Anybus .NET Bridge

Message Mode

USER MANUAL

SCM-1202-121 1.1 en-US ENGLISH

Connecting Devices™

Table of Contents

Page

1 Preface

1.1 About This Document

The Anybus .NET Bridge has two different modes, Message Mode and Streamer Mode.

This manual describes how to install, configure and use the Anybus .NET Bridge in Message Mode.

For additional related documentation and file downloads, please visit the support website at www.anybus.com/support.

1.2 Document History

Revision list

Version	Date	Description
1.0	2016-09-22	SCM-1202-098 First release
1.1	2018-03-14	SCM-1202-098 Major rewrite
1.2	2019-04-29	Changed Document ID from SCM-1202-098 to SCM-1202-120 for Streamer Mode and SCM-1202-121 for Message Mode Major rewrite

1.3 Related Documentation

The following documents with network and application development specific information are additional to this user manual.

To download the documents, please visit www.anybus.com/support.

Document	Author
.NET Bridge Startup Guide	HMS
.NET Bridge using TIA Portal with PROFINET Application Note	HMS
.NET Bridge using Beckhoff TwinCAT 3 with EtherCAT Application Note	HMS
.NET Bridge using Rockwell Studio 5000 with EtherNet/IP Application Note	HMS
.NET Bridge using TIA Portal with PROFIBUS Application Note	HMS

1.4 Document Conventions

Ordered lists are used for instructions that must be carried out in sequence:

1. First do this
2. Then do this

Unordered (bulleted) lists are used for:

- Itemized information
- Instructions that can be carried out in any order

...and for action-result type instructions:

► This action...

  → leads to this result

Bold typeface indicates interactive parts such as connectors and switches on the hardware, or menus and buttons in a graphical user interface.

```
Monospaced text is used to indicate program code and other
kinds of data input/output such as configuration scripts.
```

This is a cross-reference within this document: *Document Conventions, p. 4*

This is an external link (URL): www.hms-networks.com

ⓘ *This is additional information which may facilitate installation and/or operation.*

! This instruction must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.

⚠ **Caution**
This instruction must be followed to avoid a risk of personal injury.

⚠ **WARNING**
This instruction must be followed to avoid a risk of death or serious injury.

2 Description

2.1 Product Description

The .NET Bridge enables factory-floor data to be presented to .NET software applications.

Data sent from a PLC system can be used in .NET applications for statistics, analysis or maintenance.

Data can also be sent from the .NET environment to the PLC for use in daily operation.

The .NET Bridge is a state-of-the-art IT/OT bridge, easily bridging the gap between the operational technology (OT) and the information technology (IT).

The .NET Bridge can be used with a wide range of use cases, such as simple transfer of KPI values, advanced messages with structured data types and transfer of I/O data for big data.

The .NET Bridge acts as a translator between a function block in a PLC and a .NET object in a computer.

The .NET Bridge has two different modes:

Message Mode

A tag data structure is used and the communication is synchronized with handshake.

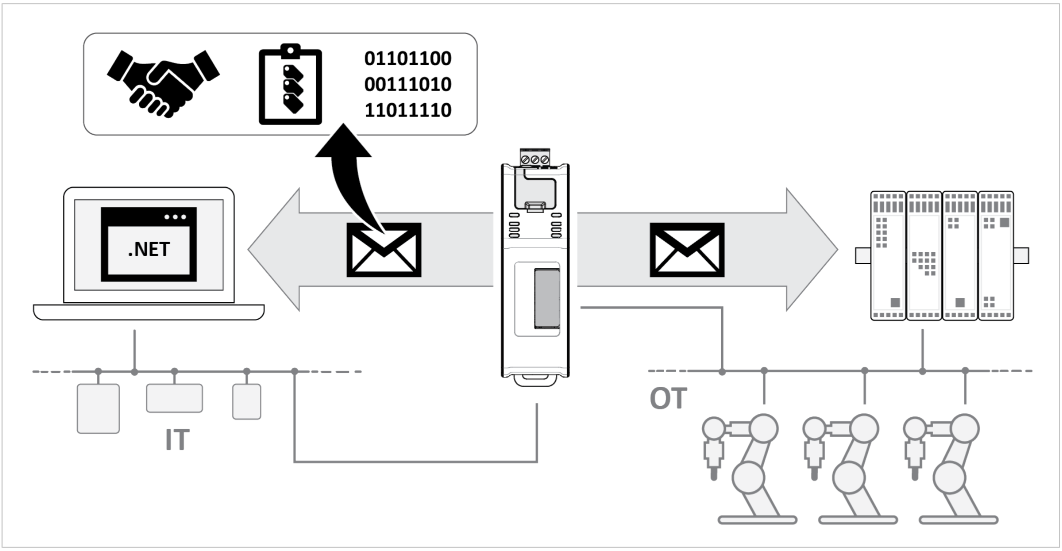


Fig. 1 Message Mode data exchange

Streamer Mode

Raw data is sent between the PLC and the .NET application.

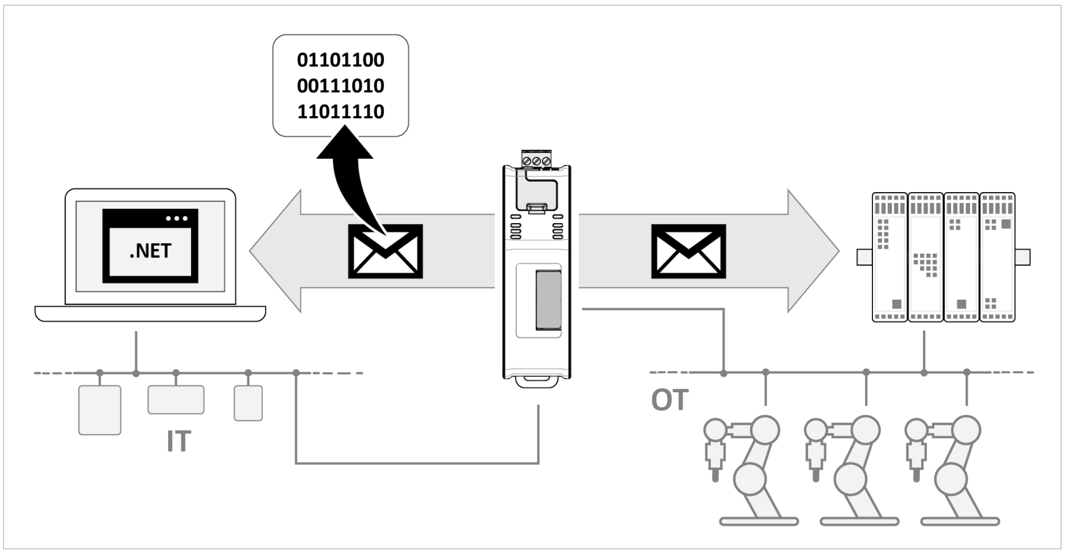


Fig. 2 Streamer Mode data exchange

2.2 Network Security Considerations

The communication between the .NET application and the .NET Bridge is not encrypted, similar to how the communication on the industrial network is not encrypted.

It is recommended to only communicate between the .NET application and the .NET Bridge in a private network.

A virtual private network (VPN) may be used to encrypt communication between the .NET application and the .NET Bridge, if it is needed to communicate over the internet.

3 Preparation

3.1 Prerequisites

3.1.1 Wall Mount Option

! When the .NET Bridge is used in an environment exposed to vibration, increased stability is required.

Use the *Wall-Mount Accessory Kit* and mount the device on a wall instead of a DIN rail.

The *Wall-Mount Accessory Kit* is ordered separately, please visit www.anybus.com for more information.

3.1.2 Required HMS Software

The following HMS software are required:

- Anybus .NET Bridge Setup: Includes Windows-based software needed during the message structure design process and to simulate the .NET application and the PLC application.
- IPconfig: A Windows-based software for configuration of TCP/IP settings in HMS devices.

Download the software applications at www.anybus.com/support.

3.1.3 Required Third Party Software

When developing .NET applications, the following third party software are required:

- Microsoft Excel, or equivalent software that supports the Office Open XML Workbook (xlsx) file format. Needed when using the .NET Bridge Message Mode.
- Microsoft Visual Studio
- Microsoft .NET Framework SDK version 4.5 or later.

On the production computer where the .NET application is installed, only the .NET Framework 4.5 (or later) runtime is required.

.NET Core Runtime is not supported.

3.2 Software Installation

3.2.1 Installing Anybus .NET Bridge Setup

Anybus .NET Bridge Setup software tools are used to generate configuration files and to simulate .NET applications and PLC applications.

Before You Begin

Visit www.anybus.com/support and download the *Anybus .NET Bridge Setup* zip file.

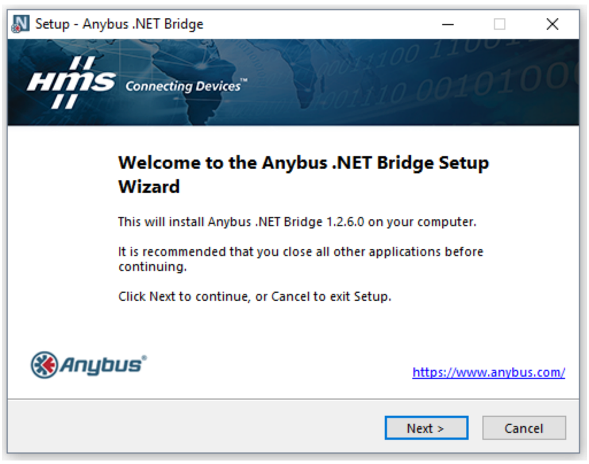
Before starting the installation, it is recommended that you close all other applications.

Procedure

Installing Anybus .NET Bridge Setup software tools:

1. Unzip the *Anybus .NET Bridge Setup* zip file.
2. Double-click the *Anybus .NET Bridge Setup* application file.
3. The Setup Anybus .NET Bridge installer window appears.

   Click **Next** to begin installation.



Welcome to the Anybus .NET Bridge Setup Wizard

This will install Anybus .NET Bridge 1.2.6.0 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Fig. 3 Anybus .NET Bridge Setup Wizard

4. Follow the prompts in the Anybus .NET Bridge Setup Wizard to complete the installation.
5. Click **Finish**.

Result

→ The following Anybus .NET Bridge Setup software tools are installed on your computer:
  - Anybus .NET Bridge Generator
  - Anybus .NET Bridge Message Mode PLC Simulator
  - Anybus .NET Bridge Message Mode .NET Simulator
  - Anybus .NET Bridge Streamer Mode PLC Simulator
  - Anybus .NET Bridge Streamer Mode .NET Simulator

3.2.2 Installing IPconfig

IPconfig is used to configure the TCP/IP settings in HMS devices.

Before You Begin

Visit www.anybus.com/support and download the *IPconfig - Utility for module TCP/IP configuration* zip file.

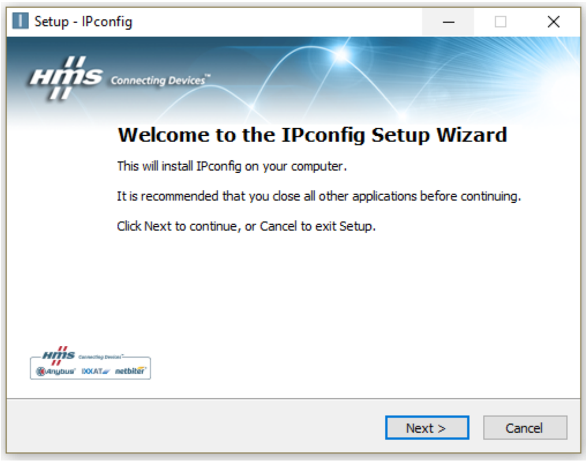
Before starting the installation, it is recommended that you close all other applications.

Procedure

Installing IPconfig:

1. Unzip the *IPconfig - Utility for module TCP/IP configuration* zip file.
2. Double-click the*IPconfig Setup* application file.
3. The Setup IPconfig installer window appears.

   Click **Next** to begin installation.


Setup - IPconfig

Welcome to the IPconfig Setup Wizard

This will install IPconfig on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Fig. 4 **IPconfig Setup Wizard**

4. Follow the prompts in the IPconfig Setup Wizard to complete the installation.
5. Click **Finish**.

Result

→ IPconfig is installed on your computer.

4 Communication Design

4.1 How the Communication Works

The data exchange between a .NET application and a PLC is made using messages.

The .NET Bridge acts as a bridge between the .NET library (DLL) in a PC and a function block (FB) in a PLC.

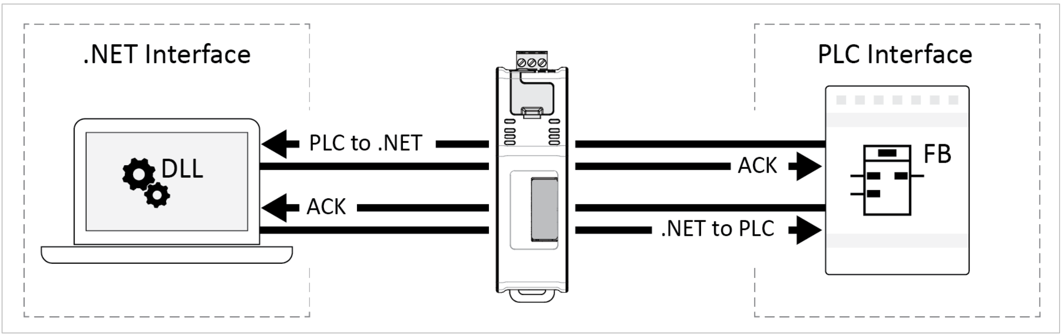


Fig. 5 **Communication between the .NET Interface and the PLC Interface**

The messages are sent in sequence over the same I/O data channel.

Handshake

The communication is synchronized with handshake.

When a message is received, the receiver always sends an acknowledge (ACK) back to the sender. The sender must receive an acknowledge (ACK) before it can send a new message.

Acknowledge (ACK) is always sent from the .NET application or the PLC application.

Messages and acknowledgements (ACK) are always forwarded transparently through the .NET Bridge to the .NET application and the PLC application.

4.2 The Communication Design Process

This is the process of designing the communication between the .NET application and the PLC application.

When these steps are completed the development of the .NET application and the PLC application can start.

1. **Plan the Message Exchange**

   The data exchange between the .NET application and the PLC application is made using messages.

   The incoming and outgoing messages to be exchanged and how these messages are to be exchanged must be determined by the .NET programmer and PLC programmer together.

2. **Map the Messages Exchange**

   Define and describe the message exchange in the *.NET Bridge Configuration Spreadsheet Template*.

   The spreadsheet template is retrieved from the Anybus .NET Bridge Generator.

3. **Generate Configuration Files**

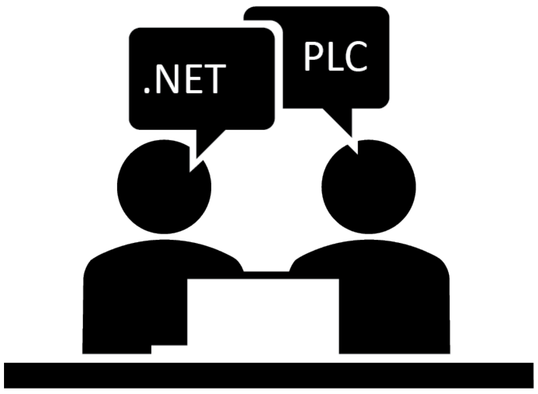
   The *.NET Bridge Configuration Spreadsheet* with the mapped messages is used to create configuration files; C# source code, .NET library, function blocks and device description files needed when developing the .NET application and the PLC application.

   The configuration files are generated by Anybus .NET Bridge Generator.

4.3 Plan the Message Exchange

The messages exchange between the .NET application and the PLC application are defined and described in the *.NET Bridge Configuration Spreadsheet Template*.

Names and data types according to guidance in the *Configuration Spreadsheet Template* must be agreed between the .NET programmer and PLC programmer.



For information about the message setup in the *Configuration Spreadsheet Template*, refer to *Map the Messages Exchange, p. 13*.

To develop the .NET application and the PLC application configuration files are needed.

The configuration files are created in Anybus .NET Bridge Generator using the *Configuration Spreadsheet* with the defined message setup.

For information about the configuration files, refer to *Configuration Files, p. 20*.

4.4 Map the Messages Exchange

4.4.1 Get the .NET Bridge Configuration Spreadsheet Template

Retrieve the *.NET Bridge Configuration Spreadsheet Template* from Anybus .NET Bridge Generator.

Before You Begin

The *Configuration Spreadsheet Template* is available in two variants:

- *Simplified template*: Supports a single message channel and basic settings.
- *Advanced template*: Supports multiple message channels and advanced settings.

Ensure that the following software are installed on your computer:

- Anybus .NET Bridge Generator
- Microsoft Excel, or equivalent software that supports the Office Open XML Workbook (xlsx) file format.

Procedure

1. Open Anybus .NET Bridge Generator.
2. Choose **Get Excel Template**.

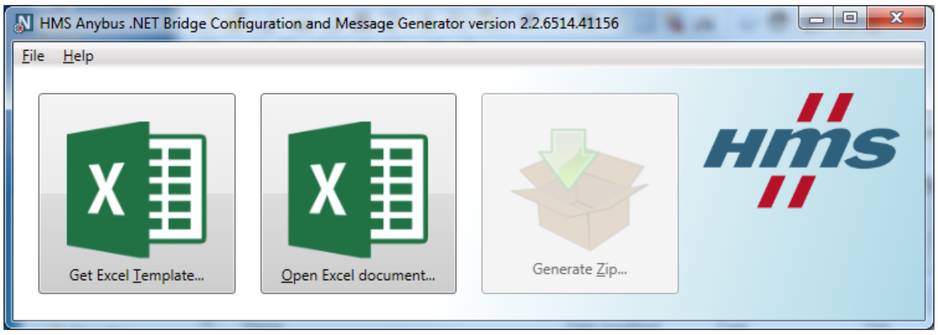
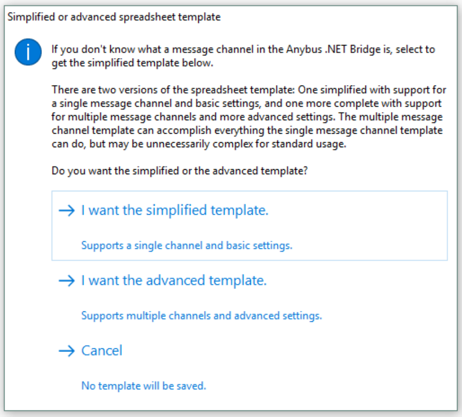

HMS Anybus .NET Bridge Configuration and Message Generator version 2.2.6514.41156
File
Help
Get Excel Template...
Open Excel document...
Generate Zip...

