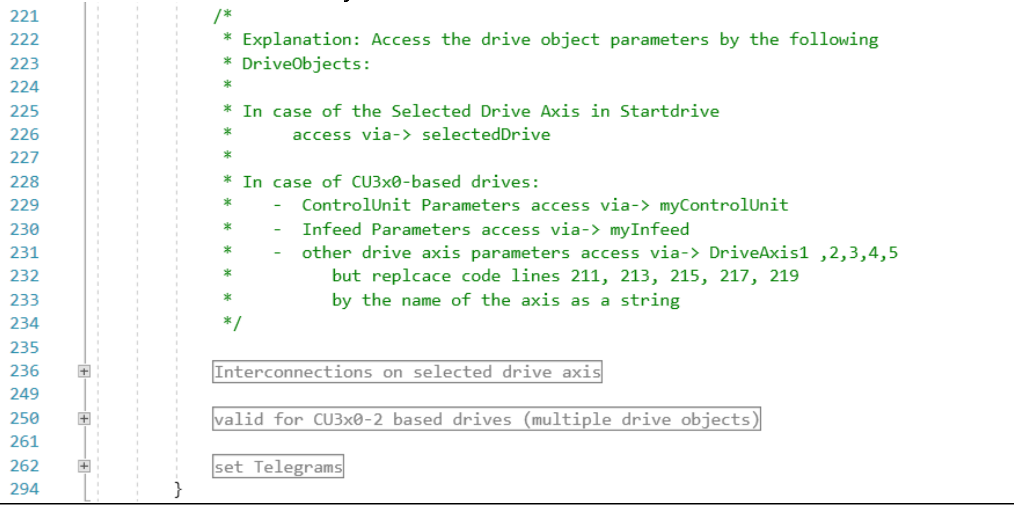
Here, instead of "Other_Drive_axis_name", the name of the axis that should be linked with drive object DriveAxis1 should be specified, i.e. the name of the axis on which the user wants to create interconnections. This name should be specified as a string. This means that all six servo or vector axes on a CU320-2 can be addressed via DriveAxis1/ DriveAxis2/ DriveAxis3/ DriveAxis4/ DriveAxis5.
The Control Unit and the infeed are found without a name.



```
198                 //get Control Unit Drive Object
199                  myControlUnit = GetControlUnit(actDriveObject);
200
201                 //get Infeed Drive Object
202                  myInfeed = GetInfeedAxis(actDriveObject);
203
204                 /*
205                  * to access any other DriveAxis, replace the string
206                  * by the name of the other drive axis
207                  */
208                 DriveAxis1 = GetDriveAxisByName(actDriveObject,
209                     "Other_Drive_axis_name");
210                 DriveAxis2 = GetDriveAxisByName(actDriveObject,
211                     "Other_Drive_axis_name");
212                 DriveAxis3 = GetDriveAxisByName(actDriveObject,
213                     "Other_Drive_axis_name");
214                 DriveAxis4 = GetDriveAxisByName(actDriveObject,
215                     "Other_Drive_axis_name");
216                 DriveAxis5 = GetDriveAxisByName(actDriveObject,
217                     "Other_Drive_axis_name");
```

</td></tr>
<tr><td>22.</td><td>

Several methods of the library are called in lines 235 – 294.



```
221             /*
222              * Explanation: Access the drive object parameters by the following
223              * DriveObjects:
224              *
225              * In case of the Selected Drive Axis in Startdrive
226              *      access via-> selectedDrive
227              *
228              * In case of CU3x0-based drives:
229              *    -  ControlUnit Parameters access via-> myControlUnit
230              *    -  Infeed Parameters access via-> myInfeed
231              *    -  other drive axis parameters access via-> DriveAxis1 ,2,3,4,5
232              *          but replcace code lines 211, 213, 215, 217, 219
233              *          by the name of the axis as a string
234              */
235
236  +          Interconnections on selected drive axis
249
250  +          valid for CU3x0-2 based drives (multiple drive objects)
261
262  +          set Telegrams
294         }
```

</td></tr>
</table>

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 and application examples – all information is accessible with just a few mouse clicks:
support.industry.siemens.com

Technical Support

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.

Queries can be sent to Technical Support using a web form:
www.siemens.com/industry/supportrequest

SITRAIN – Training for Industry

With our globally available training courses for Siemens products and solutions, we provide you with practical support, with innovative learning methods and with a customized training concept.

You can find out more about the training courses available as well as their locations and dates at:
www.siemens.com/sitrain

Service portfolio

Our service portfolio includes the following:

- Plant data services
- Spare parts services
- Repair services
- Field and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our service portfolio in the service catalog:
support.industry.siemens.com/cs/sc

Industry Online Support app

With the "Siemens Industry Online Support" app, you can obtain optimum support, even when you are on the move. The app is available for iOS and Android:
support.industry.siemens.com/cs/ww/de/sc/2067

4.2 Application Support

Siemens AG
Digital Industries
Factory Automation
Production Machines
DI FA PMA APC
Frauenauracher Str. 80
91056 Erlangen, Germany

mailto: tech.team.motioncontrol@siemens.com

4.3 Links and references

Table 4-1

No.	Subject
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to the entry page of the application example https://support.industry.siemens.com/cs/ww/en/view/109779415
\3\	

4.4 Change documentation

Table 4-2

Version	Date	Change
V1.0	11/2021	First Edition