Anybus CompactCom 40 Diagnostic Events for EtherCAT

SCM-1202–070 1.0 ENGLISH

Connecting Devices™

Important User Information

Liability

Every care has been taken in the preparation of this document. Please inform HMS Industrial Networks AB of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks AB, reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be considered as a commitment by HMS Industrial Networks AB. HMS Industrial Networks AB assumes no responsibility for any errors that may appear in this document.

There are many applications of this product. Those responsible for the use of this device must ensure that all the necessary steps have been taken to verify that the applications meet all performance and safety requirements including any applicable laws, regulations, codes, and standards.

HMS Industrial Networks AB will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features, timing, or functional side effects found outside the documented scope of this product. The effects caused by any direct or indirect use of such aspects of the product are undefined, and may include e.g. compatibility issues and stability issues.

The examples and illustrations in this document are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks AB cannot assume responsibility for actual use based on these examples and illustrations.

Intellectual Property Rights

HMS Industrial Networks AB has intellectual property rights relating to technology embodied in the product described in this document. These intellectual property rights may include patents and pending patent applications in the USA and other countries.

1 Preface

1.1 About this Document

This application note is intended to provide a description about how diagnostic events are presented in the engineering tool for the industrial network EtherCAT.

It is divided into two parts:

Part one provides a short overview of the Anybus CompactCom 40 Diagnostic Object (02h) and its diagnostic events. Part two is an example, showing how to get diagnostic messages displayed for EtherCAT using the PLC engineering tool TwinCAT 3.1.

1.1.1 Target Audience

This document is meant for trained and skilled personnel working with the equipment described.

You need electrical engineering skills for the installation and commissioning of electrical equipment.

You also need general knowledge of automation and programmable logic controllers, in particular about Beckhoff Automation software. Additionally, knowledge of the EtherCAT industrial network protocol and the Anybus CompactCom object model is necessary.

1.2 History

Revision	Date	Description	Responsible
0.9	2016-10-17	First version	OLB
1.0	2018-02-14	Converted to DOX	KaD

1.3 Referenced Documents

Description	Name / Type	Version
HMS Starter Kit	HMS Development Board 2 Rev 1.13	0314-1.1.1
Anybus CompactCom 40 module	ABCC-M40-ECT	FW V.2.08
Beckhoff PLC	CPU: CX9020	-
GSD file for the Anybus CompactCom 40 EtherCAT	HMS Anybus CompactCom 40 EtherCAT 2_ 08.xml	2.08
Anybus CompactCom 40 EtherCAT	Network Guide	V1.7
Anybus CompactCom Software Design guide	Software Design Guide	V3.4
PC with Beckhoff PLC programming software	TwinCAT XAE PLC 3.1	3.1.0.0.4016.1.2
How to configure an Anybus EtherCAT slave module with Beckhoff PLC	Application Note	1.02
IDE	KEIL uVision 5	V5.20
Anybus CompactCom Driver	Anybus CompactCom Host Application Example Code	V3.03

1.4 More Information about Networks and Products

The latest manuals and GSD files can be found on the HMS website, www.anybus.com

1.5 Trademark Information

Anybus® is a registered trademark of HMS Industrial Networks AB.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

All other trademarks are the property of their respective holders.

2 Introduction

The ability of an automation device to raise diagnostic events is an important benefit. It may not only reflect a higher level of quality setting it apart from competitors in the market, in the end it will also increase the device's reliability, reduce downtimes by relieving preventive maintenance and protect the investment.

Implementing this feature set allows the device to signal significant incidents towards the PLC. This way operators will immediately be informed in case the device experiences errors or faults.

This document describes how developers of a field device implementing an Anybus CompactCom 40 EtherCAT network interface can use its features to create diagnostic events.

3 The Diagnostic Object

The Anybus CompactCom concept is based on an object model.

For detailed information about this, please refer to the Anybus CompactCom 40 Software Design Guide.

3.1 Create a Diagnostic Event

When the CompactCom device has been started and initialized, the diagnostic object (02h) is solely built up of its object instance #0 containing attributes common to all subsequent instances which may be added in the course of the device's operation. The process of reporting a diagnostic event to the network is initiated by the host application. To this end a ***create (03h)*** event command message is sent from the host application to the diagnostic object (02h).

The CompactCom will internally create a new instance inside the diagnostic object (02h). This new instance corresponds to a **diagnostic event**.

The ***create (03h)*** command must contain info for describing the diagnostic event: a **Severity** (CMDExt[0]) and an **Event Code** (CMDExt[1]). The **severity** indicates how critical the event is and if it will recover by itself or not. CMDExt[0] additionally contains a bit called ***Extended Diagnostic***, which informs the CompactCom if the event message contains additional user specific data. If the ***Extended Diagnostic*** bit is set, the additional user specific data will be found in MsgData[0..7], and the network specific data will follow in MsgData[8..n].

An **Event Code** informs about the nature of the event, i. e. what caused the device to react, like exceeding temperature or current limit.

Command Details: Create

Details

Command Code: 03h

Valid For: Object

Description

Creates a new instance, in this case representing a new diagnostic event in the host application.

- Command details:

Field	Contents	Note
CMDExt[0]	Bit 0: Extended Diagnostic Bit 4–6: Severity Other bits Reserved. Set to zero.	
CMDExt[1]	Event Code, see previous page	
MsgData[0...1]	Slot number associated with the event Set to “0” if unknown or unsupported	These fields only exist if bit 0 (Extended Diagnostic) is set
MsgData[2...3]	ADI associated with the event Set to “0” if unknown or unsupported	
MsgData[4]	Element associated with the event Set to “255” if unknown or unsupported	
MsgData[5]	Bit in element associated with the event Set to “255” if unknown or unsupported	
MsgData[6...7]	Reserved. Set to zero	
MsgData[0/8...n]	Network specific extension (optional, definition is network specific)	MsgData[8...n] if bit 0 in CmdExt[0] is set MsgData[0...n] if bit 0 in CmdExt[0] is not set

- Response details (Success):

Field	Contents
MsgData[0...1]	The number of the instance that was created as a result from the command

- Response details (Error):

Error	Contents	Comment
Object Specific Error	MsgData[1] = 02h	Error code (Latching event not supported) The event could not be created since the module does not support latching events
	MsgData[1] = FFh	Error code (Network specific error) The event could not be created due to a network specific reason. Information about the event is found in response MsgData[2-n]

3.2 Severity Codes of Diagnostic Events

The following severity codes are defined for the CompactCom:

This parameter indicates the severity level of the event. Only bits 4 - 6 are used for severity level information.

Bit Combination	Severity	Comment
000	Minor, recoverable	-
001	Minor, unrecoverable	Unrecoverable events cannot be deleted
010	Major, recoverable	-
011	Major, unrecoverable	Causes a state-shift to EXCEPTION
101	Minor, latching	
110	Major, latching	
(other)	-	(reserved for future use)

Typically, *recoverable* events are generated by the process e.g. if a temperature exceeds a limit value defined by the device manufacturer (e.g. internal temperature of the device exceeds 50° C). The character of this event is typically a warning when the event is defined as *minor*. The temperature of the device is recoverable as the device can cool down again when some heat producers are cut off.

The device manufacturer will add a *major recoverable* event when he wants to inform the PLC that the temperature has reached a critical high temperature which can damage the device.

An ***unrecoverable*** diagnostic event is typically created when the device detects that a part of the application is malfunctioning. A ***major unrecoverable*** event signals that the device has to be stopped, inspected and searched for critical errors. A restart may solve the problem, or the malfunctioning part might need to be replaced. A ***minor unrecoverable*** event informs the users that the application is still working, but part of it needs to be searched for errors and/or replaced within the next scheduled inspection.

Minor latching and ***major latching*** allow the creation of *latching diagnostic event.*

In EtherCAT, latching events are not supported.

For more information, see the Software Design Guide and the Network Guide of the Anybus CompactCom 40 device.

> **!** The device manufacturer has to define which event will be reported as a diagnostic event to the PLC and which severity has to be used.

3.3 Anybus CompactCom Event Codes

The table below shows a list of the event codes applicable for the Anybus CompactCom 40 device.

#	Meaning	Comment
10h	Generic Error	-
20h	Current	-
21h	Current, device input side	-
22h	Current, inside the device	-
23h	Current, device output side	-
30h	Voltage	-
31h	Mains Voltage	-
32h	Voltage inside the device	-
33h	Output Voltage	-
40h	Temperature	-
41h	Ambient Temperature	-
42h	Device Temperature	-
50h	Device Hardware	-
60h	Device Software	-
61h	Internal Software	-
62h	User Software	-
63h	Data Set	-
70h	Additional Modules	-
80h	Monitoring	-
81h	Communication	-
82h	Protocol Error	-
90h	External Error	-
F0h	Additional Functions	-
FFh	NW specific	Definition is network-specific; consult separate network guide for further information.

The event code **FFh** is used if network specific diagnostics are reported (not considered within this application note).

4 EtherCAT

To show the commissioning within the Beckhoff environment we will be using the TwinCAT 3.1 engineering tool, starting with a blank project. In step one, we will configure the Beckhoff PLC. In step two, the EtherCAT network and its devices.