Fig. 6 **Anybus .NET Bridge Generator main window**

3. Choose the template you want to use, the *simplified template* or the *advanced template*.



Simplified or advanced spreadsheet template

If you don't know what a message channel in the Anybus .NET Bridge is, select to get the simplified template below.

There are two versions of the spreadsheet template: One simplified with support for a single message channel and basic settings, and one more complete with support for multiple message channels and more advanced settings. The multiple message channel template can accomplish everything the single message channel template can do, but may be unnecessarily complex for standard usage.

Do you want the simplified or the advanced template?

→ I want the simplified template.

Supports a single channel and basic settings.

→ I want the advanced template.

Supports multiple channels and advanced settings.

→ Cancel

No template will be saved.

Fig. 7 **Choose template type**

4. In the *Save Spreadsheet Template As* dialog box, choose where to save the file and click **Save**.

4.4.2 About the .NET Bridge Configuration Spreadsheet Template

Use the *.NET Bridge Configuration Spreadsheet Template* to define and describe the message exchange between the .NET application and the PLC application.

Before You Begin

The *.NET Bridge Configuration Spreadsheet Template* is available in two variants:

- **Simplified template:** Supports a single message channel and basic Bridge setup settings.

  The *simplified template* is sufficient for most applications.

- **Advanced template:** Supports multiple message channels and advanced Bridge setup settings.

  Use the *advanced template* only if you need to adjust the default Bridge setup setting values and if there is a need for more than one message channel.

Consider the following when filling out the *Configuration Spreadsheet Template*:

Enter names as single words, do not use whitespace.

Incorrectly entered data appear in red.

Leave unused rows blank.

If needed you may insert more rows.

Message Setup Parameters:

For detailed information about the message setup parameters, refer to *.NET Bridge Message Settings, p. 51*.

4.4.3 Bridge Setup

Before You Begin

Entering a Bridge type name is mandatory.

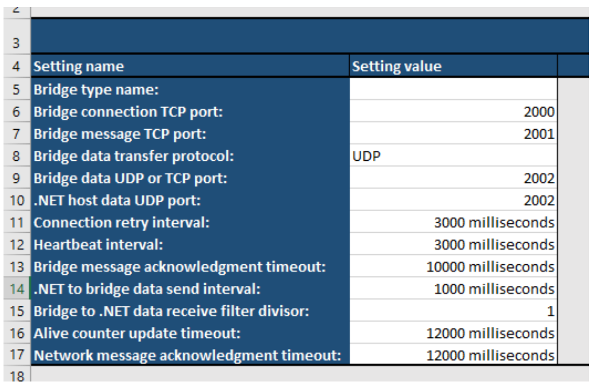
The .NET Bridge Configuration Spreadsheet Template comes with default Setting values.

For most applications, it is not necessary to change the default Setting values.

Names (bridge type name, channel names, message names and parameter names) can only be adjusted in the.NET Bridge Configuration Spreadsheet.

If names are adjusted, updated configuration files must be generated by Anybus .NET Bridge Generator and replaced in the .NET application and the PLC application.

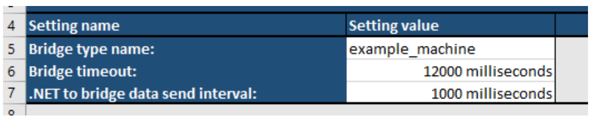
Advanced template Bridge setup



Setting name	Setting value
Bridge type name:	
Bridge connection TCP port:	2000
Bridge message TCP port:	2001
Bridge data transfer protocol:	UDP
Bridge data UDP or TCP port:	2002
.NET host data UDP port:	2002
Connection retry interval:	3000 milliseconds
Heartbeat interval:	3000 milliseconds
Bridge message acknowledgment timeout:	10000 milliseconds
.NET to bridge data send interval:	1000 milliseconds
Bridge to .NET data receive filter divisor:	1
Alive counter update timeout:	12000 milliseconds
Network message acknowledgment timeout:	12000 milliseconds

Fig. 8 Example, Default Setting values in the advanced template

Simplified template Bridge setup



Setting name	Setting value
Bridge type name:	example_machine
Bridge timeout:	12000 milliseconds
.NET to bridge data send interval:	1000 milliseconds

Fig. 9 Example, Default Setting values in the simplified template

In the *simplified template* only *Bridge type name*, *Bridge timeout* and *.NET to bridge data send interval* are available.

When the configuration files are created, the other *Setting values* are included in the C# source code file and the DLL file, built from the C# code, and set to their default values.

The *Setting values* are the default values.

The .NET application can if necessary change these values, except the *Bridge type name*, to other values when it connects to the.NET Bridge. For more information, refer to chapter *Adjusting the Setting Values, p. 19*.

Procedure

- Enter a *Bridge type name*.
- If needed, change the default *Setting values*.

  For detailed information, refer to *.NET Bridge Message Settings, p. 51*.

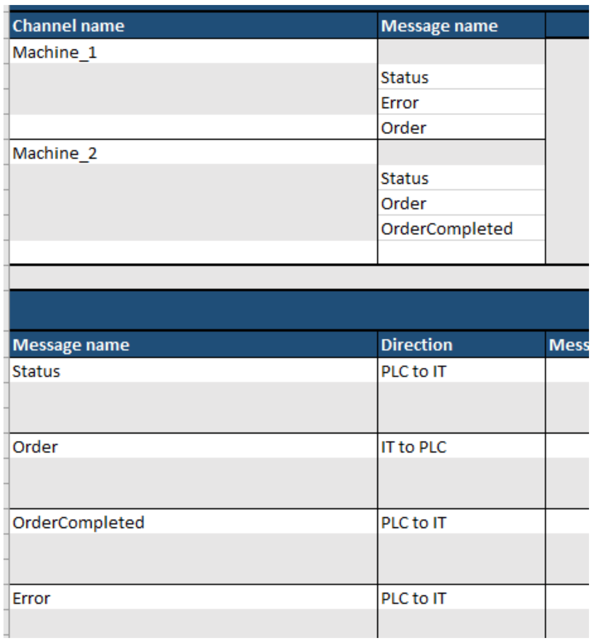
4.4.4 Channel Setup

Before You Begin

The Channel setup section is optional and only available in the advanced .NET Bridge Configuration Spreadsheet Template.

The default setting is one single channel for all messages defined.

To define a channel, you must enter both a Channel name and a Message name. The Message name must refer to a Message name specified in the Message setup section.



Channel name	Message name
Machine_1	
	Status
	Error
	Order
Machine_2	
	Status
	Order
	OrderCompleted

Message name	Direction	Mess
Status	PLC to IT	
Order	IT to PLC	
OrderCompleted	PLC to IT	
Error	PLC to IT	

Fig. 10 Example, Channel Setup

If no channel is specified, a single default channel will be created supporting all messages that you specify in the *Message setup* section.

Only one message can be sent at a time in a channel.

If you want to send multiple messages at the same time, you must add channels.

The advantage of using multiple channels is, for example, that data can be read from or instructions can be sent to multiple machines at the same time.

The bandwidth is also better utilized, when a larger amount of data is sent at the same time.

Data Size

Data from all the channels are included and sent in the same data package between the PC and the .NET Bridge.

For UDP, the maximum size of a data package is 1492 bytes.

For TCP, the maximum size of a data package is 1480 bytes.

Each channel contains the messages and the handshake for each message.

The handshake size is 3 bytes for each message.

Procedure

For each channel that should be created:

- Enter a *Channel name*, the name of the channel, in the next empty row.
- For each message to be sent over the channel, enter a *Message name*, the name of a message type, in the next empty row.

For technical support, please refer to www.anybus.com/support.

4.4.5 Message Setup

Describe the messages to be sent between the .NET application and the PLC application.

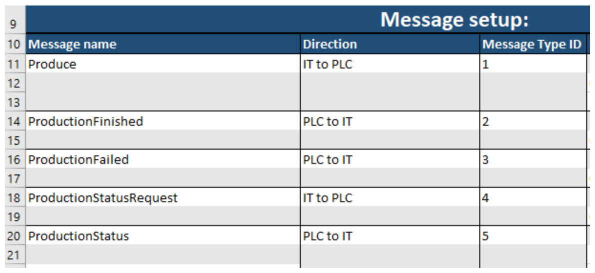
Before You Begin

The maximum size of a message is 251 bytes, including 3 bytes for the handshake.

For PROFIBUS the maximum size of a message is 239 bytes, including 5 bytes for the handshake.

Procedure

Define *Message name*, *Direction* and *Message Type ID*:



9	Message setup:		
10	Message name	Direction	Message Type ID
11	Produce	IT to PLC	1
12			
13			
14	ProductionFinished	PLC to IT	2
15			
16	ProductionFailed	PLC to IT	3
17			
18	ProductionStatusRequest	IT to PLC	4
19			
20	ProductionStatus	PLC to IT	5
21			

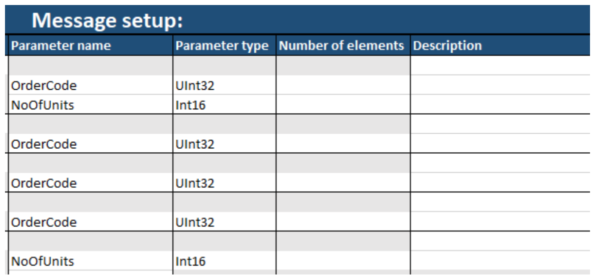
Fig. 11 **Example, Message setup: Name, Direction and Type ID**

- Enter *Message name*, the name of the message.
- Enter the direction by selecting *IT to PLC* or *PLC to IT* from the drop-down menu.

  Direction specifies if the message is sent from the PLC side to the IT side or from the IT side to the PLC side.
- Enter a ID number to identify the *Message Type ID*.

  The ID number must be unique for each message direction, *IT to PLC* and *PLC to IT*.

Define Parameter name, *Parameter type* and *Number of elements*:



Message setup:

Parameter name	Parameter type	Number of elements	Description
OrderCode	UInt32		
NoOfUnits	Int16		
OrderCode	UInt32		
OrderCode	UInt32		
OrderCode	UInt32		
NoOfUnits	Int16		

Fig. 12 **Example, Message setup: Parameter name, Type, Number of Elements and Description**

- Enter *Parameter name*, the name of a message parameter.
- Enter a *Parameter type* by selecting a parameter from the drop-down menu.
- Enter the *Number of elements*.

Adding the Number of elements is optional. If not added, the parameter becomes a scalar value.

Number of elements is the number of elements for arrays and the max length for strings.

Description:

- You can enter a description of the message or parameter.

  Adding a description is optional.

  The description will be included as a comment in the generated source code.

4.5 Adjusting the Setting Values

The *Bridge setup* setting values can be adjusted by the .NET application.

The setting values must be adjusted in the .NET application before connecting to the .NET Bridge.

Names (bridge type name, channel names, message names and parameter names) can only be adjusted in the.NET Bridge Configuration Spreadsheet.

If names are adjusted, updated configuration files must be created by Anybus .NET Bridge Generator and replaced in the .NET application and the PLC applications.

4.6 Configuration Files

4.6.1 Generate Configuration Files

The configuration files needed for the .NET programming and PLC programming are generated by the Anybus .NET Bridge Generator.

Before You Begin

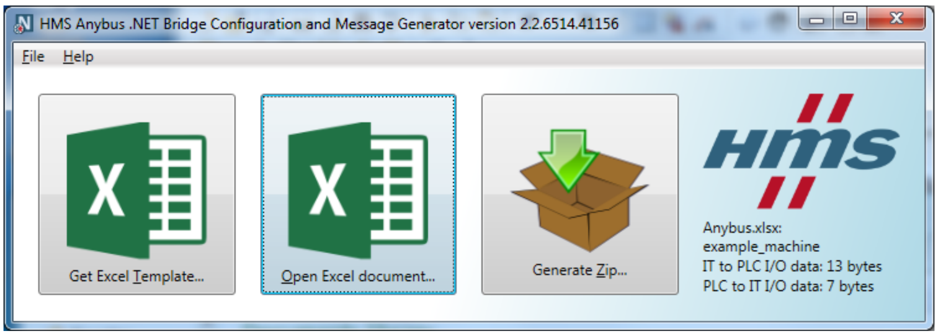
Ensure that you have the *.NET Bridge Configuration Spreadsheet*, with the defined messages, available.

Ensure that the following software are installed on your computer:

- Anybus .NET Bridge Generator
- Microsoft Excel, or equivalent software that supports the Office Open XML Workbook (xlsx) file format.

Procedure

Generate configuration files:


HMS Anybus .NET Bridge Configuration and Message Generator version 2.2.6514.41156
File
Help
Get Excel Template...
Open Excel document...
Generate Zip...
HMS

Anybus.xlsx:
example_machine
IT to PLC I/O data: 13 bytes
PLC to IT I/O data: 7 bytes

Fig. 13 Anybus .NET Bridge Generator main window

1. Open Anybus .NET Bridge Generator
2. Click **Open Excel Template**.
3. In the Open dialog box, select your *.NET Bridge Configuration Spreadsheet* file and click **Open**.
4. Choose **Generate Zip**.
5. In the Save zip file for dialog box, choose where to save the zip file and click **Save**.

Result

The zip file contains the following folders with the configuration files:

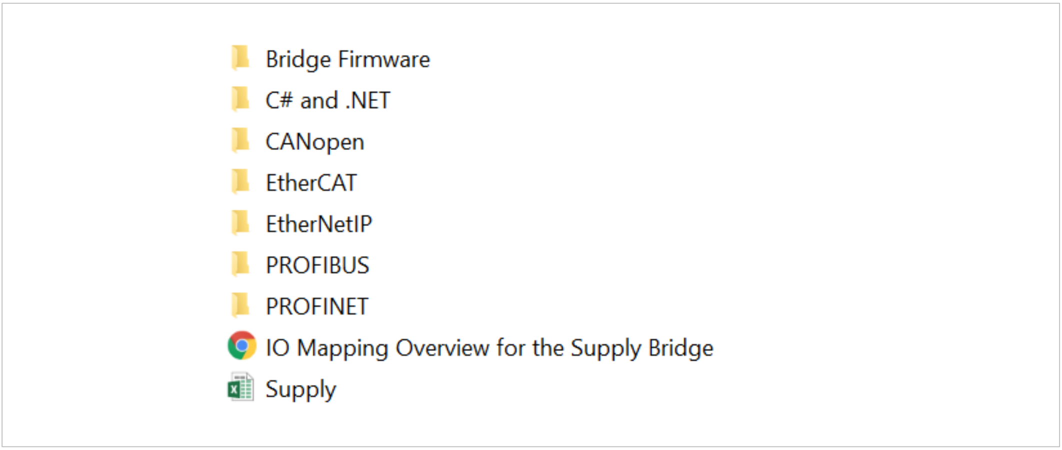

Bridge Firmware
C# and .NET
CANopen
EtherCAT
EtherNetIP
PROFIBUS
PROFINET
IO Mapping Overview for the Supply Bridge
Supply

Fig. 14 Zipped folder content, example

All the files needed for the .NET programming and PLC programming are now generated and you can continue with developing the .NET application and the PLC application.

4.6.2 .NET Bridge Firmware

Firmware update file for the .NET Bridge.

The firmware update is managed by the Firmware Manager II.

For more information about firmware update, refer to .NET Bridge Firmware Update, p. 73.

4.6.3 C# Source Code and .NET Library

The C# source code and .NET library files generated for developing the .NET application in Visual Studio are defined in the table below.

Only Microsoft Visual Studio is supported as the .NET development environment.

C# and .NET Files	Description
AnybusNetBridge.dll	DLL, Dynamic-link library, for the .NET platform that handles the communication with the PLC. The DLL is added as reference to Visual Studio projects using the .NET Bridge.
AnybusNetBridge.xml	The API documentation for *AnybusNetBridge.dll* in the XML format used by the Visual Studio *IntelliSense* features.
AnybusNetBridge.chm	The API documentation for *AnybusNetBridge.dll* in the Compiled HTML Help file format, used by *Microsoft HTML Help*.
C# file	The C# source code is based on the messages defined in the *.NET Bridge Configuration Spreadsheet*. The name of the file is based on the *Bridge type name* defined in the *.NET Bridge Configuration Spreadsheet*. The C# file is added as an *Existing Item* in the Visual Studio project.
DLL file Built version of the C# file	The DLL file is a built version of the C# file and has the same name as the C# file. The DLL file is added as an *Existing Reference* in Visual Studio.

4.6.4 Device Description and Function Block

Device Description

The *.NET Bridge device description* files are created for the industrial networks supported by the .NET Bridge.

The device description file is imported in the PLC environment. The device description file helps you to identify and include the .NET Bridge in the industrial network.

Function Block

Function blocks are created for the PLC environments.

The function block handles the communication between the PLC application and the .NET application.

Supported Development Environments and Industrial Networks

Industrial Network	PLC Development Environment	Device Description File	Function Block
EtherCAT	Beckhoff TwinCAT 3	ESI	*LIBRARY* ladder logic file
EtherNet/IP	Rockwell Studio 5000	EDS	*L5X* function block file
PROFIBUS	Siemens TIA Portal	GSD	*ZAL13* function block file
PROFINET	Siemens TIA Portal	GSDML	*ZAL13* function block file

If your PLC development environment is not supported you need to develop the function block.

For information about the general function block, refer to General Function Block Layout, p. 62.

4.6.5 Anybus .NET Bridge I/O Mapping Overview

The *Anybus .NET Bridge I/O Mapping Overview* HTML file contains an overview over the messages defined and described in the.*NET Bridge Configuration Spreadsheet*.

The file contains a summary of message lengths, message parameters and I/O data.

4.6.6 .NET Bridge Configuration Spreadsheet Original File

The original *.NET Bridge Configuration Spreadsheet* file with the defined and described messages.

5 Developing the .NET Application

5.1 .NET Programming

5.1.1 Importing References

Before You Begin

Only Microsoft Visual Studio is supported as the .NET development environment.

Ensure that you have access to the configuration files:

- The source code file based on the messages defined in your *.NET Bridge Configuration Spreadsheet* or the assembly (DLL) pre-built from the source code file.

  The *C# and .NET* folder contains both a *.cs* file and a *.dll* file.

  The files are named after the *bridge type*, defined in your *.NET Bridge Configuration Spreadsheet*.

  Both files have the same content. Use the format you prefer.
- The .NET Library dll file, *AnybusNetBridge.dll*.

For more information about the configuration files, refer to *Configuration Files, p. 20*.

Procedure

1. In Visual Studio, create a .NET Framework project.

Importing the *AnybusNetBridge.dll* file:

2. In the **Solution Explorer**, select the project ► right-click and select **Add** ► **References** ► **Browse**.
3. Browse to the *C# and .NET* folder.
4. Select the *AnybusNetBridge.dll* file and click **Add**.
5. Click **OK** to add the file in the project.

