# Migration Guide

1.0

## Migration Galileo 11

**Revision History**

| Revision | Remarks | Date | Sign |
|---|---|---|---|
| 1.0 | Initial version | 2024-10 | PDU |

Eaton Automation GmbH
Spinnereistrasse 8-14
CH-9008 St. Gallen
Tel. +41 71 243 24 24
Fax +41 71 243 24 90
www.eaton.com

© Eaton Automation GmbH. All rights reserved.

# Table of content

# 1 Introduction

**For users coming from Galileo 10.x:**
With Galileo 11, the support for the Windows CE5 resp. Windows Embedded Compact 7 based panel series XV-100 and XV-300 is discontinued. But it is of course almost seamlessly possible to migrate Galileo projects, targeted for these platforms, to the new Linux-based panel series XV-102 and XV-303. The chapter 2 "*Migrating from Galileo 10.x to* Galileo 11" gives an overview of the major changes compared to Galileo 10.

**For users coming from Galileo 8.1.x:**
If the source projects were not yet migrated to Galileo 10 already and still are based on Galileo 8, then there are some more changes to get familiar with first. With starting of Galileo 10, a few new features and concepts were introduced. When importing a Galileo project created with version 8.1.x a lot of conversions and adaptions are already done by Galileo 11. However, some cases can't be handled automatically in regard to ensure completely correctness. In such cases, it's needed that you, the user, verifies some changes manually or do some manual work. Please refer to the chapter 3 "*Migrating from Galileo 8.1.x to Galileo 10.x/11.x*".

## 1.1 Purpose of this document

**For users coming from Galileo 10.x:**
This document gives you a short overview of the major changes caused by the change to a Linux-based systems. It serves as a guide to identify the spots, which might have to be checked manually.

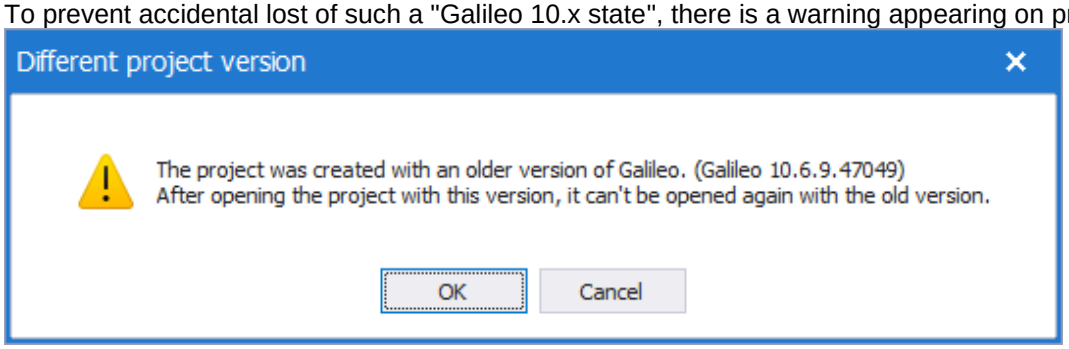
**For users coming from Galileo 8.1.x:**
On one side, this document gives you a short overview of the new concepts compared to Galileo 8. It should give you an idea of what has changed and the reasons behind the changes. Also, it shows you where the data, formerly available in Galileo 8, can now be found.

On the other side, this document serves as a guide which can lead you through the problematic cases, which can occur during an import of a Galileo 8.1.x project into Galileo 11.
You don't need to read this whole document. If the importing process detects some problems, you will see a message in the "Output" view with a reference to a chapter in this document. The message will look like:
`[Issue12] : Some short explanation`
If you see such a message, you can refer to the appropriate chapter (in this example the chapter: "Issue12")