4.1 PLC Configuration

4.1.1 Importing the EtherCAT Slave Information File (ESI)

In order to integrate the Anybus CompactCom with the Beckhoff EtherCAT controller, it is necessary to import the ESI file into the TwinCAT software. To do this, locate the installation directory of TwinCAT on your PC and paste a copy of the ESI file into the subdirectory << ..\3.1\Config\Io \EtherCAT >> as indicated in the picture below. You can now start the TwinCAT software.

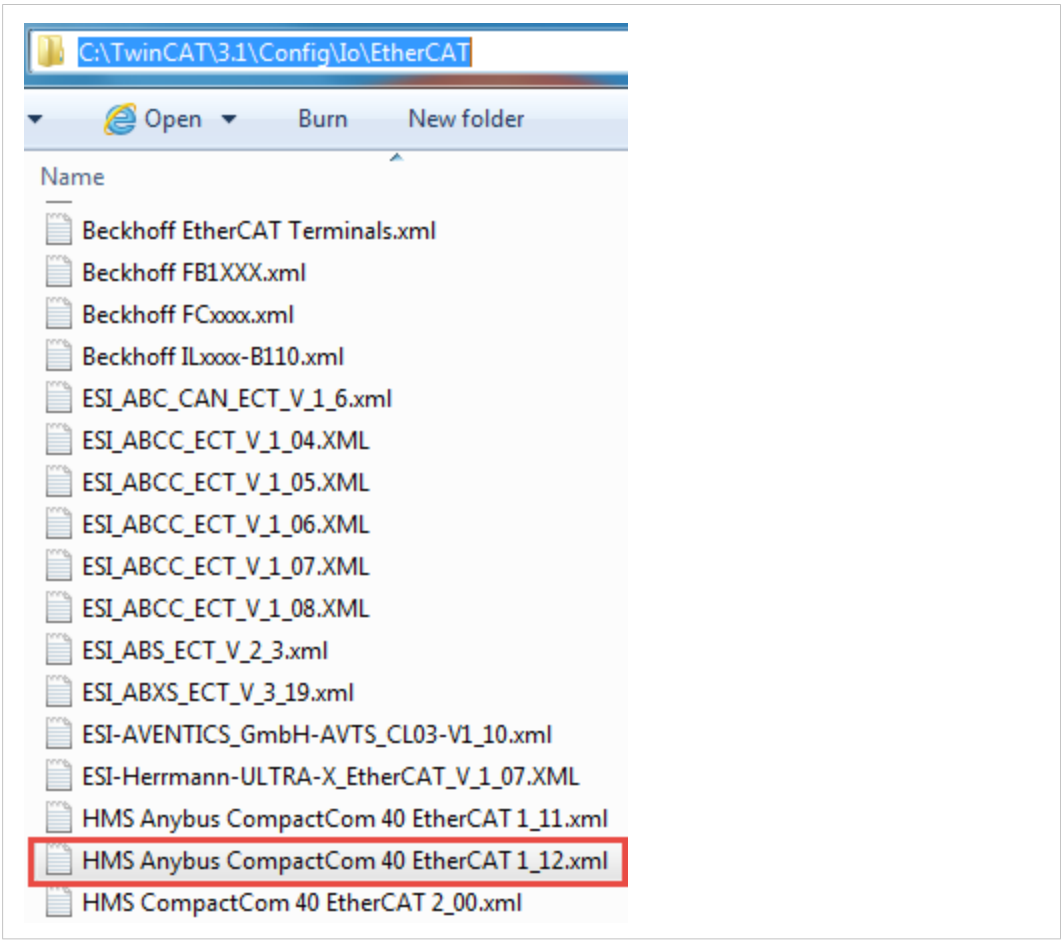

Fig. 1 Importing the EtherCAT Slave Information File (ESI)

4.1.2 Creation of a New Project

When TwinCAT starts, you will get a default interface, as shown in the following figure.

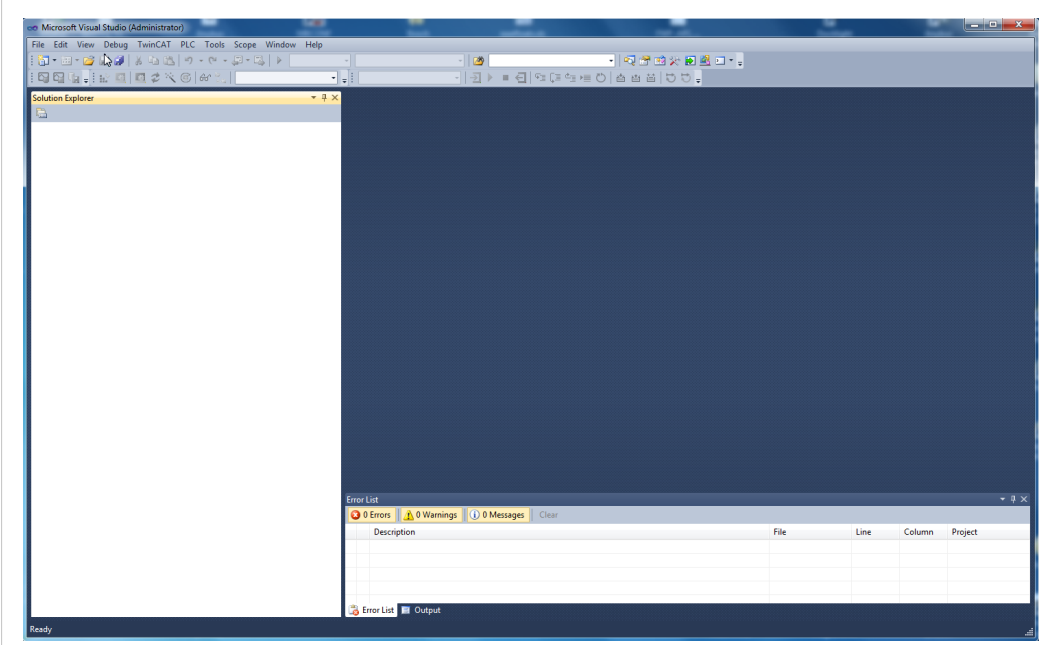

Fig. 2

We can now create a new project. To do this, go to the file menu and selects File > New > Project. We then get a window for creating a new project, as shown in the picture below. Choose a name and validate the creation of the project.

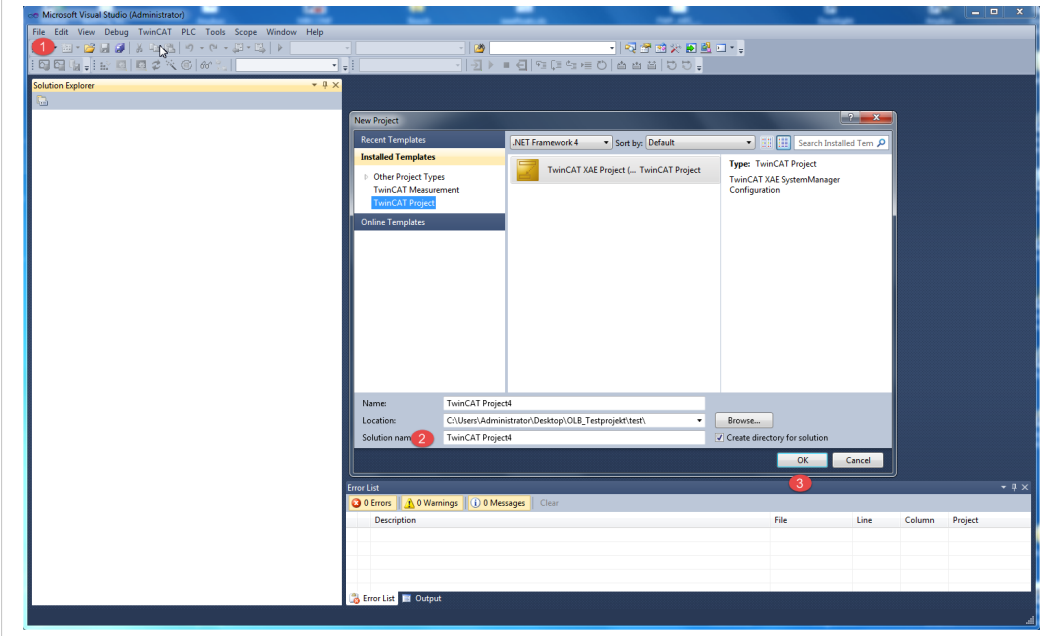

Fig. 3

After the project is created, the TwinCAT interface must match that shown in the picture below.