Use one of the following files and follow the instructions:

The DLL file named after the bridge type

Add the file as a References:

1. In the **Solution Explorer**, select the project ► right-click and select **Add References** ► **Browse**.
2. Browse to the *C# and .NET* folder.
3. Select the *DLL* file named after the *bridge type* and click **Add**.
4. Click **OK** to add the file in the project.

The source code file

Add the file as an Existing Item:

1. In the Solution Explorer, right-click on the project name and select **Add** ► **Existing Item**.
2. Browse to the *C# and .NET* folder.
3. Select the *DLL* file named after the *bridge type* and click **Add**, to add the file to the project.

Result

All references needed for the communication between the .NET application and the PLC application are now imported.

5.1.2 Namespaces

Procedure

Namespaces used in the code:

- `System.Net`: *Namespace* needed for some system functions.
- `Anybus.X`: The *namespaces* for types created from the *.NET Bridge Configuration Spreadsheet*.

  Replace X with the *Bridge type name*, defined in the *.NET Bridge Configuration Spreadsheet*.
- `HMS.AnybusNetBridge`: The *namespace* used by types in the *AnybusNetBridge.dll*, retrieved from the Anybus .NET Bridge Generator.

Example 1: Using directives

```
using System.Net;
using Anybus.X;
using HMS.AnybusNetBridge;
```

5.1.3 .NET Bridge Instance

Procedure

Add variables used by the program and create an instance of the .NET Bridge:

In the code example:

- IP address *192.168.1.3* is used, the default address to the .NET Bridge.

  Add the IP address for the .NET Bridge on the IT network where the .NET Bridge is installed.
- Replace X with the *Bridge type name*, defined in the *.NET Bridge Configuration Spreadsheet*.

Example 2: .NET Bridge `br` instance

```
string ipAddress = "192.168.1.3";
XBridge br = new XBridge(
IPAddress.Parse(ipAddress),
XBridge.DefaultConnectionTcpPort,
XBridge.DefaultDataUdpLocalPort);
```

5.1.4 Send Message

Procedure

Use a method to send a message to the PLC application:

- `br` is the instance of the .NET Bridge.
- The *Parameter names* and *Parameter types* are defined in the*.NET Bridge Configuration Spreadsheet*, *Message setup*.

In the code example:

- Replace X with the name of a message, defined in the *.NET Bridge Configuration Spreadsheet*.
- `OrderCode` and `NumOfUnits` are *Parameter names* examples.

Example 3: Posting a message

```
int orderCode = 42
short numOfUnits = 314
br.MessageChannel.PostOrderTelegram(orderCode,numOfUnits);
```

Result

If the message is sent correctly:

→ The function will return after an acknowledge (ACK) is received from the PLC application.

If an error occurs, the code throws one of the following exceptions:

Exception	Description
`Exception`	The base class for any type of exception.
`ArgumentNullException`	Data is null.
`InvalidOperationException`	The current state of the .NET Bridge prevents the message from being posted.
`HMS.AnybusNetBridge.FieldbusTimeoutException`	The message was not acknowledged in time.

5.1.5 Receive Message

Procedure

Create *event handlers:*:

► One *event handler* to receive communication exception information.

This is optional but highly recommended, as the status indicate if the .NET application is connected to the PLC.

► For each message the PLC may send to .NET: An event handler to get a callback when the PLC application has sent the message Y.

Listening to the message events are required to get the messages the PLC sends to the .NET application.

Y is a message defined in the *.NET Bridge Configuration Spreadsheet*, *Message setup*.

The *event handler* implementations are not shown for *brevity*

Example 4: Adding event handlers.

```
br.BridgeStatusChanged += Br_BridgeStatusChanged;
br.MessageChannel.YRecieved += MessageChannel_YRecieved;
br.CommunicationException += Br_CommunicationException;
```

Result

When a message is sent from the PLC application to the .NET application:

→ The event handler for this type of messages, `MessageChannel.YReceived`, is called.

→ The .NET application sends an acknowledge (ACK) to the PLC application after the message is received.

If an error occurs, the .NET application:

→ Raises the `CommunicationException` event.

→ Changes the bridge status to `Disconnected`.

5.1.6 .NET Bridge Status Changed

If the .NET Bridge state changes:

→ The .NET Bridge `BridgeStatus` value changes.

→ The .NET Bridge `BridgeStatusChanged` event is raised.

→ The .NET Bridge state is indicated by one of the following states:

State	Description
`Disconnected`	The .NET Bridge is not reachable or not connected.
`Connected`	The .NET Bridge is connected to the PLC side.
`PreOperational`	The .NET Bridge is connected to the PLC side and to the .NET side but no I/O data is exchanged.
`Operational`	The .NET Bridge is connected to the PLC side and the .NET side and I/O data is exchanged.
`Error`	General Error.

5.1.7 Start Communication

Procedure

Add code to start the communication between the .NET application and the PLC application:

- `br` is the instance of the .NET Bridge.

Example 5: Start communication

```
br.StartCommunication();
```

5.2 PLC Simulation

5.2.1 About PLC Simulation

Anybus .NET Bridge Message Mode PLC Simulator simulates PLC applications.

The .NET programmer can simulate a PLC application when developing and testing the .NET application.

No access to the PLC application is needed during the simulation.

Anybus .NET Bridge Message Mode PLC Simulator can only simulate bridges with one single message channel. For more information about channel setup, refer to Map the Messages Exchange, p. 13.

5.2.2 Setting up the .NET Bridge for PLC Simulation

Before you can start using the .NET Bridge PLC simulation you need to connect the .NET Bridge to the IT network and power.

Before You Begin

Ensure that the .NET Bridge IP settings are configured for the IT network.

For more information, refer to *TCP/IP Configuration, p. 55*.

Procedure

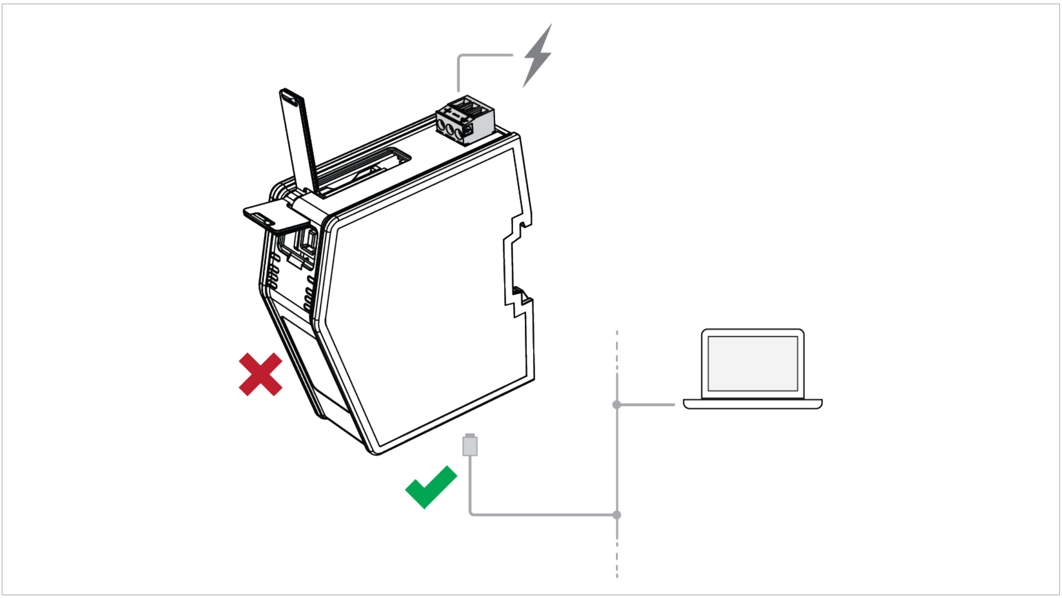


Fig. 15 **.NET Bridge connected to IT network and power**

1. Connect the .NET Bridge to the IT network via the IT Network Connectors, located at the bottom of the .NET Bridge.
2. Connect the .NET Bridge to power.

   For more information about the Power Connection, refer to *Connecting to Power, p. 46*.

Do not connect the .NET Bridge to the PLC.

LED indication:

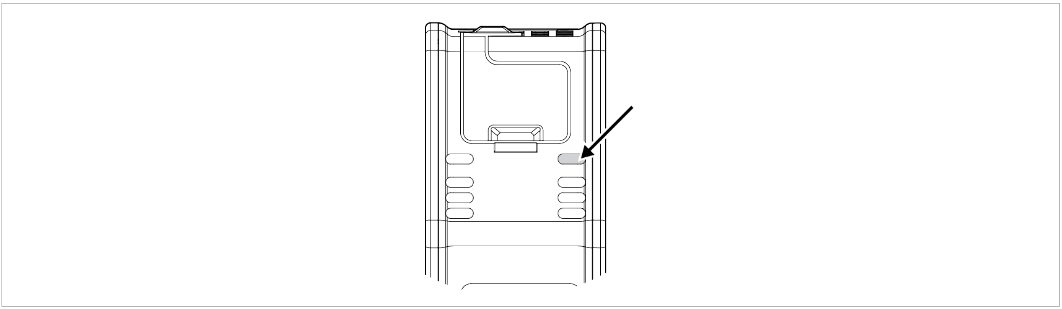


Fig. 16 **IT Status LED**

3. Check that the *IT Status LED* turn solid green.

5.2.3 Enable PLC Simulation

Before you can start the PLC simulation you must enable PLC simulation.

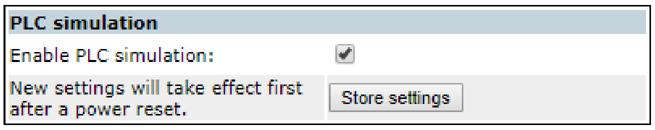
Before You Begin

Ensure that you have access to the .NET Bridge IP address on the IT network.

Procedure

Enable PLC Simulation:

1. In your web browser, type the .NET Bridge IP address and enter the Anybus .NET Bridge Web Interface start page.
2. In PLC simulation, check the **Enable PLC simulation** checkbox.
3. Click **Store settings**.



PLC simulation	
Enable PLC simulation:	✓
New settings will take effect first after a power reset.	Store settings

Fig. 17 **Enable PLC simulation**

4. For changes to take effect, restart the .NET Bridge.

LED indication:

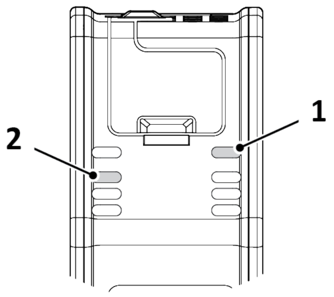


Fig. 18 **IT Status LED (1) and SIM PLC Simulation LED (2)**

5. Check that the *IT Status LED* (1) is solid green.
6. Check that the *SIM PLC Simulation LED* (2) is blinking green.

   For more information about the LED status indicators, refer to *LED Guide, p. 68*.

To Do Next

Simulation is now enabled, and you can proceed with the PLC simulation by starting the .NET application.

5.2.4 Starting the .NET Application

Before you can start the PLC simulation you must start the .NET application.

Procedure

In Visual Studio, start the .NET application:

1. Click **Start**, to start running the .NET application.

LED indication:

2. Check that the *IT Status LED* (1) is solid green.
3. Check that the *SIM PLC Simulation LED* (2) turn solid green.

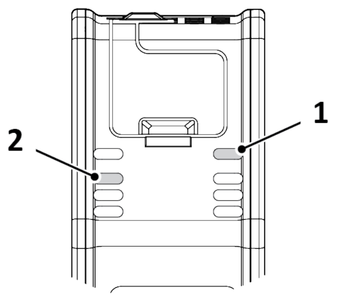


Fig. 19 IT Status LED (1) and SIM PLC Simulation LED (2)

To Do Next

The .NET application is now ready to start sending and receiving messages.

5.2.5 Connect PLC Simulator to .NET Bridge

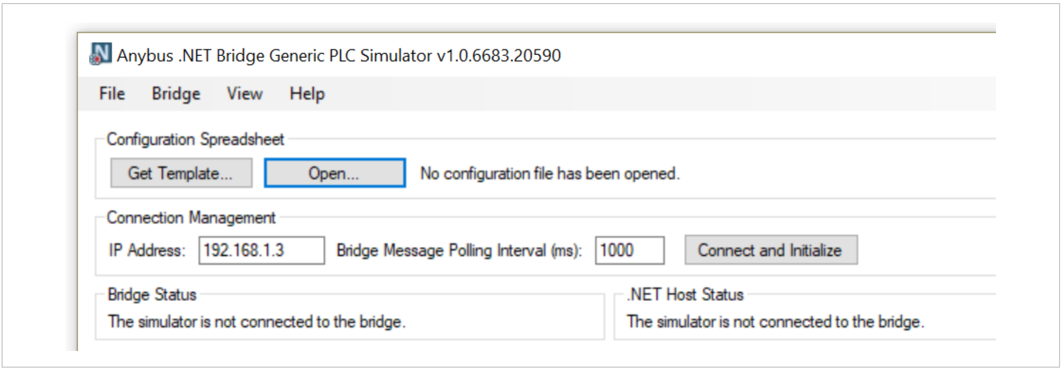
Before You Begin

Ensure that:

- PLC simulation is enabled in the Anybus .NET Bridge Web Interface.
- you have access to the .NET Bridge IP address, on the IT network.
- you have access to the *.NET Bridge Configuration Spreadsheet* with the defined messages.
- the Anybus .NET Bridge Message Mode PLC Simulator is installed on your computer.
- the .NET application is running.

Procedure

Connect and Initialize PLC simulation:



No configuration file has been opened.

IP Address: 192.168.1.3

Bridge Message Polling Interval (ms): 1000

Bridge Status: The simulator is not connected to the bridge.

.NET Host Status: The simulator is not connected to the bridge.

Fig. 20 Connect and Initialize the .NET Bridge

1. Open Anybus .NET Bridge Message Mode PLC Simulator.
2. In the Configuration Spreadsheet pane, click **Open**.
3. Browse the *.NET Bridge Configuration Spreadsheet* with the defines messages and click **Open**.
4. In the Connection Management pane, enter the IP address to the .NET Bridge on the IT network.
5. To connect to the .NET Bridge, click **Connect and Initialize**.
6. In the *Bridge Status* pane, verify that the status is *The simulation is enabled*.
7. Check that the .NET application is connected to the simulator:

   In the *.NET Host Status* pane, verify that the status is *A .NET host is connected to the bridge*.

Result

→ All message types are now imported and listed in the Message to post pane.

→ The .NET Bridge is connected to the PLC simulator.

5.2.6 Post and Receive Messages in PLC Simulator

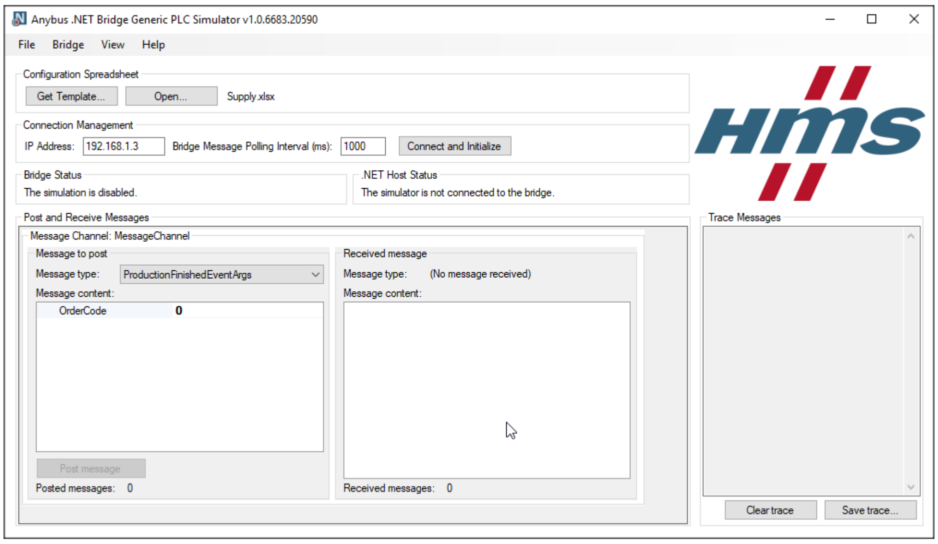
Send messages from the simulated PLC to the .NET application and view the messages received from the .NET application.

Before You Begin

- Ensure that the .NET application is running.

In Anybus .NET Bridge Message Mode PLC Simulator, ensure that the:

- Bridge status is *The simulation is enabled*.
- .NET Host Status is *A .NET host is connected to the bridge*.


Anybus .NET Bridge Generic PLC Simulator v1.0.6683.20590

Supply.xlsx

IP Address: 192.168.1.3
Bridge Message Polling Interval (ms): 1000

Bridge Status
The simulation is disabled.

.NET Host Status
The simulator is not connected to the bridge.

Message Channel: MessageChannel

Message type: ProductionFinishedEventArgs
Message content:
OrderCode 0

Posted messages: 0

Message type: (No message received)
Message content:
Received messages: 0

Fig. 21 Example, PLC Simulator main window

Post messages from the simulated PLC

Procedure

In the Message to post pane you can send a messages from the simulated PLC application to the .NET application:

1. Select *Message type*.
2. Add a *Message content*.
3. To send a message, click **Post message**.

Result

→ The message is sent from the simulated PLC to the .NET application.

The method `MessageChannel_YRecieved` is called.

In the .NET application the event *Y* is raised on the `MessageChannel` property of the .NET Bridge instance. The event *Y* is the message type selected for posting in the PLC simulator.

Refer to *Receive Message, p. 25*.

► In the *Message Channel* pane, the *Posted messages* value is increased.

Receive message in the simulated PLC

Procedure

When a message has been sent from the .NET application to the simulated PLC, the received message is shown in the *Received Message* pane and the number of revived messages is increased.

6 Developing the PLC Application

6.1 .NET Simulation Startup

6.1.1 About .NET Simulation

Anybus .NET Bridge Message Mode .NET Simulator simulates the .NET application.

The PLC programmer can simulate the .NET application when developing and testing the PLC application.

No access to the .NET application is needed during the simulation.

6.1.2 Setting up the .NET Bridge for .NET Simulation

Before you can start using the .NET simulation you need to connect the .NET Bridge to IT network, PLC and power.

Before You Begin

Ensure that *PLC simulation* is disabled in the Anybus .NET Bridge Web Interface. The *Enable PLC simulation* checkbox must be unchecked.

Ensure that the .NET Bridge IP settings are configured for the IT network and the industrial network.

For more information, refer to *TCP/IP Configuration, p. 55*.