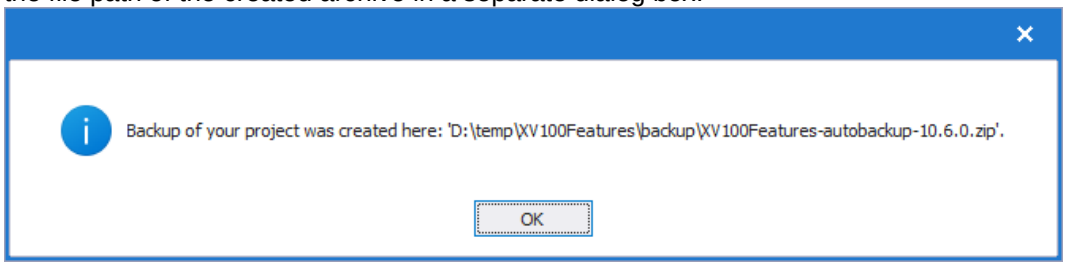
# 2 Migrating from Galileo 10.x to Galileo 11.x

## 2.1 Automatic backup of the last project state before opening

Once a project was opened and saved with Galileo 11.x it can't be used with Galileo 10.x anymore.
To prevent accidental lost of such a "Galileo 10.x state", there is a warning appearing on project opening:



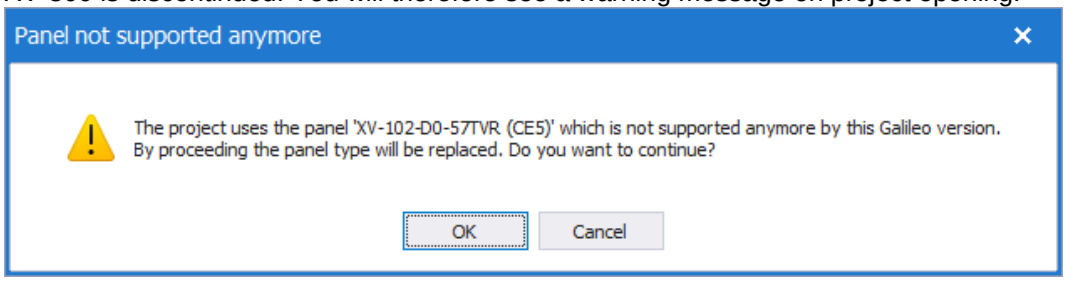

If you decide to continue an automatic backup of the current project will be created first and you will see the file path of the created archive in a separate dialog box:



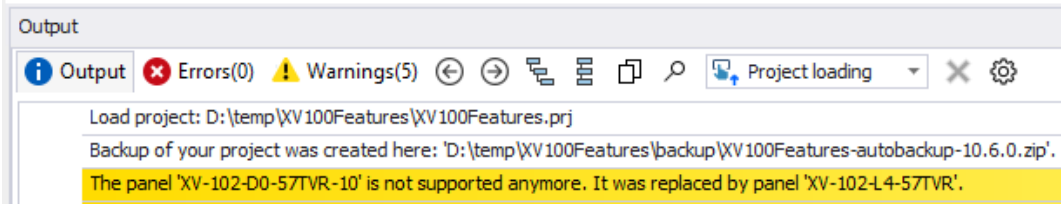

If required, simply unpack the content of the archive into an empty folder and you are able again to open the project with Galileo 10.x