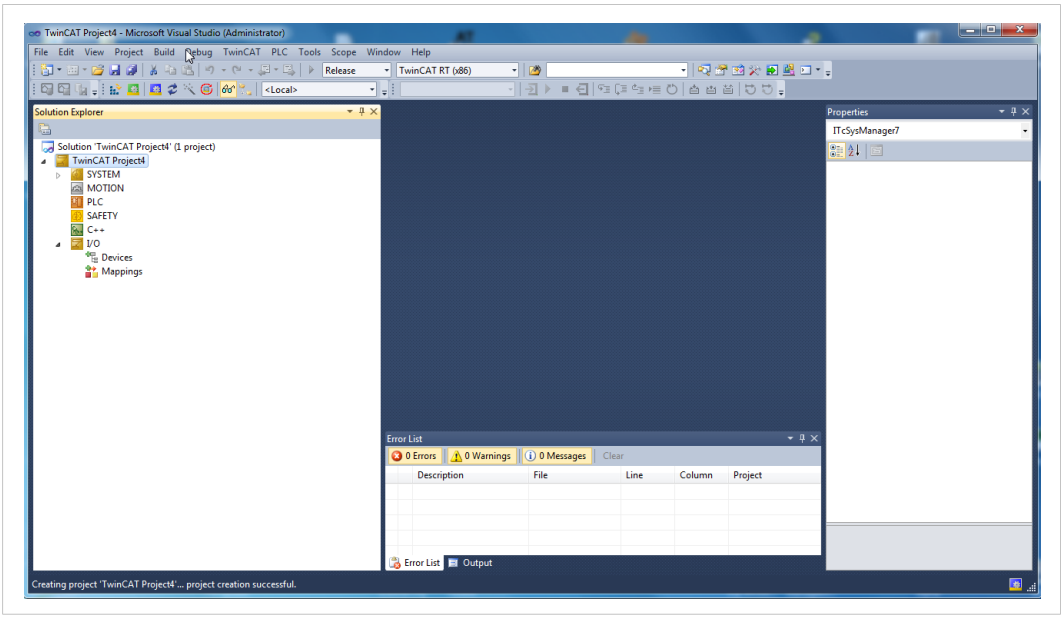

Fig. 4

4.1.3 Discovery and Addition of EtherCAT PLC

The next step is to add the PLC to the project. EtherCAT provides PLC detection capabilities over a LAN. We will rely on these features in the rest of this chapter. Let's start by selecting the target of our project, via the item 'Choose Target System' from the drop-down menu, as shown below in the picture.

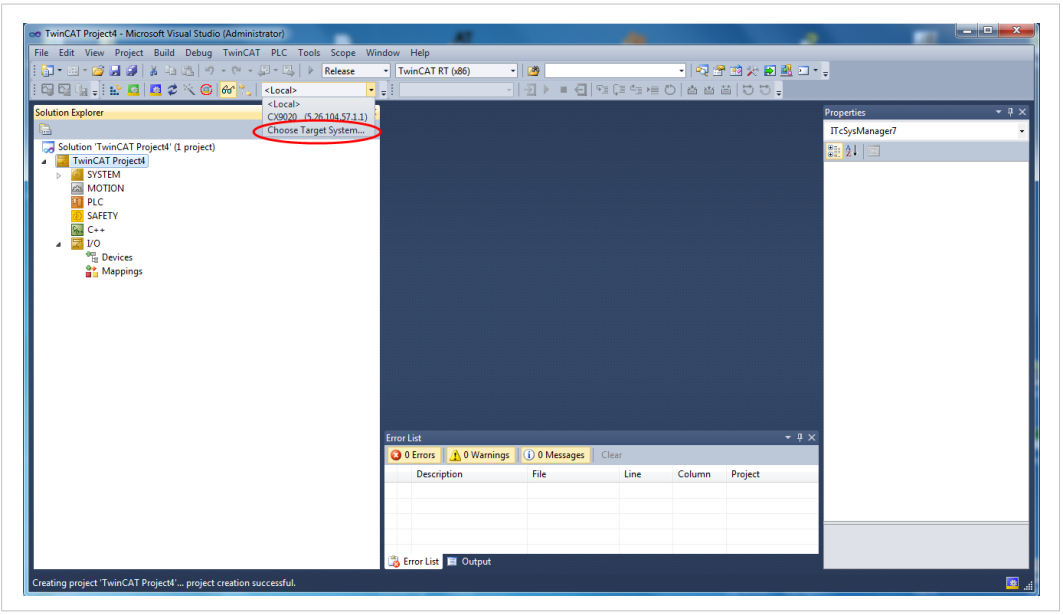

Fig. 5

We will then get the window shown below.

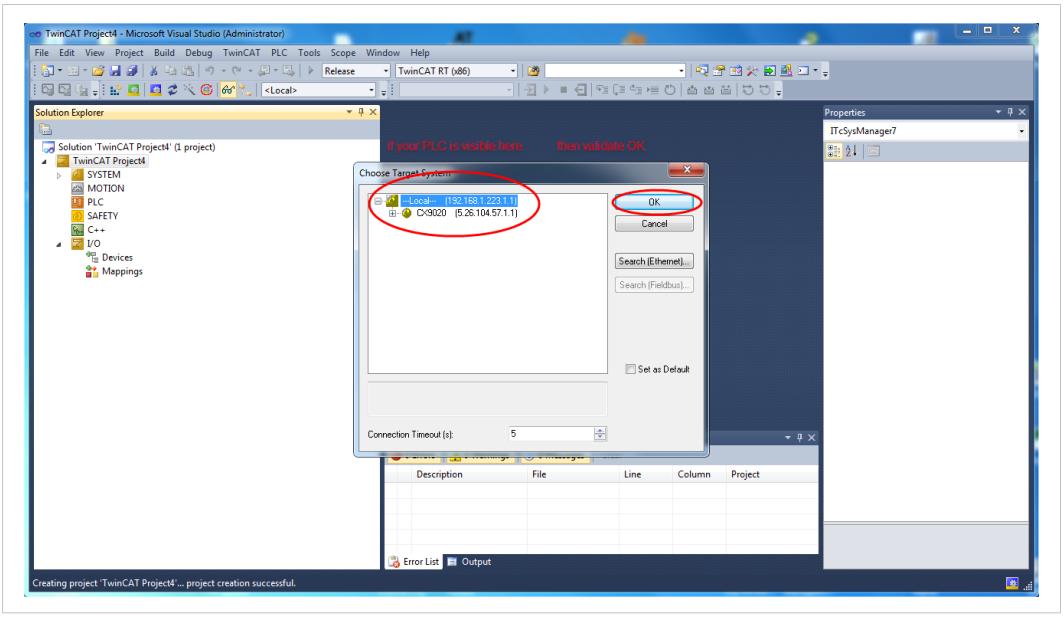

Fig. 6

If our controller is already visible in the 'Choose Target System' window in the picture, we can directly select it here, validate with OK and go directly to *Network Configuration and Activation, p. 16*

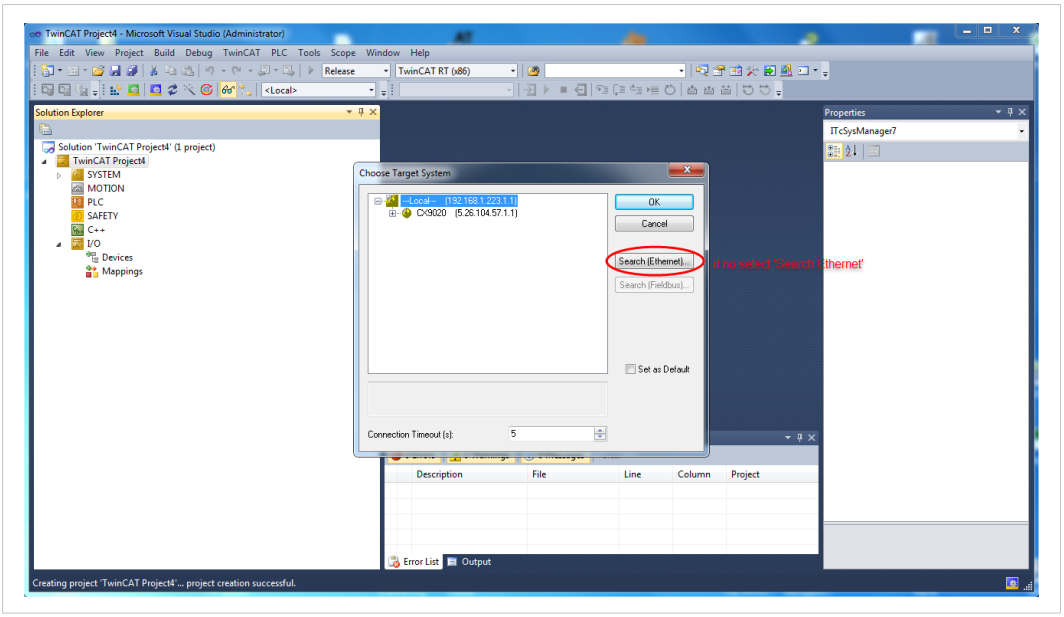

Fig. 7

If not, proceed as usual. We will look for the PLCs on our Ethernet network. Select 'Search (Ethernet) ...' as shown above in the picture. Keep the default settings as shown in the next picture below and click 'Broadcast search'.