Procedure

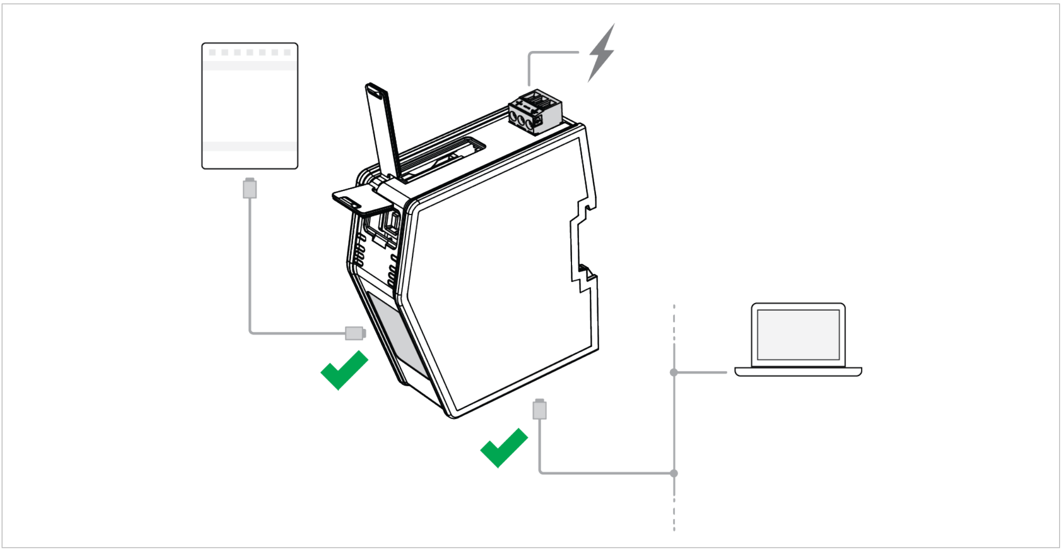


Fig. 22 .NET Bridge connected to IT network and PLC

Do not connect the .NET Bridge to the industrial network.

1. Connect the .NET Bridge to the IT network via the IT Network Connectors, located at the bottom of the .NET Bridge.
2. Connect the .NET Bridge to the PLC via the Industrial Network Connectors, located at the front of the .NET Bridge.
3. Connect the .NET Bridge to power.

   For more information about the Power Connection, refer to *Connecting to Power, p. 46*.

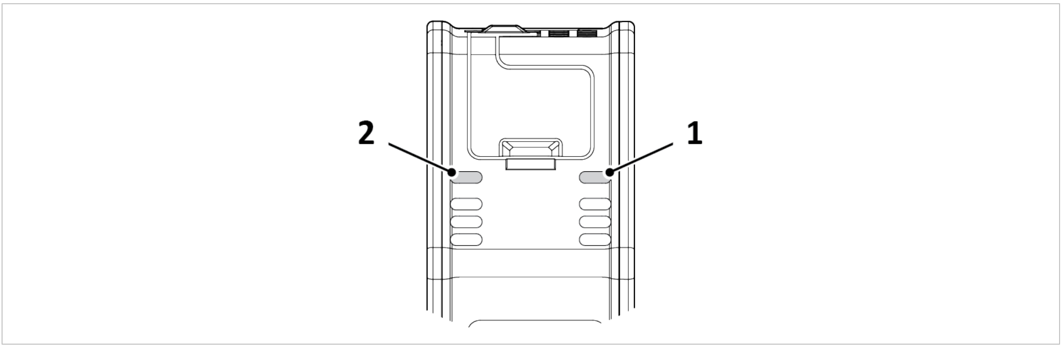


Fig. 23 **IT Status LED (1) and OT Status LED (2)**

LED indication:

4. Check that the *IT Status LED* (1) turn solid green.
5. Check that the *OT Status LED* (2) turn solid green.

6.1.3 Connect .NET Simulator to .NET Bridge

Connect and initialize the .NET Bridge in Anybus .NET Bridge Message Mode .NET Simulator.

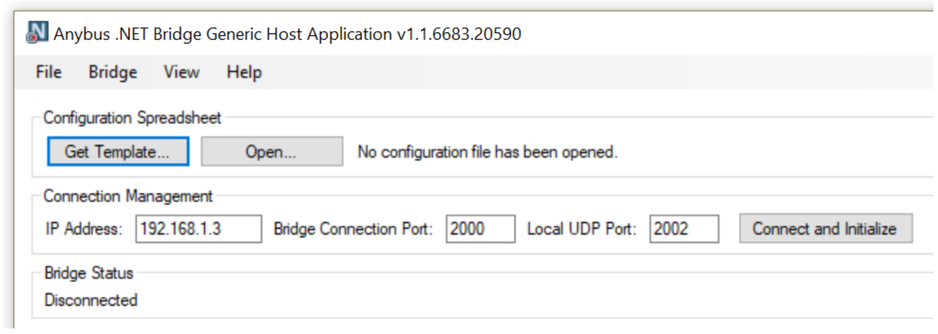
Before You Begin

Ensure that:

- you have access to the .NET Bridge IP address, on the IT network.
- you have access to the *.NET Bridge Configuration Spreadsheet* with the defined messages.
- the Anybus .NET Bridge Message Mode .NET Simulator simulator is installed on your computer.

Procedure

Connect and Initialize .NET simulation:


Anybus .NET Bridge Generic Host Application v1.1.6683.20590

No configuration file has been opened.

IP Address: 192.168.1.3 Bridge Connection Port: 2000 Local UDP Port: 2002

Disconnected

Fig. 24 **Connect and Initialize the .NET Bridge**

1. Open Anybus .NET Bridge Message Mode .NET Simulator.
2. In the Configuration Spreadsheet pane, click **Open**.
3. Browse the *.NET Bridge Configuration Spreadsheet* with the defines messages and click **Open**.
4. In the Connection Management pane, enter the IP address to the .NET Bridge on the IT network.
5. To connect to the .NET Bridge, click **Connect and Initialize** .
6. In *Bridge Status* pane, verify that the status change to *The simulation is enabled*.

Result

→ All message types are now imported and listed in the Message to post pane.

→ The .NET Bridge is connected to the PLC simulator.

6.2 PLC Programming

6.2.1 Including Configuration Files

Before You Begin

If Rockwell Studio 5000 is used, make sure it is in Offline mode.

Before You Begin

Ensure that you have access to the configuration files:

- The *Device description* file adapted for your PLC environment; ESI, EDS or GSD file.

  The device description file is used to identify and include the .NET Bridge in the industrial network.

- The *Function block* file adapted for your PLC environment.

  The function block handles the message exchange between the .NET application and the PLC application.

HMS provides function blocks for the following PLC environments:

Rockwell Studio 5000, Siemens TIA Portal and Beckhoff TwinCAT 3 (ladder logic).

If you are using another PLC environment you need to develop the function block.

For more information, refer to General Function Block Guide, p. 61

For detailed information about the configuration files, refer to *Configuration Files, p. 20*.

Procedure

In the PLC development environment, include the .NET Bridge in the industrial network:

Device Description

- Import the device description file and include it in the hardware structure.

Function Block

- Import the function block and add it in the PLC project.
- Create an instance of the function block in the PLC program.

For industrial network and PLC environment specific examples, refer to code examples, videos and application notes at www.anybus.com/support.

6.2.2 Connect Tags

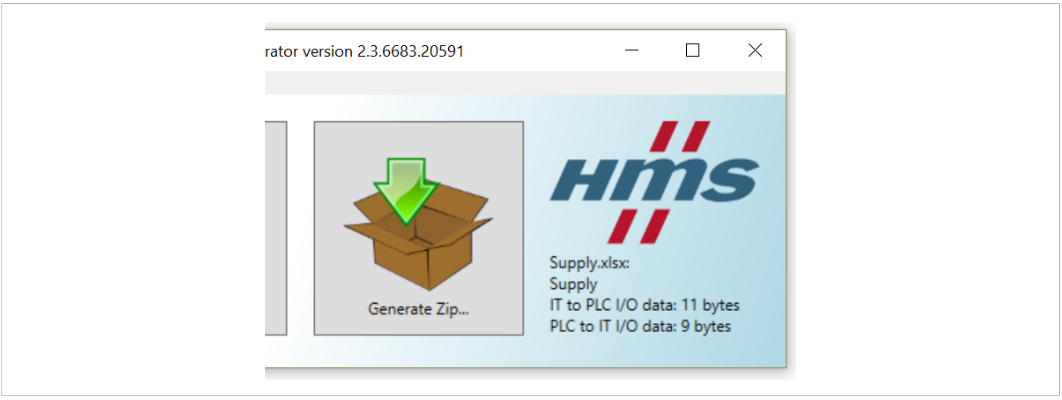
Connect the I/O tags to the corresponding tags on the function block.

For industrial network and PLC environment specific examples, refer to code examples, videos and application notes at www.anybus.com/support.

6.2.3 Input Data and Output Data Size

When the Configuration spreadsheet is opened in the Anybus .NET Bridge Generator the size of the input data and the output data, sent between the .NET application and the PLC application, show in the main window.

I/O Data size example



Supply.xlsx:
Supply
IT to PLC I/O data: 11 bytes
PLC to IT I/O data: 9 bytes

Fig. 25 **I/O Data size example**

In the total data size there are 3 bytes included for the handshake signals.

In some PLC development environments the handshake signals are hidden.

Input data to PLC

Tag	Signal Type	Input data to PLC, length
`AliveCounterToPlc`	Alive Counter	1 byte
`HandshakeToPlcReq`	Handshake	1 byte
`HandshakeToItAck`	Handshake	1 byte
`MessageTypeIdToPlc`	Message	2 bytes
`MessageToPlc`	Message	6 bytes
		IT to PLC I/O Data: 11 byte

Output data from PLC

Tag	Signal Type	Output data from PLC, length
`AliveCounterToIt`	Alive Counter	1 byte
`HandshakeToItAck`	Handshake	1 byte
`HandshakeToItReq`	Handshake	1 byte
`MessageTypeIdToIt`	Message	2 bytes
`MessageToIt`	Message	4 bytes
		PLC to IT I/O Data: 9 byte

6.2.4 Message Structure

This is a general example of the message structure in function block and ladder programming, how to receive and send messages between the PLC application and the .NET simulator.

For industrial network and PLC environment specific examples, refer to code examples, videos and application notes at www.anybus.com/support.

PLC Receives a Message from the IT System

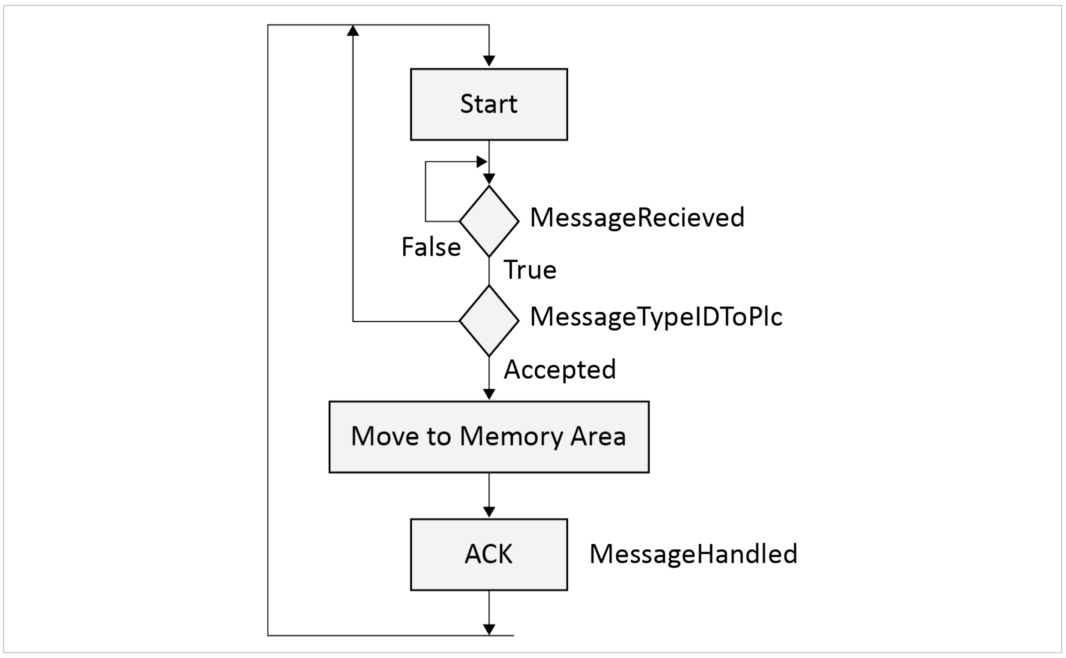


Fig. 26 PLC application receives a message from the .NET application

Receive message from .NET application:

1. Wait for a positive edge of `MessageReceived` signal on the function block.

   Read from output pin on the function block.

2. Check the incoming message ID is valid by setting `MessageTypeIDToPlc`.
3. Move the message to the PLC memory area, corresponding to the message ID.
4. Send acknowledgement (ACK) to the IT side by setting the signal `MessageHandled`.

PLC Sends a Message to the IT System

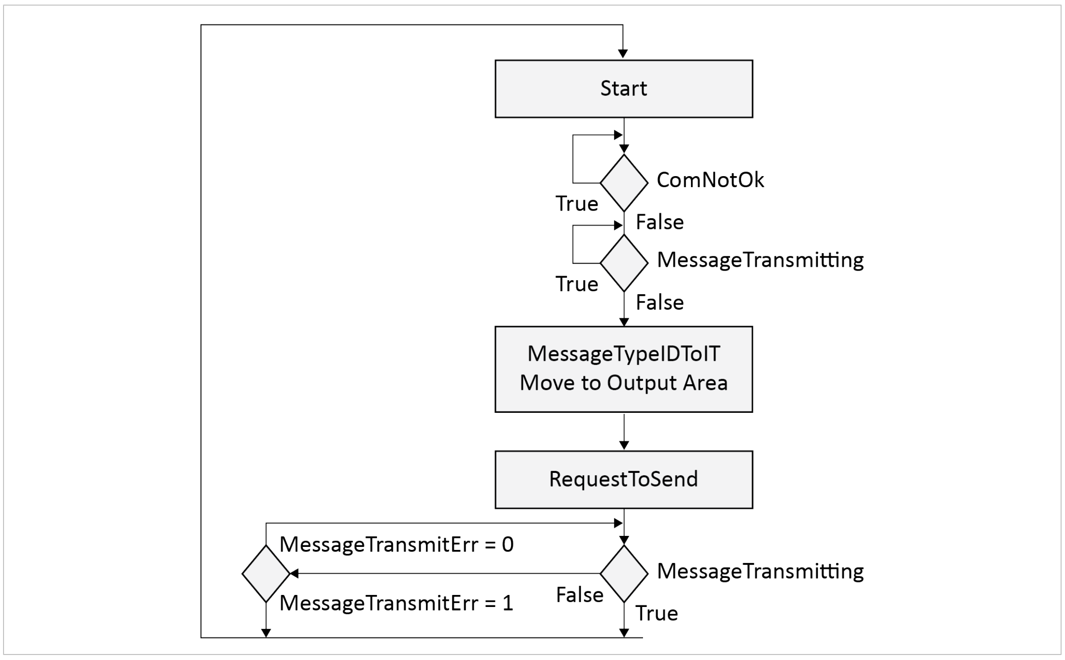


Fig. 27 The PLC application sends a message to the .NET application

Send message to .NET application:

5. Start condition to send message.

   Set the PLC condition to request a message transmission.

6. Check that there is no communication failure between the IT and the OT side by using the function block signal `ComNotOk`.

   Read from output pin on the function block.

7. Check that there is no ongoing transmission by using the function block signal `MessageTransmitting`.

   Read from output pin on the function block.

8. Set the message ID, `MessageTypeIDToIT`, value of the message that you want to send.

9. Move the message that you want to transmit to the message output area.

10. Set the `RequestToSend` signal to the function block.

    Trigger input pin on the function block.

11. Check that there is no ongoing transmission by using the function block signal `MessageTransmitting`.

    Read from output pin on the function block.

    – If `MessageTransmitting` signal is *True* the message was transmitted to the .NET application.

      `MessageTransmitErr` = 0.

    – If `MessageTransmitting` signal is *False* the message transmission to the .NET application failed.

      `MessageTransmitErr` = 1.

6.3 Running the .NET Simulation

6.3.1 PLC Verification

Procedure

1. Download the PLC project to the PLC.
2. Verify that the PLC application is running:
   - In Siemens TIA Portal and Beckhoff TwinCAT 3:

     Check that the `AliveCounterToPlc` pin on the function block is increasing.
   - In Rockwell Studio 5000:

     Check that the `ComNotOk` signal in the add-on instruction is false after the timeout period has passed.

To Do Next

Continue with the simulation, refer to *Post and Receive Messages in .NET Simulator, p. 39*.

6.3.2 Post and Receive Messages in .NET Simulator

Send messages from the simulated .NET application to the PLC application and view the messages received from the PLC application.

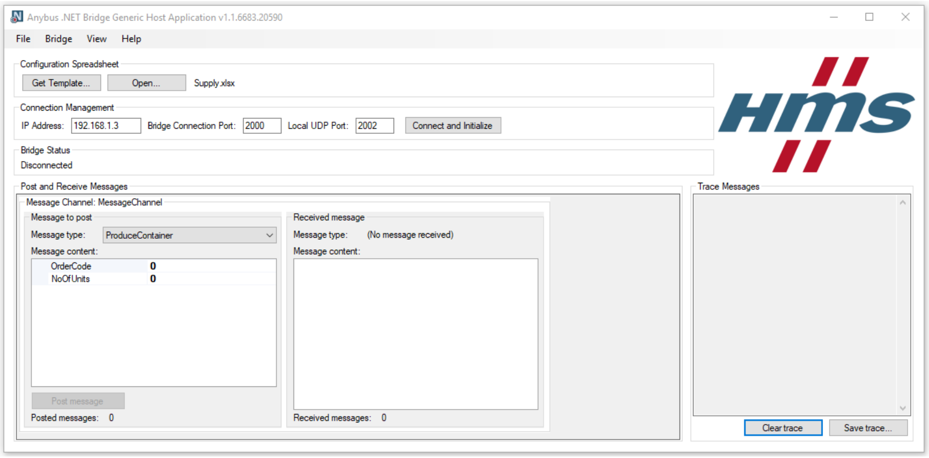
Before You Begin

- In Anybus .NET Bridge Message Mode .NET Simulator, ensure that the Bridge status is *The simulation is enabled*.

Tip: The configuration file, *I/O Mapping Overview* is a useful tool when verifying that a message is sent to the correct pin on the function block.

You can get a quick overview of message lengths, message parameters and I/O data.

For more information, refer to *Configuration Files, p. 20*.



Anybus .NET Bridge Generic Host Application v1.1.6683.20590

Configuration Spreadsheet: Supply.xlsx

Connection Management — IP Address: 192.168.1.3 Bridge Connection Port: 2000 Local UDP Port: 2002

Bridge Status: Disconnected

Post and Receive Messages — Message Channel: MessageChannel

Message to post — Message type: ProduceContainer

Message content:

OrderCode	0
NoOfUnits	0

Posted messages: 0

Received message — Message type: (No message received)

Message content:

Received messages: 0

Fig. 28 Example, .NET Bridge Message Mode .NET Simulator main window

Post Messages from the simulated .NET Application

Procedure

In the Message to post pane you can send a messages from simulated PLC to the .NET application:

1. Select a *Message type*.
2. Add *Message content*.
3. To send the message, click **Post message**.

Result

→ The message is sent from the simulated .NET application to the PLC.