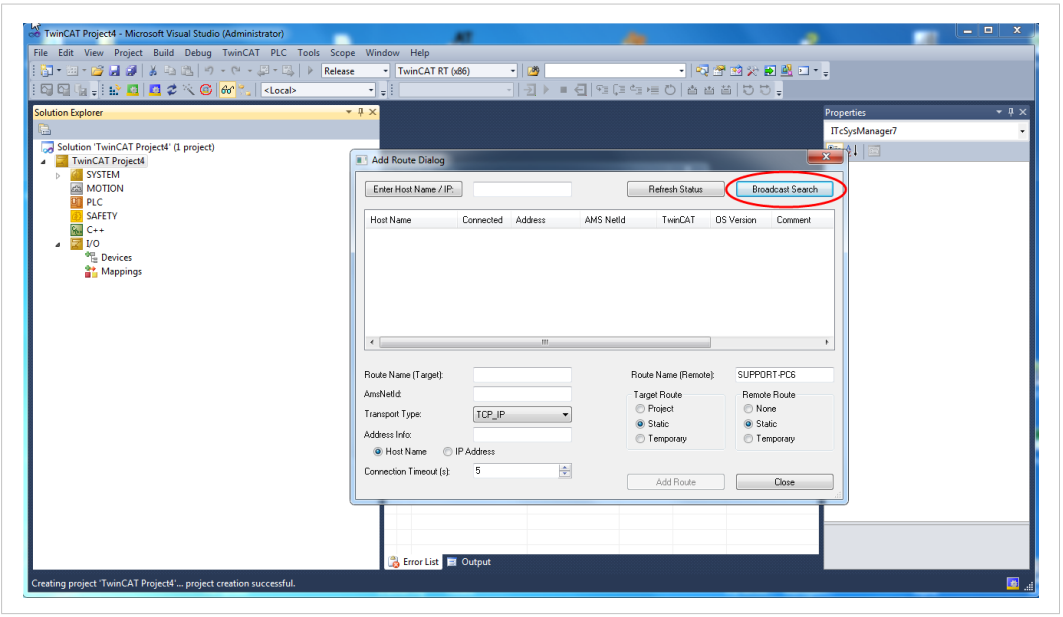

Fig. 8

We locate our Beckhoff controller on the network, as shown in the next picture: Select the PLC route and add it.

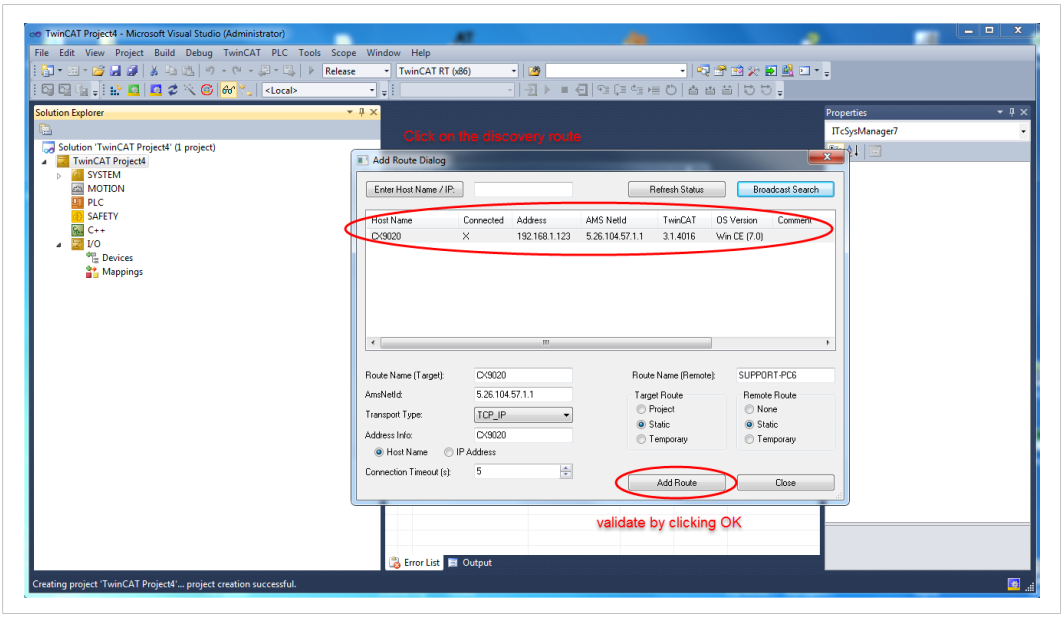

Fig. 9

We will then get a login window as shown in the picture below. Here, we enter the login IDs of our controller.

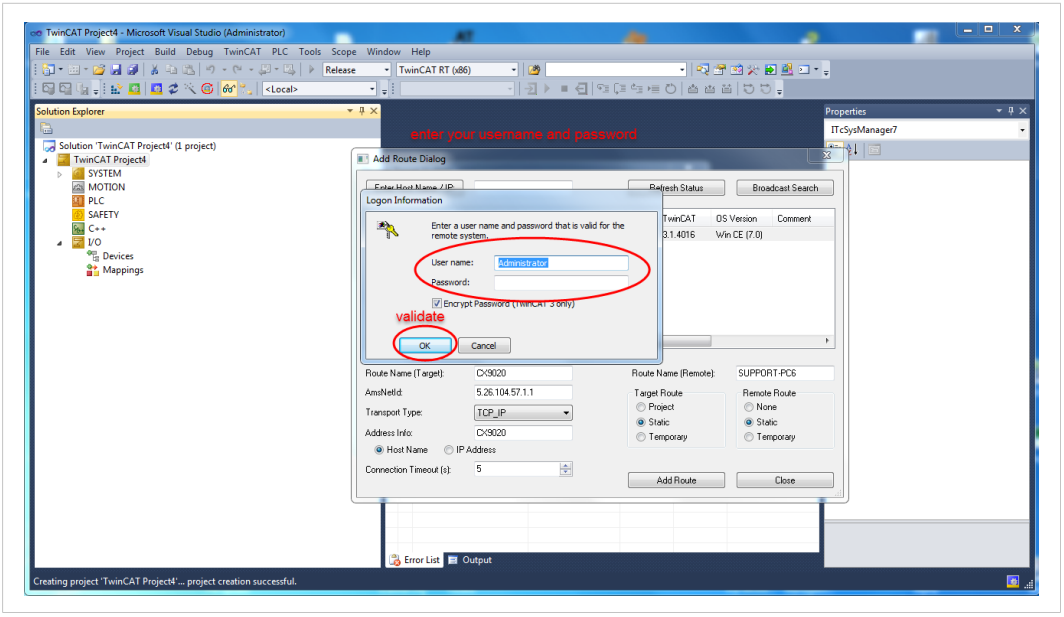

Fig. 10

We can now select our controller, as shown in the next picture. Let us validate by clicking on ‘OK’

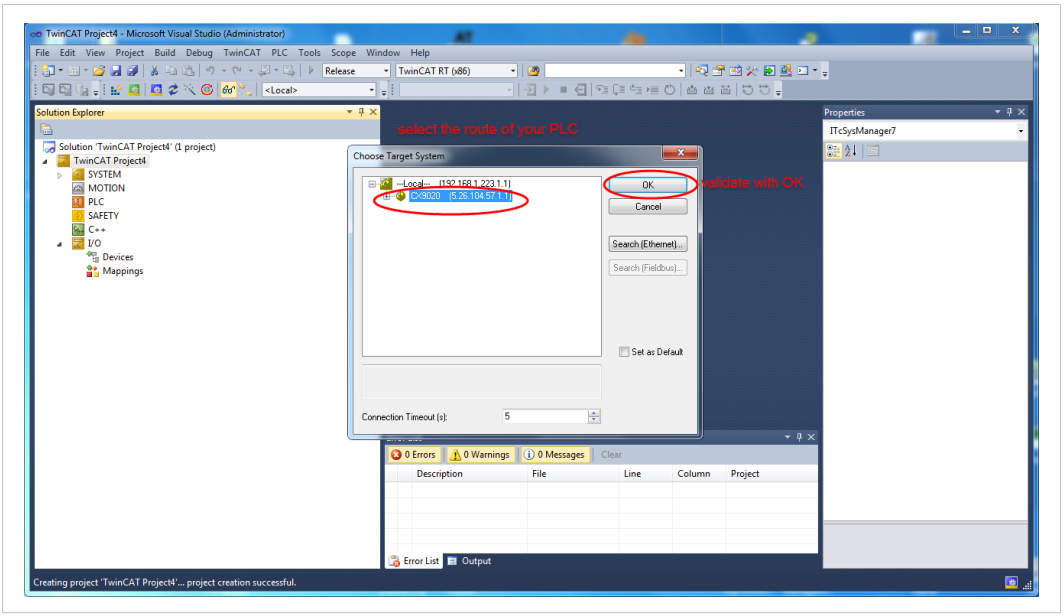

Fig. 11

4.1.4 Network Configuration and Activation

Confirm the change of environment on the window in the picture below by clicking on 'Yes'.