The signal `MessageRecieved` is called.

► Open the PLC environment and verify that the message has been received.

Refer to *Message Structure, p. 37*.

Receive Message in simulated .NET Application

Procedure

When a message is received in the .NET application, the .NET Bridge instance automatically sends an acknowledge (ACK) back to the PLC.

For more information, refer to *Send Message, p. 24*.

Received message pane:

Messages sent from the PLC to the simulated .NET application are shown in the *Received message* pane.

7 Installation

7.1 Product Overview

7.1.1 External Parts

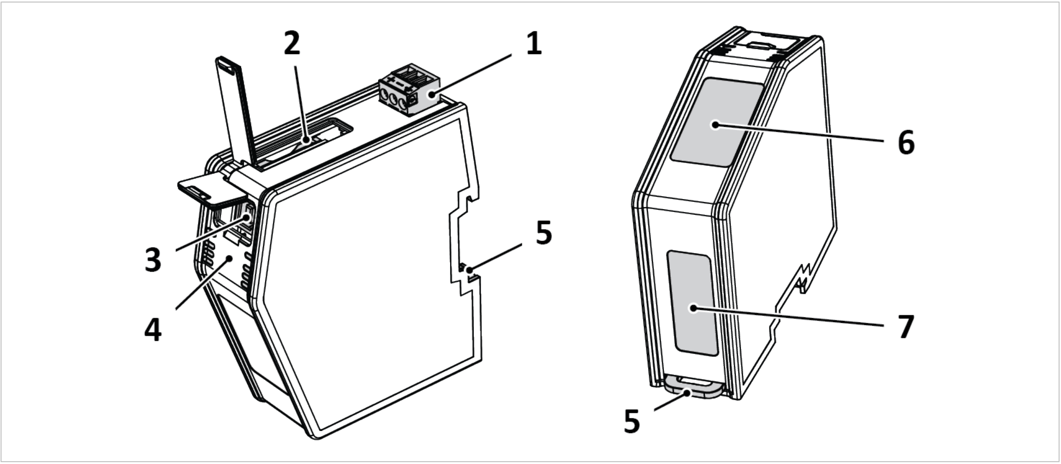


Fig. 29 .NET Bridge external parts

1. **Power Connector**
2. **SD Card Slot** (Currently not in use.)
3. **USB Port**
4. **Status LEDs**
5. **DIN Rail Connector**
6. **Industrial Network Connectors**
7. **IT Network Connectors**

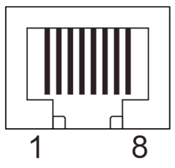
7.1.2 Network Connectors

RJ45 Connector

The RJ45 Connectors located at the front of the .NET Bridge are used to connect the .NET Bridge to an industrial network.

On the .NET Bridge for PROFIBUS there is a D-sub Connector at the front.

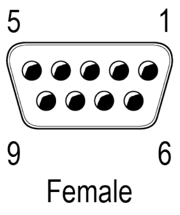
The RJ45 Connectors located at the bottom of the .NET Bridge are used to connect the .NET Bridge to a IT network.

Pin No.	Description	Connector
1	TX+	
2	TX-	
3	RX+	
6	RX-	
4, 5, 7, 8	Not connected	
Housing	Shield	

D-sub Connector

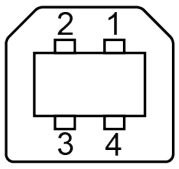
The D-sub Connector is used to connect the .NET Bridge to a PROFIBUS network.

On the .NET Bridge variants for EtherCAT, PROFINET and EtherNet/IP there is an RJ45 Connector at the front.

Pin No.	Description	PROFIBUS Connector DB9F
3	B-line	
4	RTS	
5	GND bus	
6	+5 V bus out	
8	A-line	
1, 2, 7, 9	Not connected	
Housing	PE (Protective Earth)	

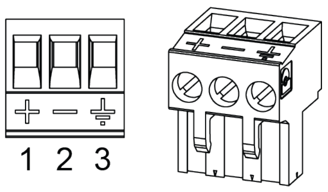
7.1.3 USB Port Type B

The USB Port Type B is used to connect a PC to the .NET Bridge to perform firmware upgrades.

Pin No.	Description	Connector
1	+5 V Input	
2	USBDM (USB communication signals)	
3	USBDP (USB communication signals)	
4	Signal GND	
Housing	Cable shield	

7.1.4 Power Connector

The Power Connector is used to connect the .NET Bridge to power and to Protective Earth (PE).

Pin No.	Description	Connector
1	+24 V DC -15% to +20%	
2	GND	
3	Protective Earth (PE)	

7.2 Mechanical Installation

7.2.1 DIN Rail Mounting Option

The .NET Bridge is designed to be mounted on a DIN rail.

Before You Begin

! The unit must be electrically grounded through the DIN rail for EMC compliance. Make sure that the unit is correctly mounted on the rail and that the rail is properly grounded.

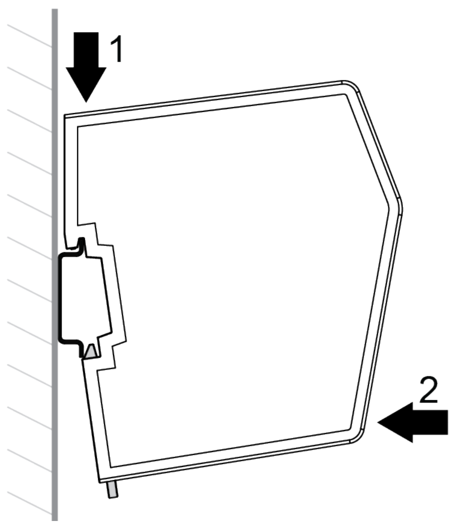
! When the .NET Bridge is used in an environment exposed to vibration, increased stability is required.

Use the *Wall-Mount Accessory Kit* and mount the device on a wall instead of a DIN rail.

The *Wall-Mount Accessory Kit* is ordered separately, please visit www.anybus.com for more information.

Procedure

Mount the .NET Bridge on DIN rail:



1. Hook the .NET Bridge DIN Rail Connector on the DIN rail.
2. Push the .NET Bridge against the DIN rail to make it snap on.

7.2.2 DIN-Rail Demounting

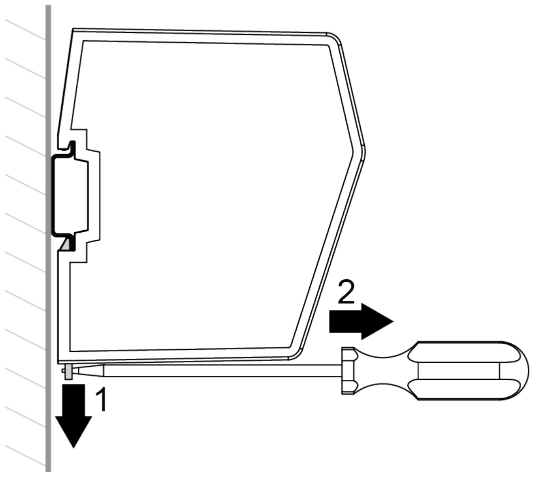
To remove the .NET Bridge from the DIN rail, do the following.

Before You Begin

Have a screwdriver available.

Procedure

Demount the .NET Bridge from the DIN rail:



1. Use the screwdriver to push the DIN Rail Fastening Mechanism down until it locks in the fixed and open position.
2. Unhook the .NET Bridge from the DIN rail.

To Do Next

Do not leave the module with the DIN Rail Fastening Mechanism in the fixed and open position.

This may wear the fastening mechanism out and it cannot be used efficiently.

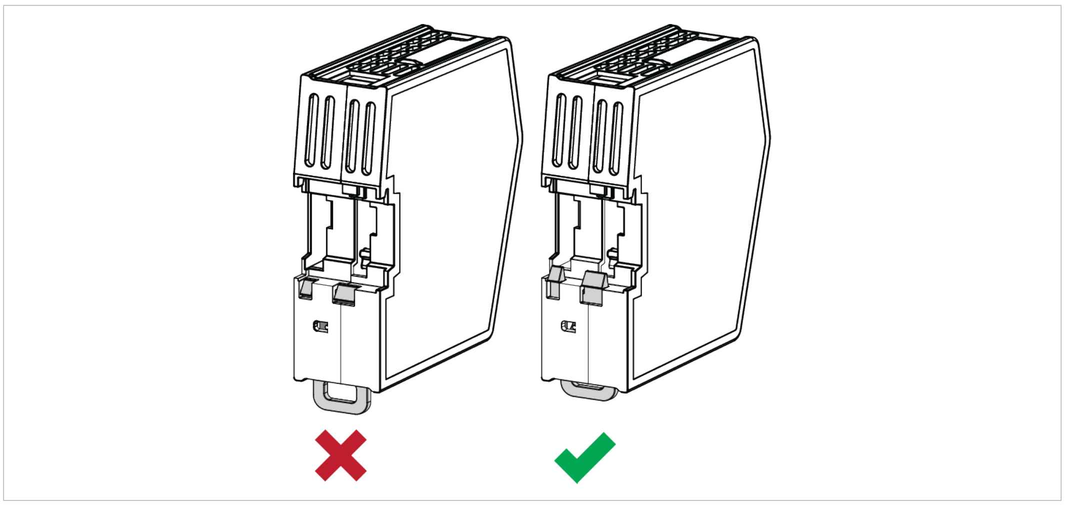


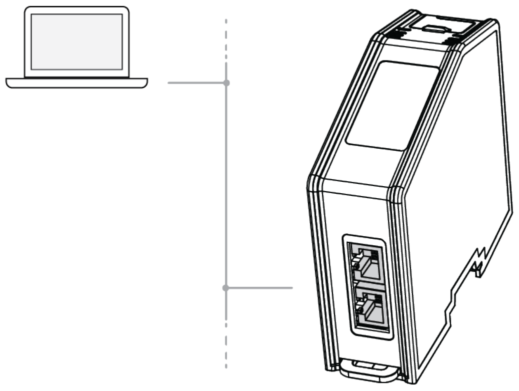
Fig. 30 DIN Rail Fastening Mechanism in open position and closed position

1. After demounting the module, push the DIN Rail Fastening Mechanism into the fixed and closed position.

7.2.3 Connecting to IT Network

Connect the .NET Bridge to the IT network, where the device with the .NET application is installed.

Procedure

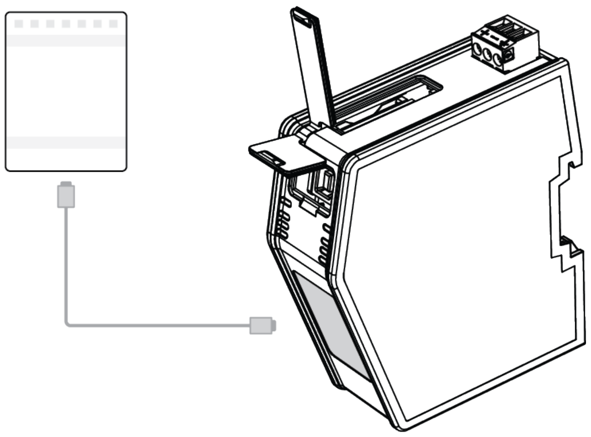


1. Connect the .NET Bridge to the IT network via the dual port switch.

7.2.4 Connecting to Industrial Network

Connect the .NET Bridge to the industrial network, where the PLC is installed.

Procedure



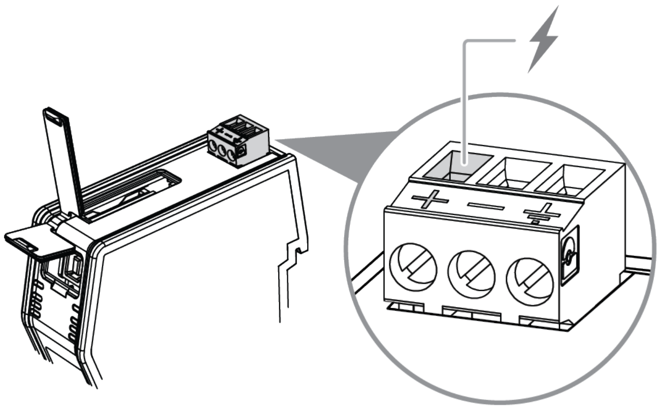
1. Connect the .NET Bridge to the PLC via the network connectors.

7.2.5 Connecting to Power

Connect the .NET Bridge to a power source.

> **!** Connecting power with reverse polarity or using the wrong type of power supply may damage the equipment. Make sure that the power supply is connected correctly and of the recommended type.

Procedure

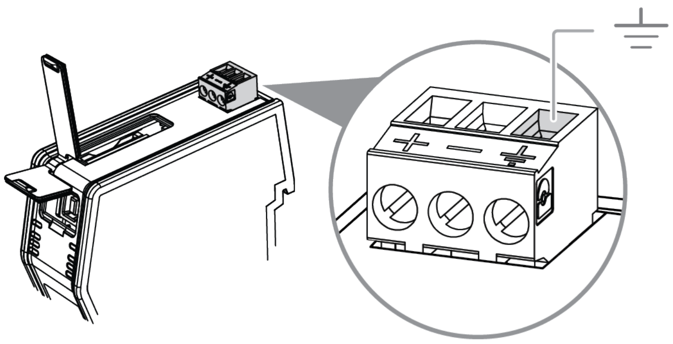


1. Connect the .NET Bridge to a power source via the +24 V DC Power Connector.

7.2.6 Connecting to Protective Earth

> **!** For compliance with EMC standards, the .NET Bridge must be connected to Protective Earth (PE).
>
> When the .NET Bridge is mounted on a wall, connect the device to Protective Earth (PE) via the Protective Earth Connector.
>
> When the .NET Bridge is mounted on a DIN rail, the device is electrically grounded through the DIN rail.

Procedure



1. Connect the .NET Bridge to Protective Earth (PE) via the Protective Earth Connector.

7.3 Network Settings

7.3.1 Configuring the .NET Bridge

The .NET Bridge must be configured before it can be connected to the IT network and the industrial network and before network settings, such as IP addresses, can be changed.

Before You Begin

The .NET Bridge comes with the default IP address *192.168.1.3* for the IT network.

Procedure

Configure the .NET Bridge:

1. Connect the .NET Bridge to the IT network,

   where the PC device with the .NET application, developed for the .NET Bridge, is installed.

2. Start the .NET application.

   → The .NET application connects to the .NET Bridge and configures it.

To Do Next

Continue with the .NET Bridge network settings.

7.3.2 IT Network Settings

The .NET Bridge comes with the default IP address *192.168.1.3* for the IT network.

Assign an IP address to the .NET Bridge, on the IT network where the .NET Bridge is installed.

Use IPconfig to assign the new IP address, refer to *TCP/IP Configuration, p. 55*

7.3.3 Industrial Network Settings

Depending on the industrial network type, assign an IP address or a node address to the .NET Bridge, on the industrial network where the .NET Bridge is installed.

Industrial Network	**Settings**	**Software/Device/Web**
EtherNet/IP	Assign an IP address to the .NET Bridge.	IPconfig
PROFINET	Assign an IP address to the .NET Bridge.	IPconfig **Tip:** You can use the PROFINET controller to assign the IP address.
PROFIBUS	Assign a PROFIBUS node address to the .NET Bridge.	Anybus .NET Bridge Web Interface
EtherCAT	Scan the EtherCAT network to locate the .NET Bridge using the EtherCAT PLC. The .NET Bridge is automatically assigned an node address by the PLC.	EtherCAT PLC

For information about IPconfig, refer to *IT Network Settings, p. 47*.

For information about Anybus .NET Bridge Web Interface, refer to *PROFIBUS Node Address Settings, p. 58*.

7.4 Check Operating Status

When the application development and installation are completed, verify that the .NET Bridge is in operation.

Before You Begin

Ensure that the:

- .NET application is installed on the PC device and is running.
- PLC application is downloaded on the PLC and is running.
- .NET Bridge IP settings are configured for the IT network.
- .NET Bridge IP/Node settings are configured for the industrial network.
- .NET Bridge is connected to the IT network and to the industrial network.

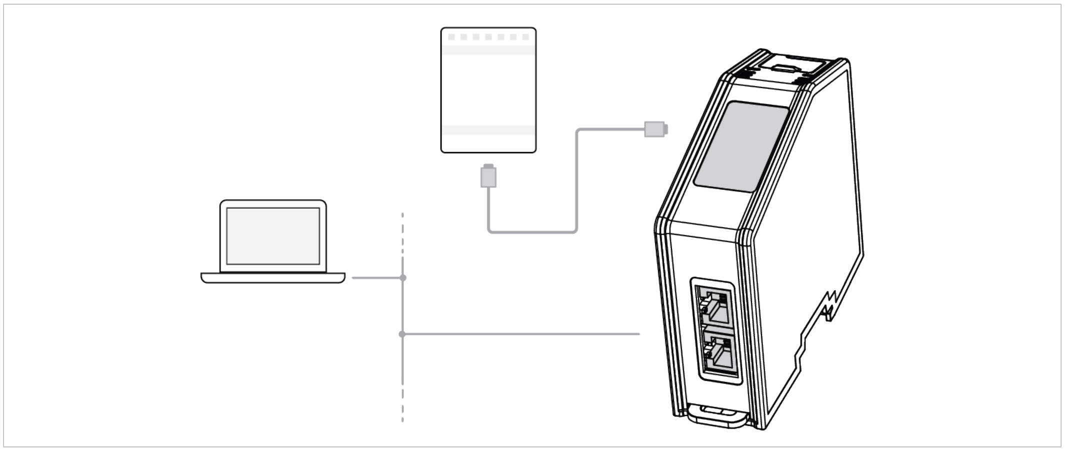


Fig. 31 .NET Bridge connected to IT network and industrial network

- .NET Bridge is connected to power.

Procedure

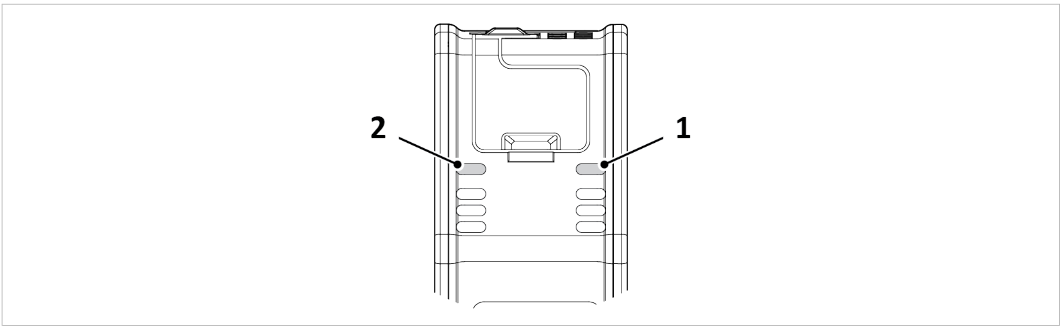


Fig. 32 IT Status LED (1) and OT Status LED (2)

IT network Operating Status

1. When the .NET Bridge is connected to the IT network:
   - → The .NET application connects to the .NET Bridge.
   - → The IT Status LED (1) start blinking slowly green.