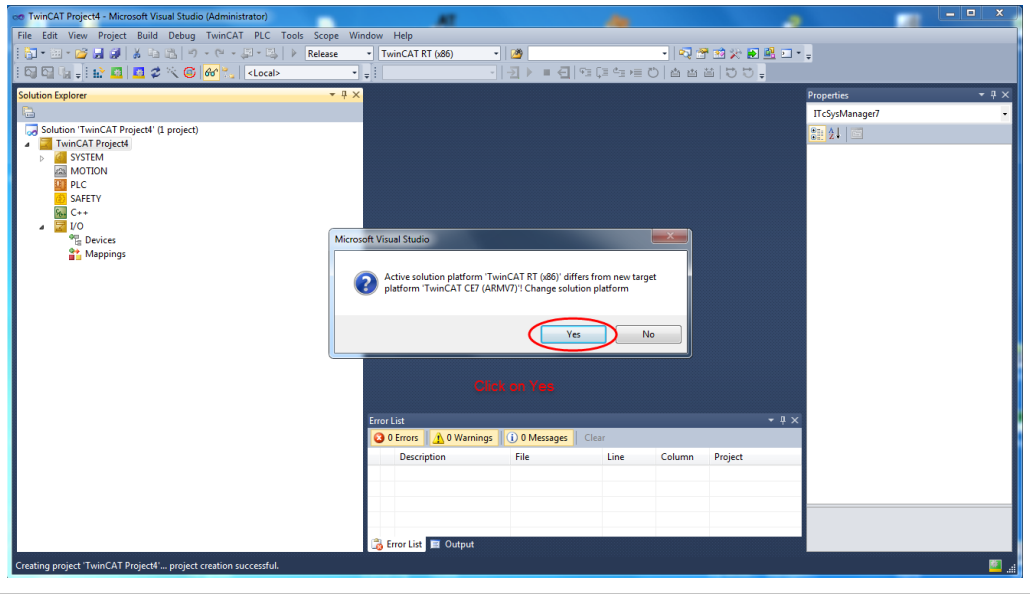

Fig. 12

We will now restart our TwinCAT controller in config mode. To do this, select 'Restart TwinCAT (Config Mode)' from the TwinCAT menu before validating, as shown in the picture below.

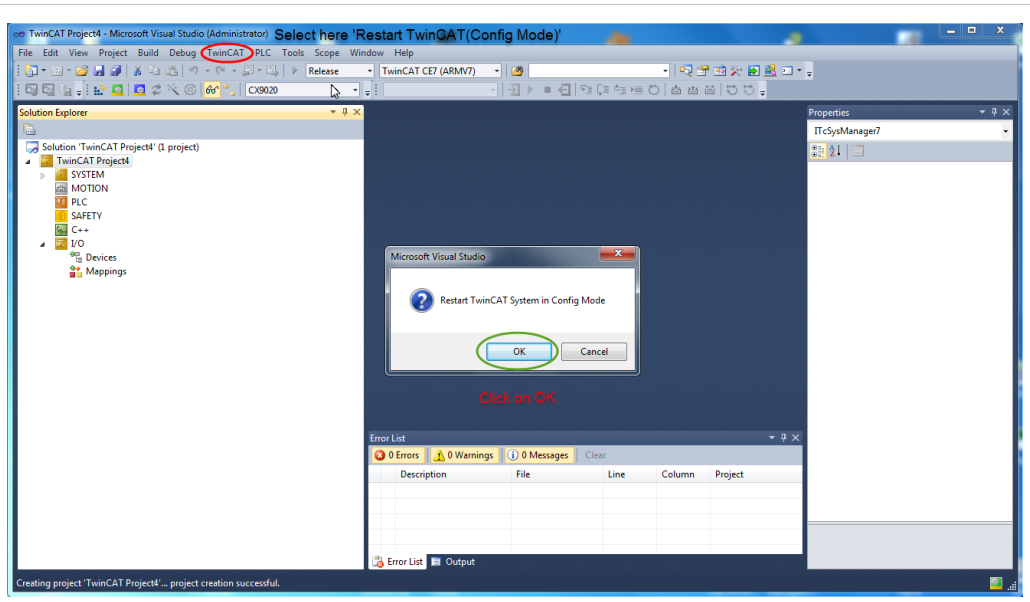

Fig. 13

We will then get a confirmation message, as shown in the next picture.

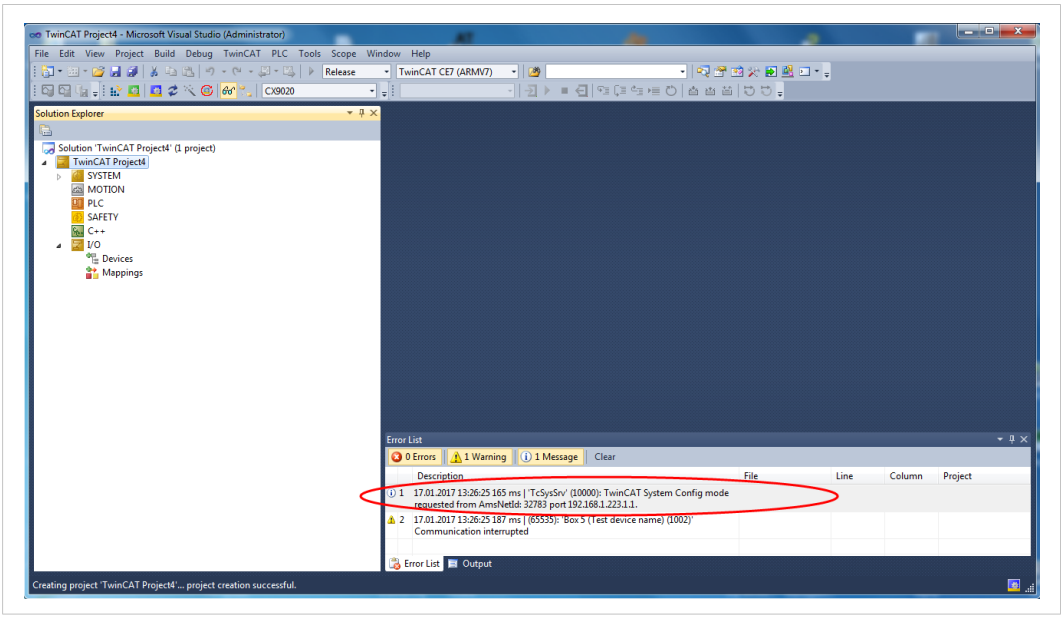

Fig. 14

We will now ask our controller to scan the slaves attached to its EtherCAT interface. To do this, we right-click on the Device item under the 'I / O' group, as shown below.

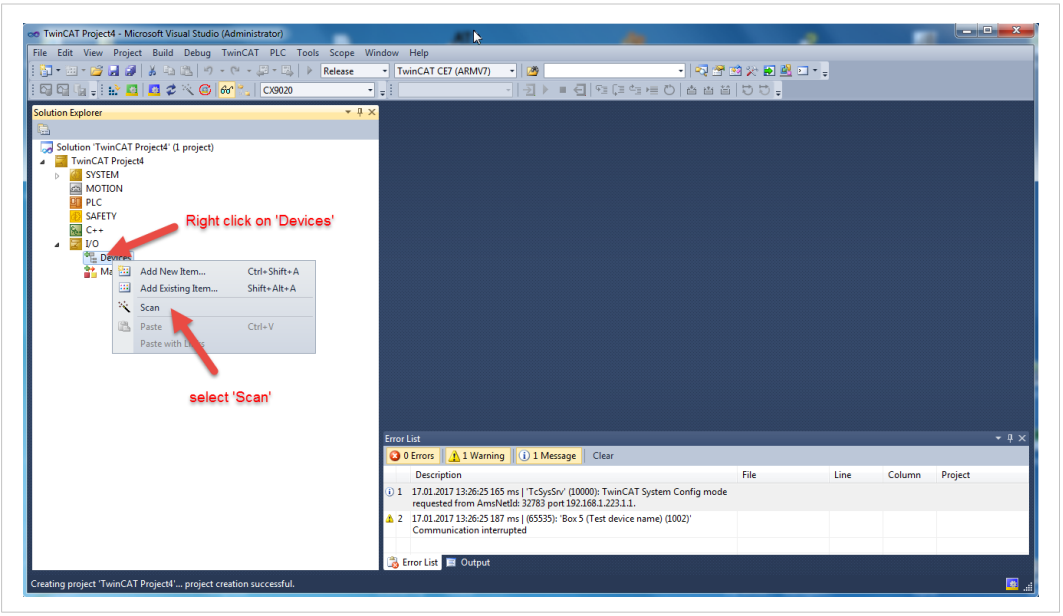

Fig. 15

A message appears, warning us that certain types of slaves may not be discovered automatically, as shown below in the picture. This does not concern us. Validate by clicking OK.