Industrial network Operating Status

2. When the .NET Bridge is connected to the industrial network:
   - → The PLC application connects to the .NET Bridge.
   - → The IT Status LED (1) and the OT Status LED (2) turn solid green.

For more information about the LED status indicators, refer to *LED Guide, p. 68*

8 Technical Data

8.1 Technical Specifications

Order code	**AB9077–C — Anybus .NET Bridge PROFINET** **AB9078–C — Anybus .NET Bridge EtherNet/IP** **AB9079–C — Anybus .NET Bridge EtherCAT** **AB9071–C — Anybus .NET Bridge PROFIBUS**
Dimensions	110•35•101 mm, 4,33•1,38•3,98"
Weight	160 g, 0,35 lb
Operating temperature	-25 to +70 °C, -13 to +158 °F
Storage temperature	-40 to +85 °C, -13 to +185 °F
Relative Humidity	5-95% noncondensing
Protection class	IP20, NEMA rating 1
Mounting	DIN rail (35•7,5/15) or Wall mount
Current consumption	Typical: 150 mA @ 24 V DC
Power consumption	24 V DC +/- 10%
Certifications	Refer to datasheet at www.anybus.com/support.

A .NET Bridge Message Settings

A.1 Setting Name and Default Values

The *Setting name* and *Setting values* are defined in the *.NET Bridge Configuration Spreadsheet Template*.

The bridge type name can only be changed in the .NET Bridge Configuration Spreadsheet Template.

If the bridge type name is changed new configuration files must be generated from Anybus .NET Bridge Generator and imported in Visual Studio.

The .NET application can change the setting values before the .NET application is connected to the .NET Bridge.

The Bridge connection TCP port value in the .NET Bridge Configuration Spreadsheet must match the Port configuration in the .NET Bridge Web Interface.

Setting name	Source Code Parameter	Default Setting Value	Available in Template
Bridge type name	The *Bridge type name*, with `Bridge` appended, is used to name the generated .NET type that manages the communication with the .NET Bridge.	N/A	Simplified and Advanced
Bridge timeout	`AliveCounterUpdateTimeout`	12000 ms	Simplified
Bridge connection TCP port	`DefaultDataTransferProtocol`	2000	Advanced
Bridge message TCP port	`DefaultConnectionTcpPort`	2001	Advanced
Bridge data transfer protocol	`DefaultMessageTcpPort`	UDP	Advanced
Bridge data UDP or TCP port	`DefaultDataRemotePort`	2002	Advanced
.NET host data UDP port	`DefaultDataUdpLocalPort`	2002	Advanced
Connection retry interval	`ConnectionRetryInterval`	3000 ms	Advanced
Heartbeat interval	`HeartbeatInterval`	3000 ms	Advanced
Bridge message acknowledgment timeout	`BridgeMessageAcknowledgmentTime-out`	10000 ms	Advanced
.NET to bridge data send interval	`DataSendInterval`	1000 ms	Simplified and Advanced
Bridge to .NET data receive filter divisor	`DataReceiveFilterDivisor`	1 UDP- or TCP-packet per cycle	Advanced
Alive counter update timeout	`AliveCounterUpdateTimeout`	12000 ms	Advanced
Network message acknowledgment timeout	`NetworkMessageAcknowledgmentTime-out`	12000 ms	Advanced

A.2 Setting Name Description

A.2.1 Bridge type name

Name the .NET Bridge.

Enter Bridge type name as a single word, do not use whitespace.

A.2.2 Bridge timeout

Use short Bridge timeout to reduce time to:

- detect a communication error
- time to establish a new connection after the communication error is corrected

Using shorter timeouts will increase the network traffic load.

A.2.3 Bridge connection TCP port

The TCP port the .NET Bridge is configured to listen on for incoming connections.

A.2.4 Bridge message TCP port

The TCP port the .NET Bridge is configured to listen on for incoming bridge messages.

A.2.5 Bridge data transfer protocol

Choose UDP or TCP.

Use UDP when a fast communication rate is required, when messages will be sent often. Example: For process data exchange.

Use TCP if the communication rate is low. TCP generates less data traffic between the .NET Bridge and the .NET application.

A.2.6 Bridge data UDP or TCP port

The setting must match the Bridge data transfer protocol setting, UDP or TCP.

The UDP or TCP port the .NET Bridge is configured to listen on for messages sent from the .NET application to the PLC.

A.2.7 .NET host data UDP port

The UDP port which the .NET host will listen on for messages sent from the PLC to the .NET host.

A.2.8 Connection retry interval

The Connection retry interval is the interval between the .NET application trying to connect to the .NET Bridge, after the previous connection was lost or could not be established.

The .NET application will continue the attempts to connect until the instance is stopped or removed.

A.2.9 Heartbeat interval

The Heartbeat interval value determines how often the .NET application sends a heartbeat message to the .NET Bridge.

It is the time between two consecutive heartbeat message transmissions from the .NET application to the .NET Bridge.

If the .NET Bridge do not receive a heartbeat message from the .NET application during this interval, the .NET Bridge will consider the connection to the .NET application lost.

A.2.10 Bridge message acknowledgment timeout

The Bridge message acknowledgment timeout is the maximum time between a message is sent until an acknowledgement from the receiver is received.

This applies to message sent from:

- .NET application to .NET Bridge
- .NET Bridge to .NET application

If no acknowledgement is received, the .NET Bridge status can change to disconnected.

A.2.11 .NET to bridge data send interval

Data throughput = DataSendInterval multiplied by 2.

Maximum DataSendInterval value is 1000 ms.

Value Set in	Minimum DataSendInterval Value	New data will be exchanged at most
Configuration Spreadsheet Template	100 ms	every 2*100 ms
.NET application	20 ms	every 2*20 ms

A.2.12 Bridge to .NET data receive filter divisor

The Ethernet traffic load between .NET Bridge and .NET application can be limited.

The Data Receive Filter Divisor setting determine how often the .NET Bridge forwards data from the PLC to the .NET application.

Depending on the Bridge data transfer protocol setting, either UDP or TCP is used.

UDP is the default setting.

By default frames are sent each industrial network process data cycle, or equivalent for industrial networks without a fixed cycle length.

By changing the default settings the Ethernet traffic load between the .NET Bridge and the .NET application can be limited.

UDP or TCP frames will not be sent each industrial network process data cycle or equivalent for industrial networks without a fixed cycle length.

Filter settings for when data is forwarded	Description
0	Each time the received I/O data has changed. Do not use this filter setting when the bridge data transfer protocol UDP is used.
1	Each time I/O data is received from the industrial network.
2–255	Each DataReceiveFilterDivisor time the data is received from the industrial network.

Some industrial networks do not have a fixed network cycle. For those networks this setting increases or decreases the frequency the .NET Bridge sends I/O data to the .NET application. There is no relation to a network cycle length.

A.2.13 Alive counter update timeout

Alive Counter Update Timeout is the maximum time before the sender should have received a mirrored Alive Counter value back from the receiver.

It is recommended to use the value of the Alive counter update timeout as the PLC system timeout setting, if the industrial network should detect loss of connection to the .NET side.

This applies to Alive Counter values sent from:

- .NET application to PLC
- PLC to .NET application

The .NET application sends an `AliveCounter` value to the PLC. When the PLC receive the `AliveCounter` value it returns the same value back to the .NET application.

The .NET application increases the returned value and sends it to the PLC. Then the PLC returns the new value, and so on.

Loss of connection can be considered detected if:

- The .NET application does not receive the same value as the value sent to the PLC, before timeout occur.
- The PLC does not receive a different value than the value sent to the .NET application, before a timeout occur.

A.2.14 Network message acknowledgment timeout

Network Message Acknowledgment Timeout is the maximum time from sending a network message until an acknowledgement should have been received.

The reset message acknowledgment is also monitored by this timeout.

Applies to messages from .NET application to the PLC and from the PLC to .NET application.

If no acknowledgement is received, the .NET Bridge status may change to disconnected.

B TCP/IP Configuration

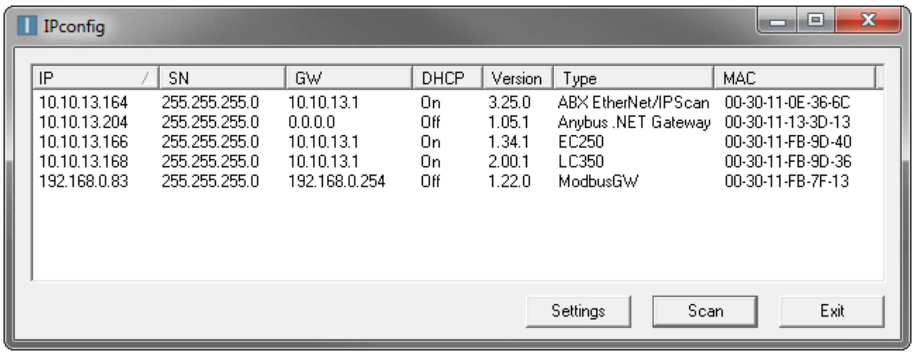
B.1 Installing the IPconfig Utility

IPconfig is a Windows-based tool for configuration of TCP/IP settings in HMS devices. The tool will detect all compatible and active HMS devices on the local network.

1. Download IPconfig from www.anybus.com/support.
2. Unpack the contents of the zip archive and run the installer program.

B.2 Scanning for Connected Devices

When IPconfig is started it will automatically scan all available local networks for HMS devices. Detected devices will be listed in the main window. To refresh the list, click on **Scan**.



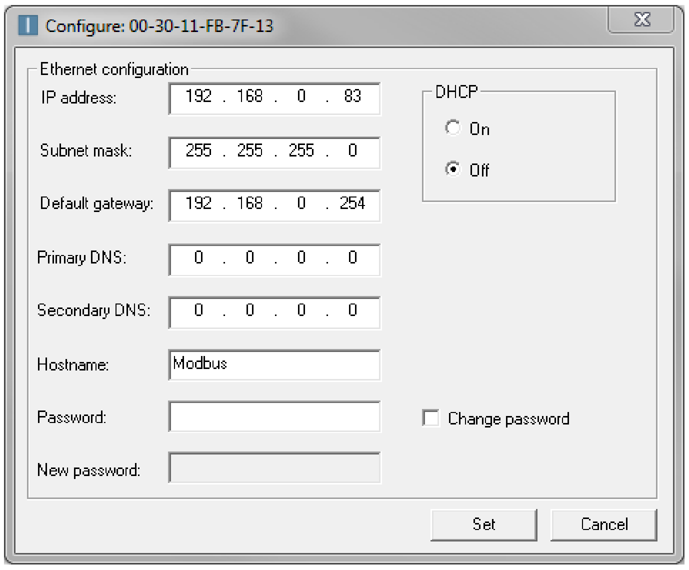
IP	SN	GW	DHCP	Version	Type	MAC
10.10.13.164	255.255.255.0	10.10.13.1	On	3.25.0	ABX EtherNet/IPScan	00-30-11-0E-36-6C
10.10.13.204	255.255.255.0	0.0.0.0	Off	1.05.1	Anybus .NET Gateway	00-30-11-13-3D-13
10.10.13.166	255.255.255.0	10.10.13.1	On	1.34.1	EC250	00-30-11-FB-9D-40
10.10.13.168	255.255.255.0	10.10.13.1	On	2.00.1	LC350	00-30-11-FB-9D-36
192.168.0.83	255.255.255.0	192.168.0.254	Off	1.22.0	ModbusGW	00-30-11-FB-7F-13

Fig. 33 IPconfig main window

IP	IP address of the device
SN	Subnet mask
GW	Default gateway
DHCP	Automatically managed IP configuration
Version	Firmware version
Type	Product name
MAC	Ethernet MAC address (System ID)

B.3 Ethernet Configuration

To change the IP settings for a device, double-click on the entry in the main window or right-click on it and select **Configuration**.


Configure: 00-30-11-FB-7F-13

Ethernet configuration

IP address:	192 . 168 . 0 . 83
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	192 . 168 . 0 . 254
Primary DNS:	0 . 0 . 0 . 0
Secondary DNS:	0 . 0 . 0 . 0
Hostname:	Modbus
Password:	
New password:	

DHCP: On [✓ Off]

☐ Change password

Fig. 34 Ethernet configuration

Enter static IP settings as required, or select DHCP if using dynamic IP addressing.

! Do not enable DHCP if there is no DHCP server available on the network.

You can add a name for the device in the **Hostname** field. Only characters a–z, A–Z, 0–9 and _ (underscore) are allowed.

The default password for changing IP settings is blank (no password). If a password has been set for the device you must enter it to be able to change the settings.

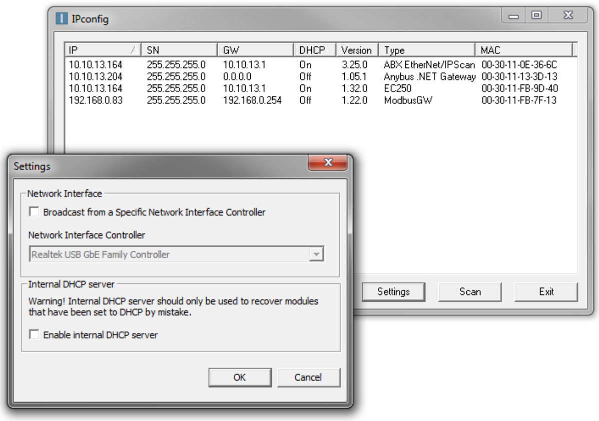
To set a new password, check the **Change password** box and enter the current password in the **Password** field, then enter the new password in the **New password** field.

! For security reasons the default password should always be changed.

Click on **Set** to save the new settings. The device will reboot automatically.

B.4 IPconfig Settings

Additional settings for IPconfig can be accessed by clicking on **Settings**.



IP	SN	GW	DHCP	Version	Type	MAC
10.10.13.164	255.255.255.0	10.10.13.1	On	3.25.0	ABX EtherNet/IPScan	00-30-11-0E-36-6C
10.10.13.204	255.255.255.0	0.0.0.0	Off	1.05.1	Anybus .NET Gateway	00-30-11-13-3D-13
10.10.13.164	255.255.255.0	10.10.13.1	On	1.32.0	EC250	00-30-11-FB-9D-40
192.168.0.83	255.255.255.0	192.168.0.254	Off	1.22.0	ModbusGW	00-30-11-FB-7F-13

Settings

Network Interface

☐ Broadcast from a Specific Network Interface Controller

Network Interface Controller: Realtek USB GbE Family Controller

Internal DHCP server

Warning! Internal DHCP server should only be used to recover modules that have been set to DHCP by mistake.

☐ Enable internal DHCP server

OK Cancel

Fig. 35 IPconfig settings

Network Interface

Check this option to select a specific network interface to use when scanning for devices from a computer which has more than one interface. If this option is left unchecked, all available networks will be scanned.

Internal DHCP Server

If a device has been set to use DHCP but there is no DHCP server on the network, the device may not be detected by IPconfig. To recover access to the device an internal DHCP server in IPconfig can be temporarily activated:

1. Click the checkbox for **Internal DHCP Server**, then click **OK**. IPconfig will automatically refresh the scan and list the missing device in the main window.
2. Select the device and configure it to use static IP addressing instead of DHCP.
3. Disable the internal DHCP server.

> **!** Do not enable the internal DHCP server if there is already an active DHCP server on the network.

C PROFIBUS Node Address Settings

Use the Anybus .NET Bridge Web Interface to assign a PROFIBUS node address for the .NET Bridge, on the PROFIBUS network where the .NET Bridge is installed.

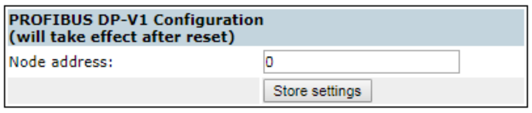
Before You Begin

Ensure that the .NET Bridge is connected to the same network as the device that is running the web browser.

Procedure

Assign a PROFIBUS node address:

1. In your web browser, type the .NET Bridge IP address and enter the Anybus .NET Bridge Web Interface start page.



PROFIBUS DP-V1 Configuration
(will take effect after reset)

Node address:	0

Fig. 36 PROFIBUS DP-V1 Configuration pane

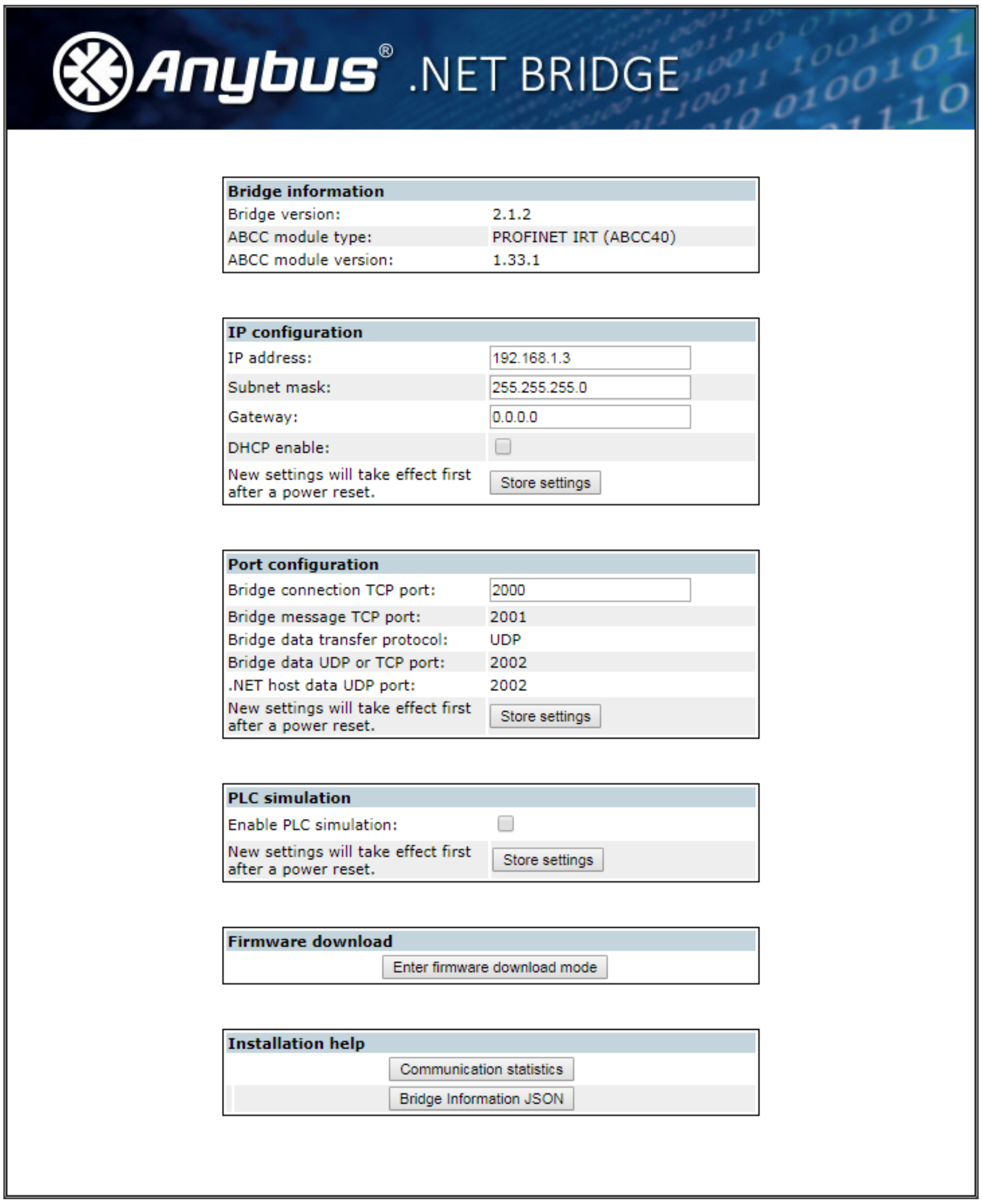
2. In the PROFIBUS DP-V1 Configuration pane, enter the desired PROFIBUS node address.
3. Click **Store settings**.
4. Restart the .NET Bridge.