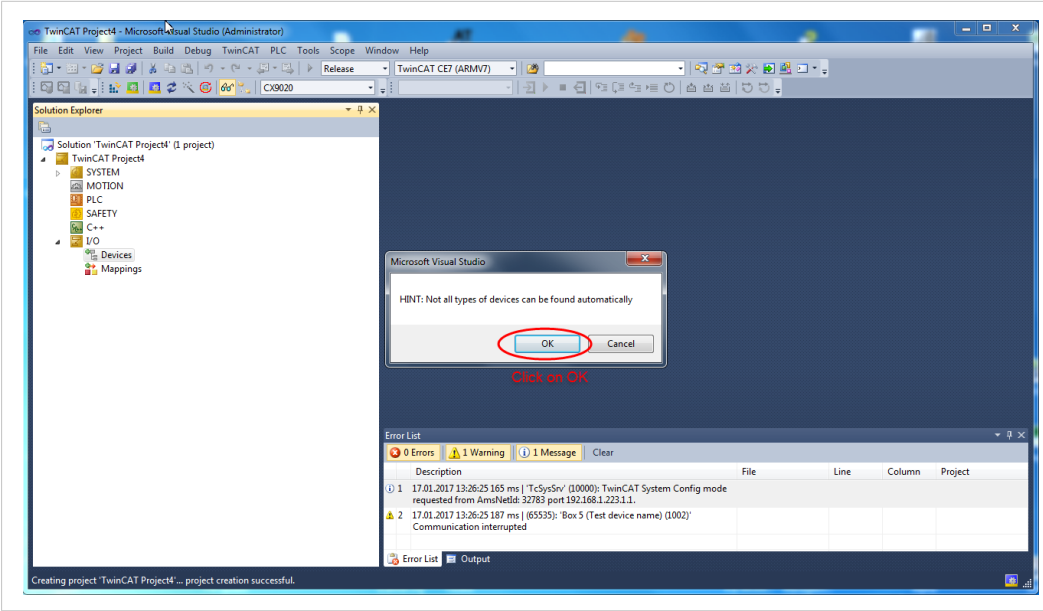

Fig. 16

We get an I / O list, as shown in the picture below. Let's select only the first item in the list before validating by clicking OK.

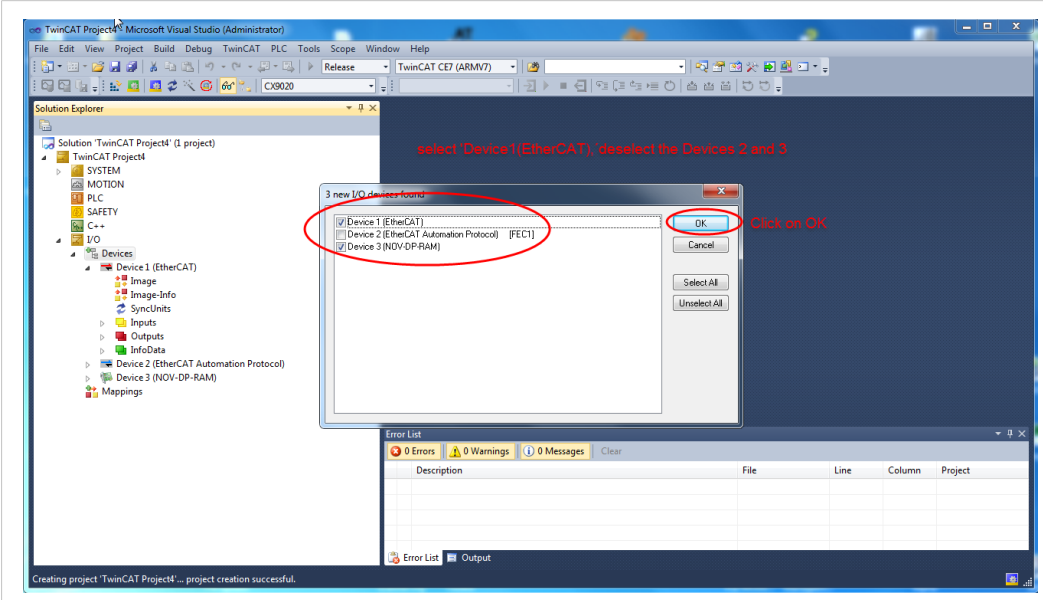

Fig. 17

A new dialog box shows 'Scan for boxes', as shown below. Let's validate again by clicking on 'Yes'.

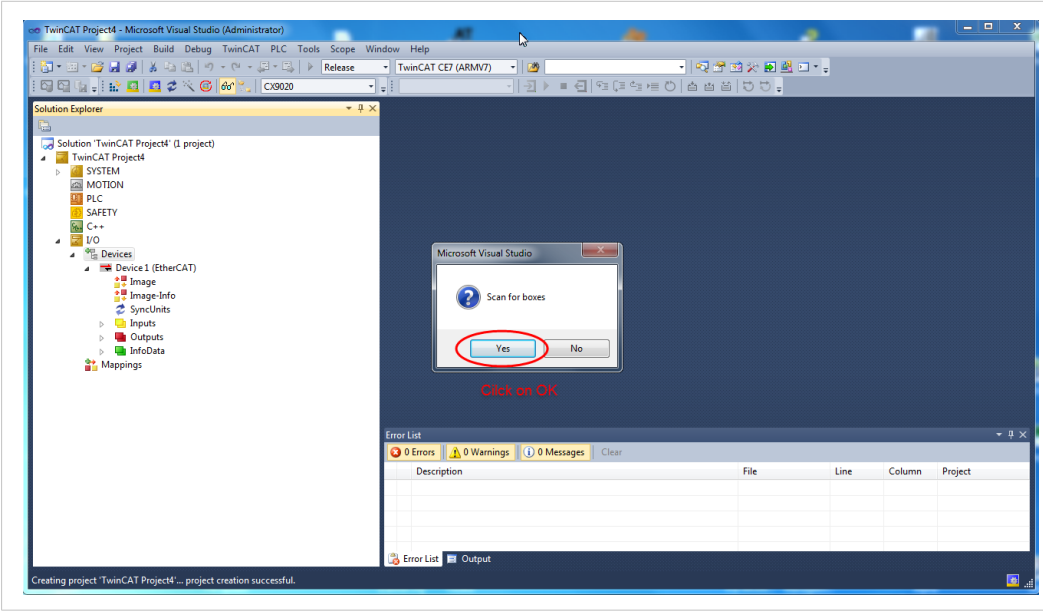

Fig. 18

Our CompactCom device is now configured on the EtherCAT controller. A new dialog box proposes to go into Free Run mode.

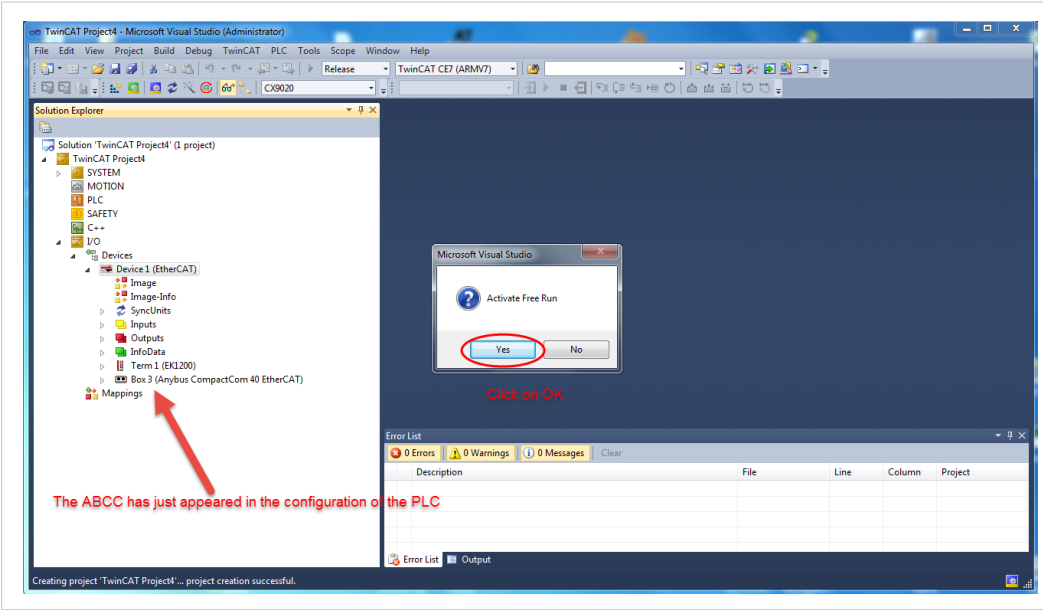

Fig. 19

If we wish, we can validate the free run mode by clicking on 'Yes'. As the CompactCom is also enabled, we can then see the Process Data Units transmitted from the CompactCom to the PLC as shown in the picture below

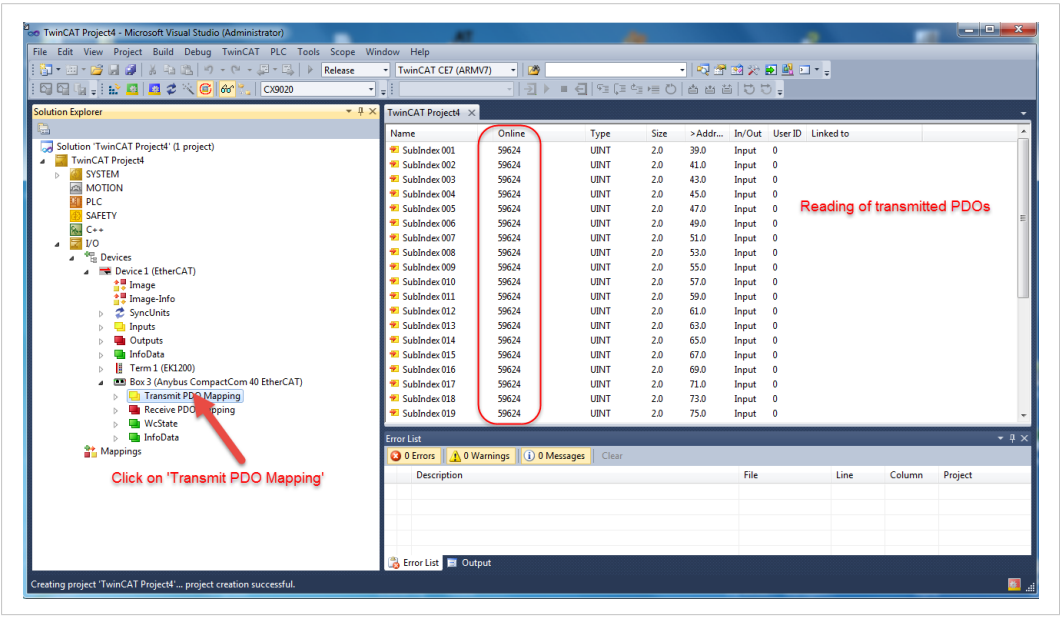

Fig. 20

4.2 EtherCAT Diagnostics

This section shows how EtherCAT handles diagnostic data.

EtherCAT provides three possibilities for reporting diagnostics:

- via SDO access
- via the diagnosis history object (0x10F3)
- via emergency messages

In this document, since the diagnosis history object (0x10F3) is not supported by the Anybus CompactCom EtherCAT slave module, we are using the default emergency object to report diagnosis messages. However, the diagnosis history object (0x10F3) can be implemented by the application itself if needed.

4.2.1 SDO Access

This is the acyclic access. The master can obtain diagnostics information from EtherCAT via SDO access.

4.2.2 Diagnosis History Object (0x10F3)

The EtherCAT specification has defined a dedicated object for reporting diagnosis messages. This object can record and display up to 250 messages, including different types of diagnosis messages: info, warning, error.

A text ID is assigned to each error message with a detailed description stored in the ESI file.

Diagnostics messages can be read e.g. by the TwinCAT system manager in the diag history tab, provided that the EtherCAT slave device supports this object. The following table shows the structure of the diagnosis object 0x10F3.

Index	Sub-Index	Description	Type	Access
0x10F3	0	Diagnosis History	-	-
	1	Maximum messages	UINT8	R/RW
	2	Newest message	UINT8	R
	3	Newest acknowl-edged message	UINT8	RW
	4	New message available	Bool	R
	5	Flags	UINT16	RW
	6….225	Sub-Index 006… 0255	OCTET STRING	R

4.2.3 Diagnosis by Emergency Messages

When an error occurs/disappears within a slave device, an emergency message will be sent out with the following defined structure. In the Beckhoff software, TwinCAT System Manager Emergency messages will be displayed in the message window.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
00h	Event Code	Error Register (1001h)	Manufacturer specific field (not used)				

The emergency telegram above is represented in hexadecimal format. It is divided into three parts: the **error code**, the **error register,** and the **data**.

The **error code** is defined by two bytes and indicates the error type, for example, the error code *0x20xx* corresponds to current and *0x30xx* to voltage. For the Anybus CompactCom 40 module the error code is contained in the lower two bytes (byte 0 and byte 1). Byte 0 is fixed with the value 00h and byte 1 corresponds to the **Event Code**. If *the event code is FFh, the severity code is part of the emergency message*. In all other cases ***only*** *the event code is reported*.

For more information see the Anybus CompactCom 40 EtherCAT network guide in section 5.3.

The error register has a size of one byte and indicates which error type is present when a bit is set.

The data part consists of five bytes and carries either the manufacturer specific error field or diagnostic data. For the Anybus CompactCom 40 module, the data part includes byte 3 to byte 7.

The manufacturer specific information – which are represented by bytes 3 to 7- in the emergency request cannot be changed from the host application. Since the network specific diagnostics are not reported, the severity code will be not displayed. This means that it will be not possible to distinguish between *major recoverable*, *minor recoverable* and *minor unrecoverable.*

We will focus on the minor recoverable diagnostic event. Since the three other severity codes cannot be displayed, we only need to show one example illustrating the behavior of all of the three severity codes. In the end, we will also explain what happens when a major unrecoverable diagnostic event is created.

5 Application Example

In this example we are using the Anybus CompactCom Starter Kit in which we integrate the Anybus CompactCom 40 device. We use the host application example code for the Windows platform simulating the host application. We also run the automation engineering tool TwinCAT 3.1 by Beckhoff for configuring the PLC. In this example, we employ a CX9020 controller. The engineering tool will also report the diagnostic events sent by the CompactCom module to the PLC.

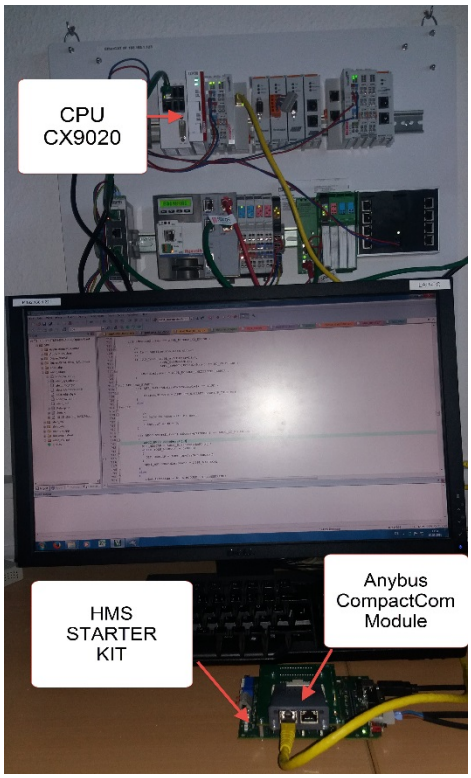

Fig. 21 Hardware overview

The following section will explain what has to be done from application side to initiate the CompactCom EtherCAT module to send field device diagnosis to the network master.

5.1 Code Sample for Creating a Diagnostic Event

This sample is intended to show how to create a diagnostic event in the CompactCom using the CompactCom host application example code. The example below shows the structure of a ***create (03h) event command message*** that generates a diagnostic event defined as *minor recoverable* for the event code *temperature*.

```
// message header part
ABCC_SetMsgHeader (psMsg,                          // buffer
                   ABP_OBJ_NUM_DI,                 // diagnostic object
                   0,                              // instance
                   0,                              // attribute
                   ABP_CMD_CREATE,                 // command type
                   10,                             // Message data size
                   ABCC_GetNewSourceId ());        // source id



// severity
psMsg->sHeader.bCmdExt0 = ABP_DI_EVENT_SEVERITY_MINOR_REC;

// event code
psMsg->sHeader.bCmdExt1 = ABP_DI_EVENT_TEMPERATURE;


// message data part
ABCC_SetMsgData16 (psMsg, 0, 0);
```

Fig. 22

The *create (03h)* event command above has created a diagnostic event including the severity code (***minor recoverable (0x00)***) and the event code (***temperature (0x40)***). There is no data included in the message in this case.

6 Readout of Diagnostics in TwinCAT 3.1

Start a TwinCAT project with all the settings complete, including your Anybus CompactCom EtherCAT slave module. Double-click on the CompactCom EtherCAT slave module icon (1) in the system manager. A window will be opened on the right, with all the tabs including all the objects supported by the CompactCom slave module (if available).

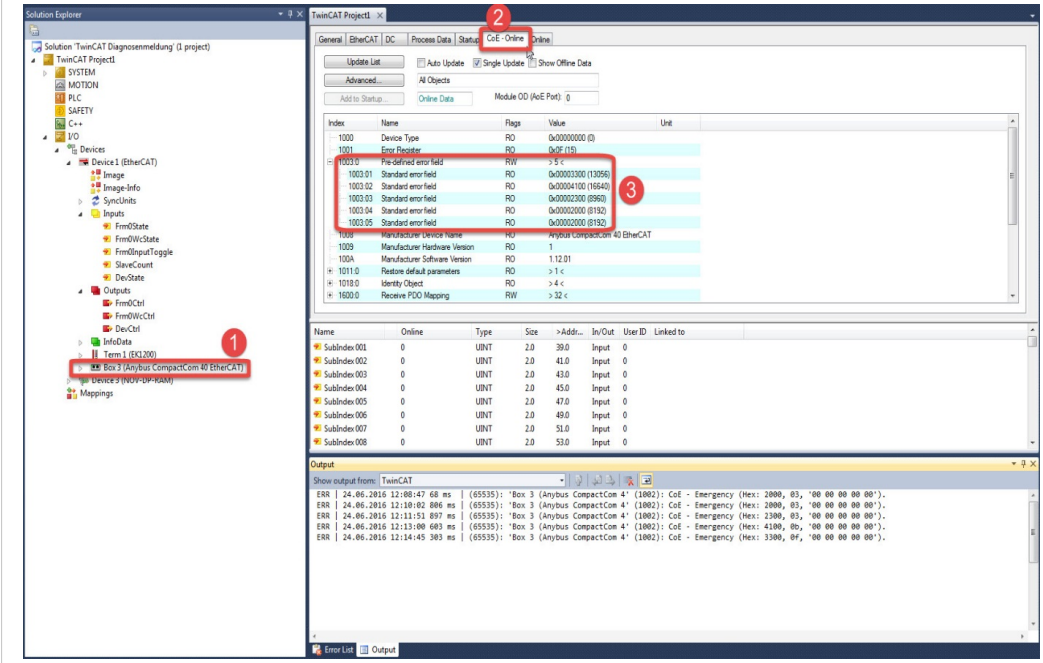

Fig. 23 Error information

Select the CoE-Online tab (2) and expand the predefined error field object (3) to get the information about the errors.