Result

→ The change take effect after restart.

D .NET Bridge Web Interface

The web interface for the .NET bridge is available via IP address 192.168.1.3, default setting.



Bridge information	
Bridge version:	2.1.2
ABCC module type:	PROFINET IRT (ABCC40)
ABCC module version:	1.33.1

IP configuration	
IP address:	192.168.1.3
Subnet mask:	255.255.255.0
Gateway:	0.0.0.0
DHCP enable:	☐
New settings will take effect first after a power reset.	Store settings

Port configuration	
Bridge connection TCP port:	2000
Bridge message TCP port:	2001
Bridge data transfer protocol:	UDP
Bridge data UDP or TCP port:	2002
.NET host data UDP port:	2002
New settings will take effect first after a power reset.	Store settings

PLC simulation	
Enable PLC simulation:	☐
New settings will take effect first after a power reset.	Store settings

Fig. 37 The Anybus .NET Bridge web interface

The Anybus .NET Bridge Web Interface Functions

Pane	Description
Bridge Information	Show the .NET Bridge version and other information vital for support.
IP configuration	Show current TCP/IP settings. IP configuration is editable.
Port configuration	Current port settings .NET Bridge connection TCP port is editable. .NET Bridge message TCP port and .NET Bridge data TCP/UDP port are configured via the .NET application.
PLC simulation	Checking the **Enable PLC simulation** checkbox enables the .NET Bridge to enter PLC simulation mode. The .NET Bridge must be reset before the changes take effect.
Network specific configuration	Available for certain networks. On PROFIBUS, this is where you set the node address. The .NET Bridge must be reset before the changes take effect
Firmware download	Pressing the **Enter firmware download mode** button makes the .NET Bridge ready for firmware download. The .NET Bridge must be connected via the USB interface. The .NET Bridge can not be accessed any other way, until it has been restarted. Firmware is downloaded using the Firmware Manager II from HMS.
Installation help	The **Communication statistics** button show detailed information about packets and communication.
	The **Bridge information JSON** button show information about the .NET Bridge embedded in a JSON script

The .NET Bridge TCP/IP settings can also be configured in IPconfig. Refer to TCP/IP Configuration, p. 55.

E General Function Block Guide

E.1 General Function Block Message Structure

This is the message structure for a general function block for single channel bridges.

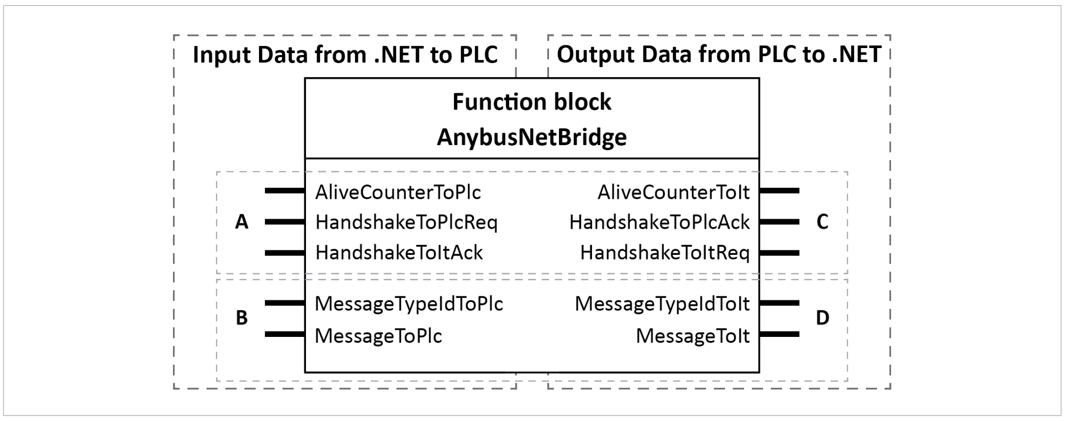


Fig. 38 AnybusNetBridge Function block Input and Output Data

Input Data from .NET application to PLC

	Byte	Length (bytes)	Name	Description
A	1	1	`AliveCounterToPlc`	Value from 1-99. Used to detect if there is an active communication partner. Once the PLC has detected a new value here, a timer can be started to detect communication failure with the IT system. The timeout value is configured in the *.NET Bridge Configuration Spreadsheet*.
	2	1	`HandshakeToPlcReq`	0: No action 1: New message The `MessageData` part contains a new message from the IT system side.
	3	1	`HandshakeToItAck`	0: No action 2: ACK The previous message sent from the PLC was received by the IT system side.
B	4	2	`MessageTypeIdToPlc`	Used to determine which message has been sent.
	6 –n	1-x	`MessageToPlc`	Data to the PLC side.

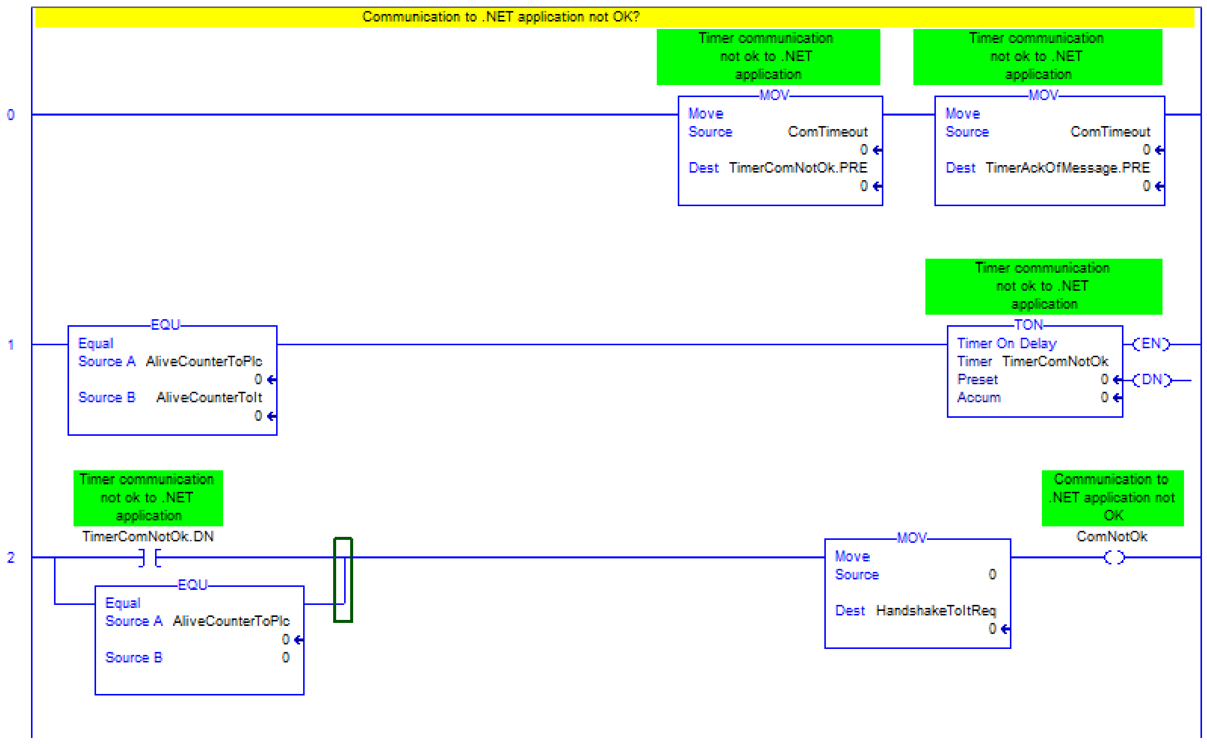
Output Data from PLC to .NET application

	Byte	Length (bytes)	Name	Description
C	1	1	`AliveCounterToIt`	Used to detect if there is an active communication partner. The PLC shall mirror the incoming `AliveCounterToPlc` value
	2	1	`HandshakeToPlcAck`	0: No action 2: ACK The previous message sent from the IT system was received by the PLC.
	3	1	`HandshakeToItReq`	0: No action 1: New message The `MessageData` part contains a new message from the PLC.
D	4	2	`MessageTypeIdToIt`	Used to determine which message has been sent.
	6 –n	1-x	`MessageToIt`	Data to the IT system side.

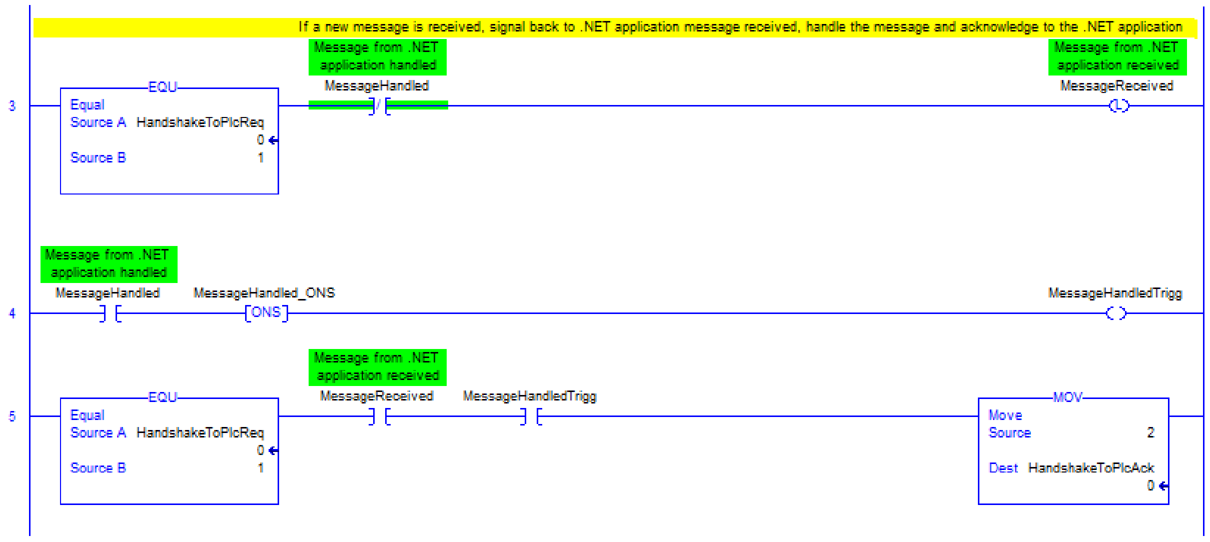
E.2 General Function Block Layout

The code example presented below can be used as a guide to develop function blocks for any type of PLC.

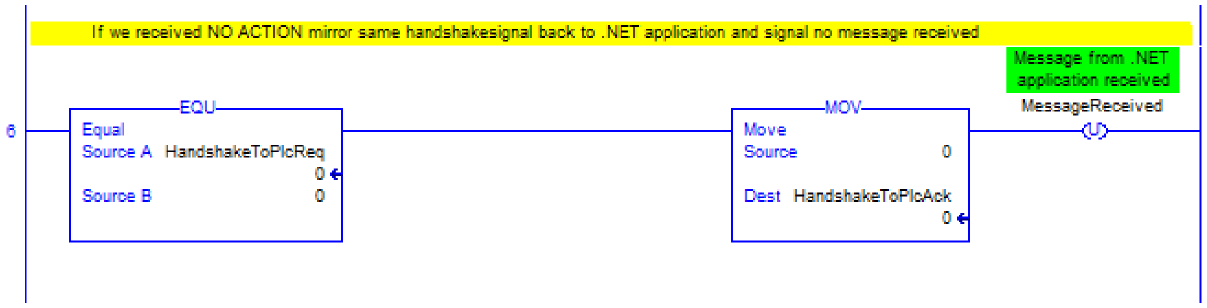
Communication to .NET application not Ok?


Communication to .NET application not OK?
Timer communication not ok to .NET application
Timer communication not ok to .NET application
0
MOV
Move
Source ComTimeout
0
Dest TimerComNotOk.PRE
0
MOV
Move
Source ComTimeout
0
Dest TimerAckOfMessage.PRE
0
Timer communication not ok to .NET application
1
EQU
Equal
Source A AliveCounterToPlc
0
Source B AliveCounterToIt
0
TON
Timer On Delay
Timer TimerComNotOk
Preset 0
Accum 0
EN
DN
Timer communication not ok to .NET application
TimerComNotOk.DN
Communication to .NET application not OK
ComNotOk
2
EQU
Equal
Source A AliveCounterToPlc
0
Source B 0
MOV
Move
Source 0
Dest HandshakeToItReq
0

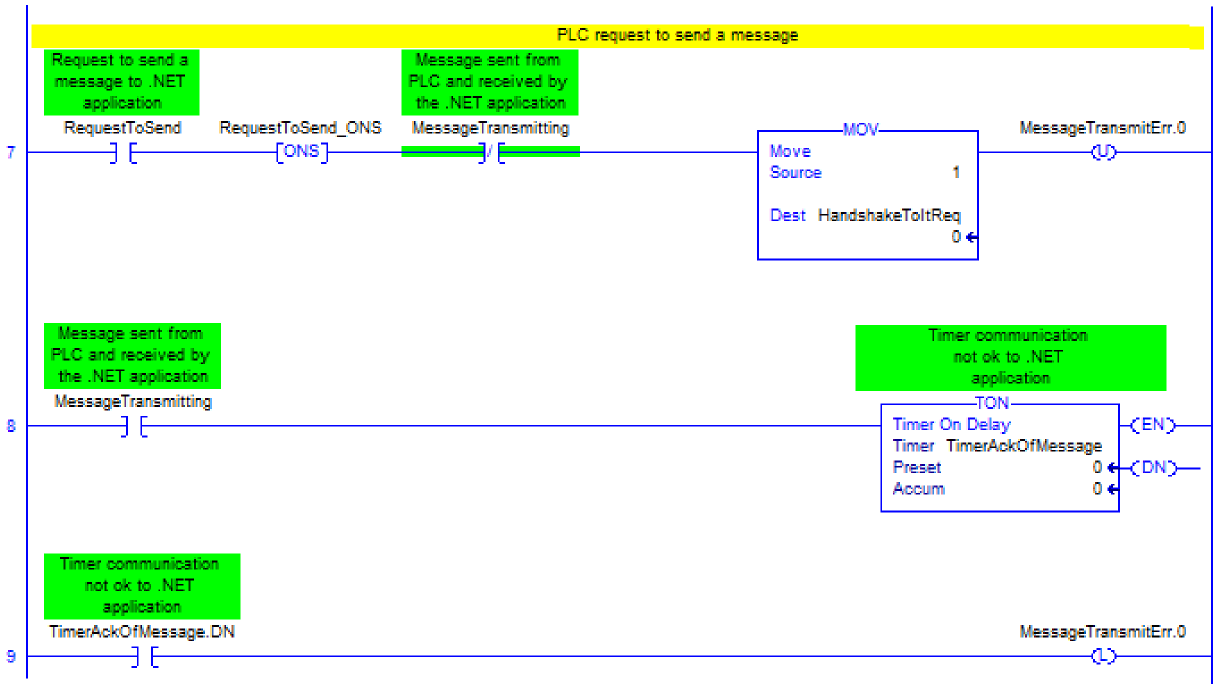
New Message Received


If a new message is received, signal back to .NET application message received, handle the message and acknowledge to the .NET application
Message from .NET application handled
MessageHandled
Message from .NET application received
MessageReceived
3
EQU
Equal
Source A HandshakeToPlcReq
0
Source B 1
L
Message from .NET application handled
MessageHandled
MessageHandled_ONS
ONS
MessageHandledTrigg
4
Message from .NET application received
MessageReceived
MessageHandledTrigg
5
EQU
Equal
Source A HandshakeToPlcReq
0
Source B 1
MOV
Move
Source 2
Dest HandshakeToPlcAck
0