Once a diagnostic event is created in the Anybus CompactCom 40 module, diagnosis messages can be read out at any time from the system manager output window as shown in the picture below.

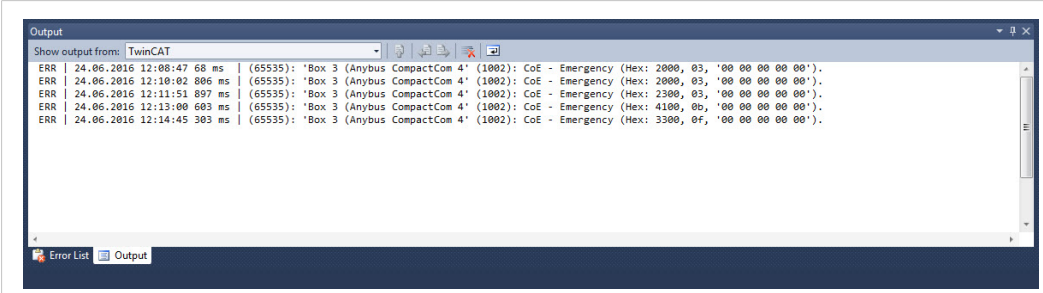

Fig. 24 System manager output window

6.1 Severity: Minor Recoverable and Major Recoverable

Creation of a diagnostic event

The picture above presents, for example, five active diagnostic events defined as *minor recoverable*.

The diagnostic window below shows the diagnostic information of an Anybus CompactCom 40 module.

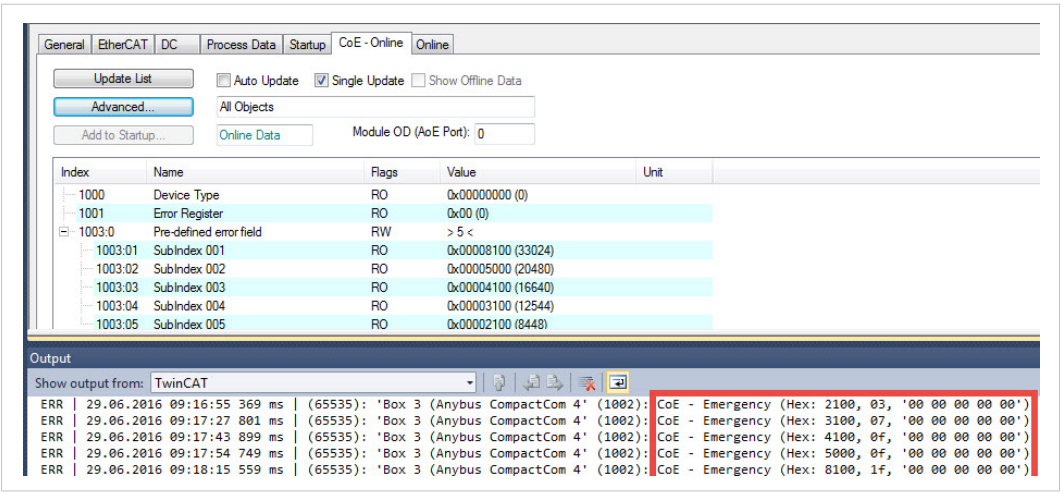

Fig. 25 Minor recoverable active diagnostic events

For the first diagnostic message in the red frame according to *Diagnosis by Emergency Messages, p. 22*, these bytes have the following meaning:

Byte 0, Byte 1:	21h 00h (error code) 21h: event code (“current device input side”) 00h: first byte of the error code
Byte 2:	03h (error register) informs that bit 0 and bit 1 are set the error type corresponding to current is present
Byte 3...7:	00h 00h 00h 00h 00h (data part)

Deletion of a diagnostic event

The figure below shows how to delete a diagnostic event.

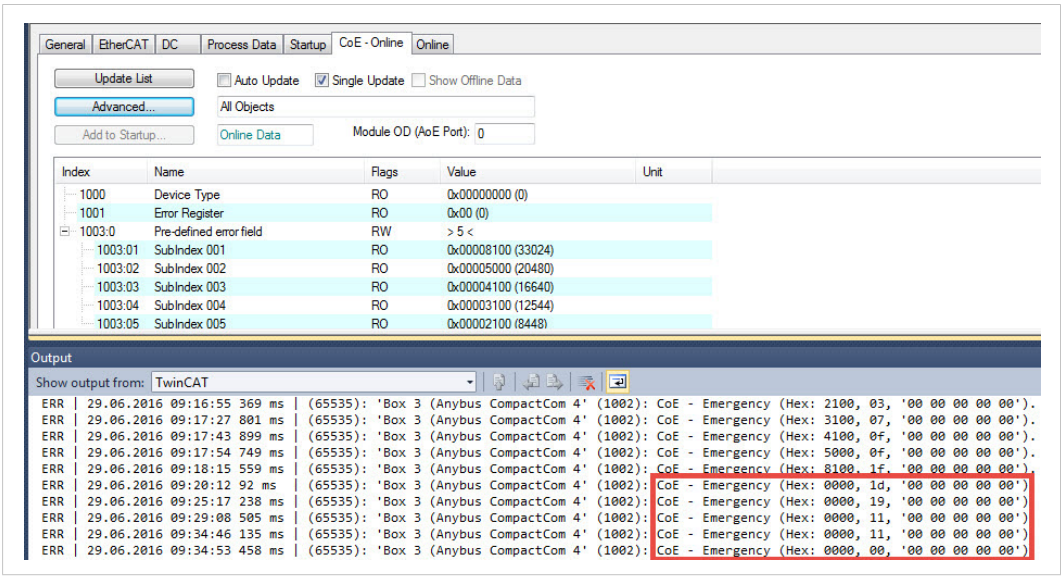

Fig. 26 Minor recoverable active diagnostic event deleted

A diagnostic event is deleted when the emergency telegram is sent with the error code **0000h**. The five diagnostic events created above are successfully deleted, one by one. The last emergency telegram in the red frame shows beside the error code (0000h) also that the error register has changed his value, namely all bits are set to 0 now - which means there is no error present.

6.2 Severity: Major Unrecoverable

The creation of a *major unrecoverable* diagnostic event causes the Anybus CompactCom 40 module to enter the EXCEPTION state. This disconnects the module from the network, which makes it impossible to send out an emergency message on the bus. The picture below shows the message in the output of TwinCAT system manager.

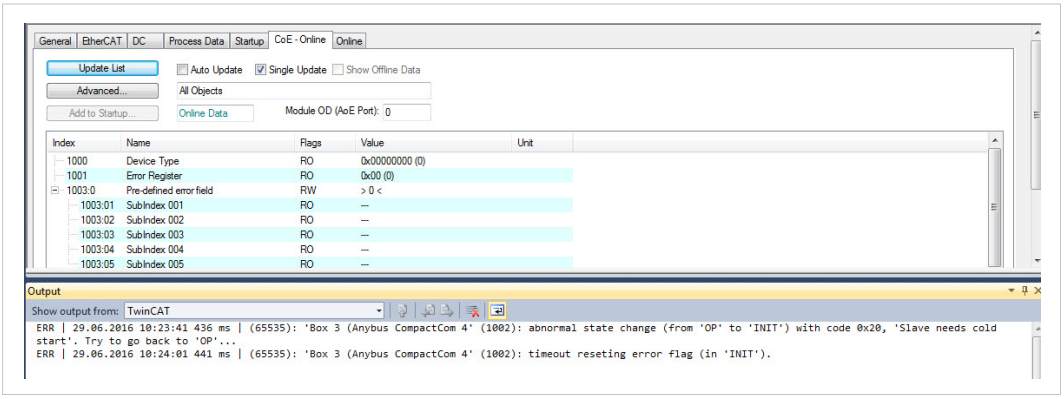

Fig. 27 Disconnection from the network

Box 4126
300 04 Halmstad, Sweden

info@hms.se
SCM-1202–070 1.0.7420 / 2018-02-14