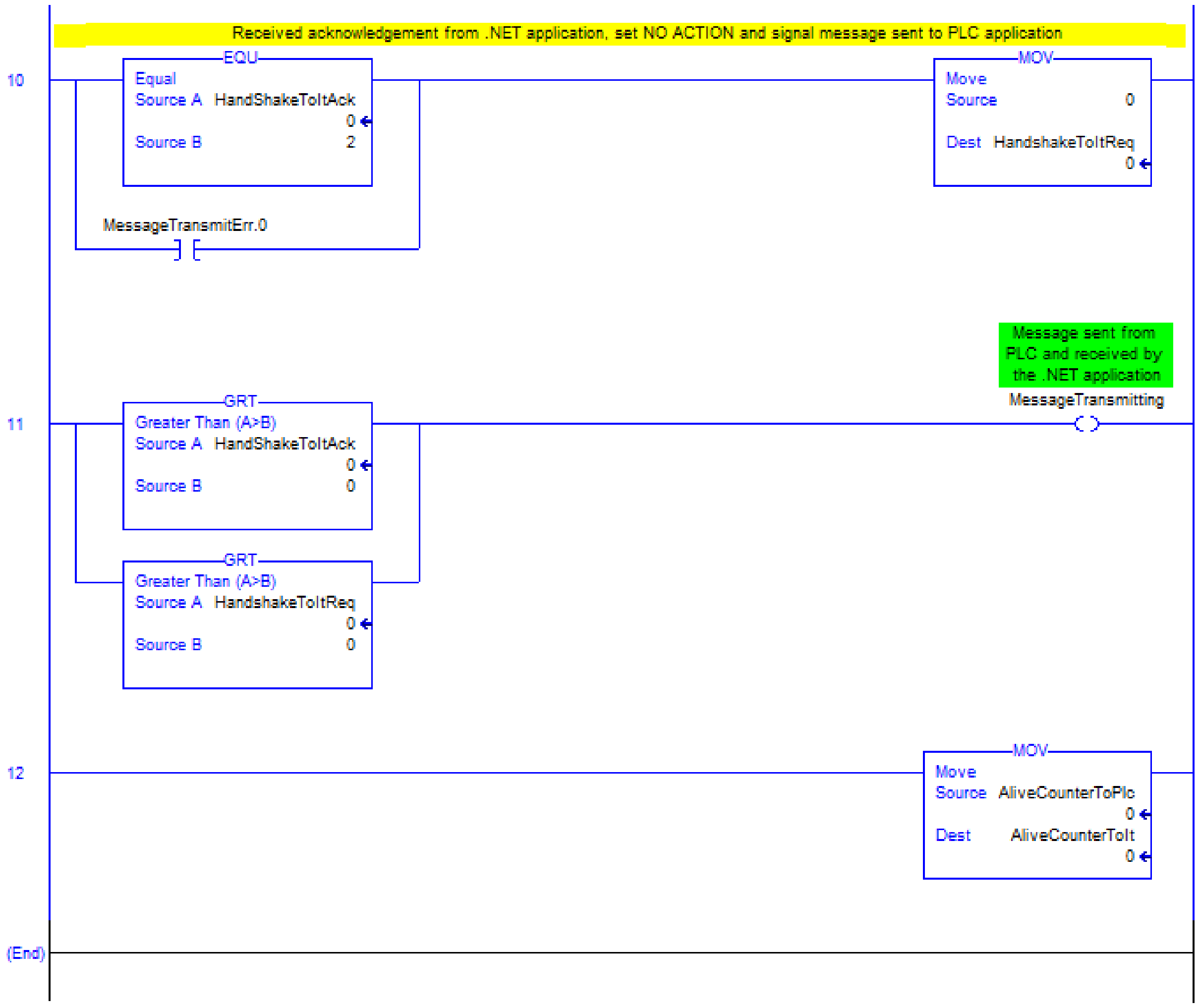
Send Handshake


If we received NO ACTION mirror same handshakesignal back to .NET application and signal no message received
Message from .NET application received
MessageReceived
6
EQU
Equal
Source A HandshakeToPlcReq
0
Source B 0
MOV
Move
Source 0
Dest HandshakeToPlcAck
0
U

PLC Request to send a message


PLC request to send a message
Request to send a message to .NET application
RequestToSend
RequestToSend_ONS
ONS
Message sent from PLC and received by the .NET application
MessageTransmitting
7
MOV
Move
Source 1
Dest HandshakeToItReq
0
MessageTransmitErr.0
U
Message sent from PLC and received by the .NET application
MessageTransmitting
8
Timer communication not ok to .NET application
TON
Timer On Delay
Timer TimerAckOfMessage
Preset 0
Accum 0
EN
DN
Timer communication not ok to .NET application
TimerAckOfMessage.DN
9
MessageTransmitErr.0
L

Acknowledgement (ACK) from .NET application


Received acknowledgement from .NET application, set NO ACTION and signal message sent to PLC application
10
EQU
Equal
Source A HandShakeToItAck
0
Source B 2
MOV
Move
Source 0
Dest HandshakeToItReq
0
MessageTransmitErr.0
Message sent from PLC and received by the .NET application
MessageTransmitting
11
GRT
Greater Than (A>B)
Source A HandShakeToItAck
0
Source B 0
GRT
Greater Than (A>B)
Source A HandshakeToItReq
0
Source B 0
12
MOV
Move
Source AliveCounterToPlc
0
Dest AliveCounterToIt
0
(End)

F Network Specific Function Blocks

F.1 TIA Portal Function Block Technical Overview

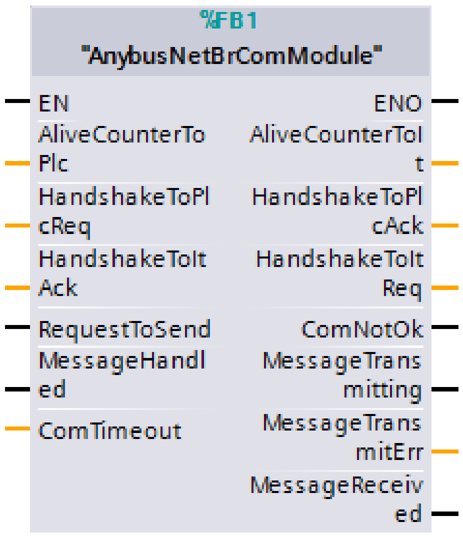


Fig. 39 AnybusNetBridge function block

Parameter	Declaration	Data type	Description
`EN`	Input	BOOL	Enable input
`ENO`	Output	BOOL	Enable output
`AliveCounter-ToPlc`	Input	BYTE	Connect to `AliveCounterToPlc` of input area
`AliveCounterToIt`	Output	BYTE	Connect to `AliveCounterToIt` of output area
`HandshakeToPl-cReq`	Input	BYTE	Connect to `HandshakeToPlcReq` of the input area
`HandshakeToPl-cAck`	Output	BYTE	Connect to `HandshakeToPlcAck` of the output area
`HandshakeToItAck`	Input	BYTE	Connect to `HandshakeToItAck` of the input area
`HandshakeToItReq`	Output	BYTE	Connect to `HandshakeToItReq` of the output area
`RequestToSend`	Input	BOOL	Set when a new message is ready to be sent
`ComNotOK`	Output	BOOL	Communication error
`MessageHandled`	Input	BOOL	Acknowledgement that the incoming message has been handled and copied to the memory area
`MessageTransmit-ting`	Output	BOOL	Ongoing message transmission
`ComTimeout`	Input	TIME	Time to wait for the IT side to respond (shall be same as in Excel sheet, default 10 s)
`MessageTransmi-tErr`	Output	INT	No response to last transmitted message (value = 1)
`MessageReceived`	Output	BOOL	Will be set when a new message is received. Will be cleared in the same cycle as `MessageHandled` is set.

F.2 Beckhoff TwinCAT 3 Function Block Technical Overview

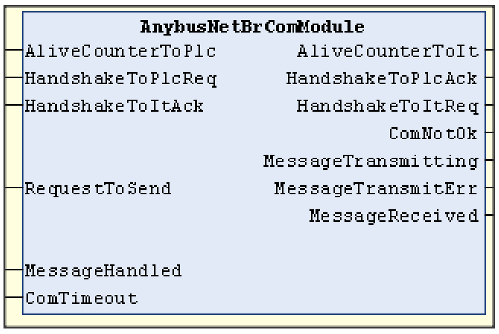
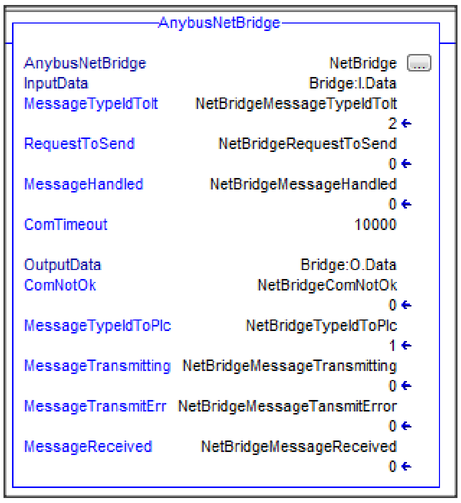


Fig. 40 AnybusNetBridge function block

Parameter	Declaration	Data type	Description
`AliveCounter-ToPlc`	Input	BYTE	Connect to `AliveCounterToPlc` of input area
`AliveCounterToIt`	Output	BYTE	Connect to `AliveCounterToIt` of output area
`HandshakeToPl-cReq`	Input	BYTE	Connect to `HandshakeToPlcReq` of the input area
`HandshakeToPl-cAck`	Output	BYTE	Connect to `HandshakeToPlcAck` of the output area
`HandshakeToItAck`	Input	BYTE	Connect to `HandshakeToItAck` of the input area
`HandshakeToItReq`	Output	BYTE	Connect to `HandshakeToItReq` of the output area
`RequestToSend`	Input	BOOL	Set when a new message is ready to be sent
`ComNotOK`	Output	BOOL	Communication error
`MessageHandled`	Input	BOOL	Acknowledgement that the incoming message has been handled and copied to the memory area
`MessageTransmit-ting`	Output	BOOL	Ongoing message transmission
`ComTimeout`	Input	TIME	Time to wait for the IT side to respond (shall be same as in Excel sheet, default 10 s)
`MessageTransmi-tErr`	Output	INT	No response to last transmitted message (value = 1)
`MessageReceived`	Output	BOOL	Will be set when a new message is received. Will be cleared in the same cycle as `MessageHandled` is set

F.3 Rockwell Studio 5000 Function Block Technical Overview



```
AnybusNetBridge
AnybusNetBridge                         NetBridge
InputData                           Bridge:I.Data
MessageTypeIdToIt    NetBridgeMessageTypeIdToIt
                                                2 ←
RequestToSend           NetBridgeRequestToSend
                                                0 ←
MessageHandled         NetBridgeMessageHandled
                                                0 ←
ComTimeout                                  10000

OutputData                          Bridge:O.Data
ComNotOk                         NetBridgeComNotOk
                                                0 ←
MessageTypeIdToPlc        NetBridgeTypeIdToPlc
                                                1 ←
MessageTransmitting NetBridgeMessageTransmitting
                                                0 ←
MessageTransmitErr NetBridgeMessageTansmitError
                                                0 ←
MessageReceived       NetBridgeMessageReceived
                                                0 ←
```

Fig. 41 AnybusNetBridge function block

Parameter	Declaration	Data type	Description
`InputData`	Input	-	Input area for the .NET Bridge, defined Name:I.Data
`MessageTypeId-ToIt`	Input	INT	Message id of the message that will be sent to the .NET application
`RequestToSend`	Input	BOOL	Set when a new message is ready to be sent
`MessageHandled`	Input	BOOL	Acknowledgement that the incoming message has been handled and copied to the memory area
`ComTimeout`	Input	DINT	Time to wait for the IT side to respond (shall be same as in Excel sheet, default 10 s)
`OutputData`	Output	-	Output area for the .NET Bridge, defined Name:O.Data
`MessageTypeId-ToPlc`	Output	INT	Message id of the received message from the .NET application
`MessageTransmit-ting`	Output	BOOL	Ongoing message transmission
`MessageTransmi-tErr`	Output	INT	No response to last transmitted message (value = 1)
`MessageReceived`	Output	BOOL	Will be set when a new message is received. Will be cleared in the same cycle as `MessageHandled` is set.

G LED Guide

G.1 LED Description

The .NET Bridge has eight LED status indicators located at the front.

The following tables describes what the different status indications mean.

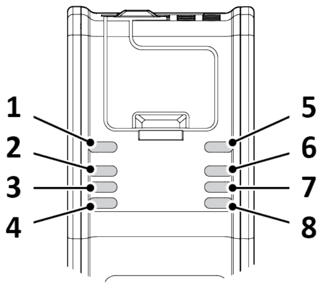


Fig. 42 .NET Bridge LED Status Indicators

General LED Status Indicators:

LED	Name	Indication	Description
1	OT OT Status	Off	Power off
		Green	Connection to PLC
2	SIM PLC Simulation	Green Green, blinking Off	Simulation started Simulation enabled Simulation mode is off
3, 4	Network specific	-	Refer to information about network specific LED status indicators.
5	IT IT status	Off	Disconnected
		Green, fast blink	Connection to IT network
		Green, slow blink	Pre-operational. Connected to IT and OT network. No I/O data exchange.
		Green	Normal operation. I/O data exchange
		Red, blinking	If this LED and the Power LED both are red, this indicates a fatal error
6	PWR Power	Green	Power on
		Red	If this LED and the IT LED both are red, this indicates a fatal error
7, 8	LA1, LA2 Ethernet Link 1 and 2	Off	No link
		Flashing green	Receiving/transmitting Ethernet packets at 100 Mbit
		Flashing yellow	Activity, receiving/transmitting Ethernet packets at 10 Mbit
		Yellow	Boot up

Status LED indicator (3) and (4) indicates different activities depending on industrial networks.

EtherNet/IP specific LED Status Indicators:

LED	Name	Indication	Description
3	MS Module Status	Off	No power
		Green	Controlled by a scanner in Run state
		Flashing green	Not configured, or scanner in Idle state
		Red	Major fault
		Flashing red	Recoverable fault(s). Module is configured, but stored parameters differ from currently used parameters
4	NS Network Status	Off	No power or no IP address
		Green	Online, connection established
		Flashing green	Online, no connection established
		Red	Duplicate IP address, FATAL error on the OT network interface
		Flashing red	Connection timed out

EtherCAT specific LED Status Indicators:

LED	Name	Indication	Description
3	RUN	Off	INIT
		Green	OPERATIONAL
		Blinking green	PRE-OPERATIONAL
		Green, one flash	SAFE-OPERATIONAL
		Flickering	BOOT
		Red	If RUN and ERR both turn red, this indicates a fatal event. Contact HMS support
4	ERR	Off	No error
		Blinking red	Invalid configuration
		Red	Application controller failure. If RUN and ERR both turn red, this indicates a fatal event. Contact HMS support
		Red, one flash	Unsolicited state change
		Red, two flashes	Sync manager watchdog timeout
		Flickering	Booting error detected

PROFIBUS specific LED Status Indicators:

LED	Name	Indication	Description
3	OP Network Status	Off	Not online
		Green	Online, data exchange
		Green, flashing	Online, clear
		Red, one flash	Parametrization error
		Red, two flashes	PROFIBUS configuration error
4	ST Module Status	Off	Not initialized
		Green	Initialized
		Green, flashing	Diagnostic event
		Red	Fatal error

PROFINET specific LED Status Indicators:

LED	Name	Indication	Description
3	NS Network Status	Off Green Green, one flash Green, three flashes Red Red, one flash Red, two flashes Red, three flashes	No connection Online (RUN): Connection established, IO controller in RUN state Online (STOP): Connection established, IO controller in STOP state Used to identify the slave Fatal error Device name error IP address error Configuration error
4	MS Module Status	Off Green Green, one flash Red Alternating red/green	Not initialized Normal operation Diagnostic event Fatal error Firmware update

G.2 Operation State LED Status

The .NET Bridge operation state is indicated by the LED status indicators.

State	LED Status	Description
Start up	IT: Off OT: Off	The .NET Bridge is connected to power and started but not yet connected to the .NET application or the PLC.
IT connected OT not connected	IT: Fast green blink OT: Off	The .NET Bridge is connected to the .NET application and has initialized the communication interface to the PLC, but there is no communication between the .NET Bridge and the PLC.
Pre-operational	IT: Slow green blink OT: Green	The .NET application and the PLC are both connected to the .NET Bridge. No I/O data exchange.
Operational	IT: Green OT: Green	The system is fully functional. I/O data exchange.
IT Disconnected OT connected	IT: Off OT: Green	The .NET application is disconnected from the .NET Bridge, after exchanging data in Operational state. The PLC side is still active.

H Firmware Update

H.1 Installing the Anybus Firmware Manager II

The Anybus Firmware Manager II is used to handle firmware in HMS devices.

Before You Begin

Visit www.anybus.com/support and download the *Firmware Manager II* zip file.

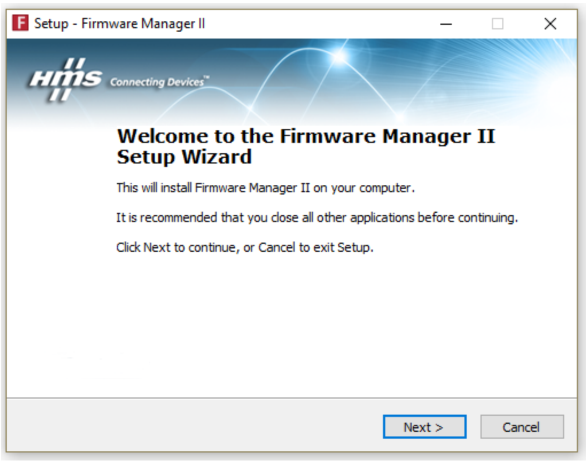
Before starting the installation, it is recommended that you close all other applications.

Procedure

Installing the Anybus Firmware Manager II:

1. Unzip the Anybus Firmware Manager Setup zip file.
2. Double-click the Firmware Manager Setup file.
3. The Setup Firmware Manager II installer window appears.

   Click **Next** to begin installation.



Welcome to the Firmware Manager II Setup Wizard

This will install Firmware Manager II on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Fig. 43 Anybus Firmware Manager II Setup Wizard

4. Follow the prompts in the Firmware Manager II Setup Wizard to complete the installation.
5. Click **Finish**.

Result

The Anybus Firmware Manager II is installed on your computer.

H.2 .NET Bridge Firmware Update

Before You Begin

- Ensure that Firmware Manager II is installed on your computer.
- Ensure that you have access to the firmware update file.

  How to access firmware update files:

  – From the Configuration file zip, generated from the Anybus .NET Bridge Generator, refer to *Configuration Files, p. 20*.

  – Firmware update files can be downloaded at www.anybus.com/support.

Procedure

Update .NET Bridge firmware:

1. Connect the .NET Bridge USB Port to your computer.
2. In your web browser, type the .NET Bridge IP address and enter the Anybus .NET Bridge Web Interface start page.
3. In the Firmware download pane, click **Enter firmware download mode**.

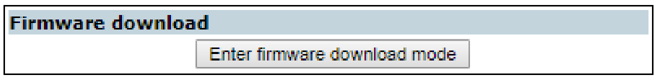


Fig. 44 Enter firmware download mode

   The .NET Bridge enters firmware download mode.

4. Open Firmware Manager II.

   ► Follow the instructions in the built-in help.

H.3 .NET Library Update

Update the .NET application with a new .NET library.

Before You Begin

- For information about Communication Design and Configuration files, refer to *Communication Design, p. 10*.
- For information about Developing the .NET application, refer to *Developing the .NET Application, p. 23*.

Procedure

In Visual Studio:

1. Open the .NET application project.
2. Remove the reference to the current *AnybusNetBridge.dll* assembly.
3. Do one of the following:
   – If the assembly (DLL) pre-built from the source code file is used, remove the assembly.
   – If the C# source code file generated from Anybus .NET Bridge Generator is used, remove that source code file.
4. To add the updated .NET library to the project, follow the steps in chapter *Importing References, p. 23*.
5. Make a new build of the .NET application.