## 2.2 Change from Windows CE5/EC7 to Linux-based panels

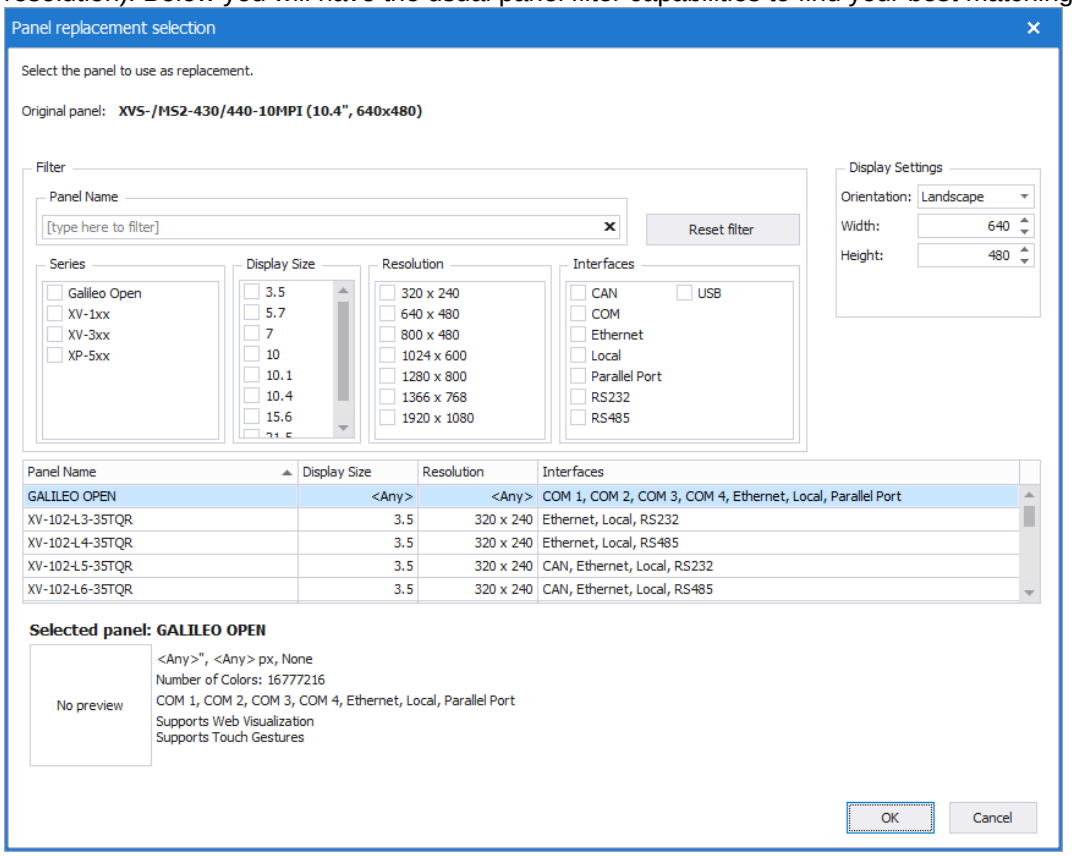
The support for the Windows CE5 resp. Windows Embedded Compact 7 based panel series XV-100 and XV-300 is discontinued. You will therefore see a warning message on project opening:




However, for most of these panels there are successor products available, fully compatible in display size and connectivity possibilities.
So, if you click on "OK", the Galileo 11 project opening procedure tries to identify the matching replacement panel and informs you accordingly in the output window:



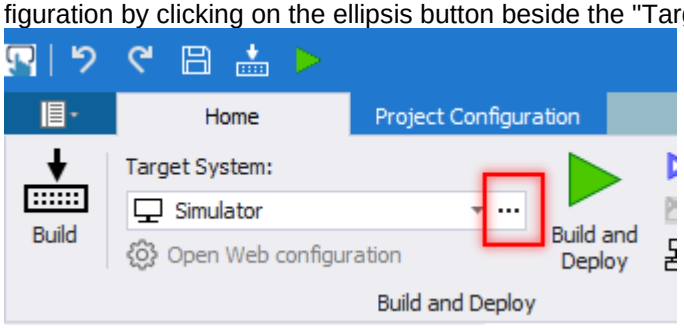

For panels for which not a real compatible replacements does exist, you will get prompted for selecting a replacement manually. On top you will see the project's original panel type (including the display size and resolution). Below you will have the usual panel filter capabilities to find your best matching replacement:



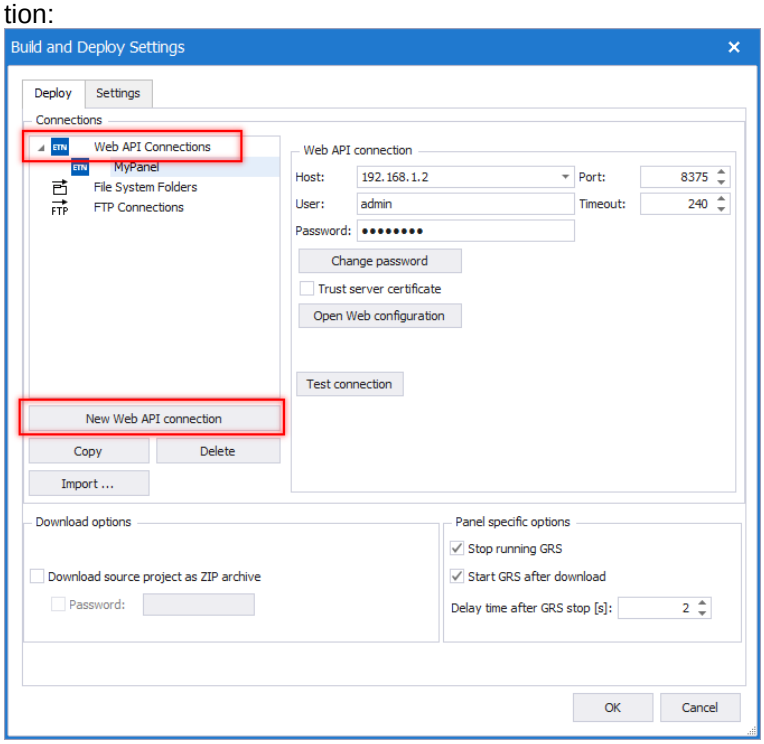
### 2.2.1 Deployment of projects to the panel: Eaton's Web API instead of FTP

On the Linux-based XV-102 and XV-303 panels there is no FTP server available anymore. Instead of that, there is a web service available which supports encrypted communication and data transfer to/from the panel using a well-defined Web API. This is a way more secure way compared to FTP and it offers also a far better data rate.

To configure such a connection to the panel, go to the "Home" ribbon and open the "Target System" configuration by clicking on the ellipsis button beside the "Target System" selection box:




Select the "Web API Connections" entry and click on "New Web API connection" to create a new connection:




Configure the connection accordingly:

- Host: Enter the IP or hostname of the panel
- Port: If no special network configurations are in place, keep it at 8375
- User: Username by default as of now always should be "admin"
- Password: Enter the same password which you used/setup for the Web Config Tool before when setting up the panel. (Remark: If no password was set on the panel until the first connection attempt from Galileo, Galileo will ask you to setup the initial password.)
- Trust server certificate: The panel uses a self-generated certificate to secure the web connection. If you tick this option, the certificate of the host will always be trusted. If this option is not ticked, you will get asked on the connection attempt on how to proceed ("Trust always"/"Trust once"/"Disconnect")

Test the configured parameters by clicking on "Test connection".

The deployment of the project then works identical as with FTP before: Simply select the target in the "Target System" selection box and click on "Build and Deploy".

### 2.2.2 Boot from StorageCard

The Windows CE5 resp. Windows Embedded Compact 7 based panel supported the boot from storage card (SD card) by placing a folder named "OS", containing the operating system file(s), on it plus the folders of a compiled Galileo project. This SD card then could be inserted into any panel and if the option "Boot from StorageCard" was activated, the panel booted up with the appropriate OS and the Galileo project. This SD card could be initialized/written using any Windows based PC because the file systems were compatible.

Due to the change to a Linux-based system, also the file system and its whole operating system architecture changed. This also has an impact on how to setup a SD card, which can be interchanged between panels. Please refer to the separate operating system manual, matching your panel, to get more information about that.

### 2.2.3 Define panel startup behavior: Replacement strategies for former "autoexec.bat" file

The Linux-based panels do not support Windows batch files like "autoexec.bat" and "hmi.bat" due to its concept and architecture. Formerly used "autoexec.bat" and "hmi.bat" files to initialize the panel on startup therefore have to be replaced manually by the user with new concepts.

#### 2.2.3.1 Startup services like FTP and VNC

One common use case of the "autoexec.bat" was to startup the FTP server or the VNC server. This is no longer needed because of following reasons:

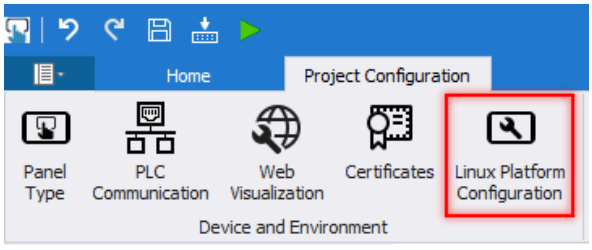
- FTP: There is no FTP server anymore. The web connection (Web API) to transfer projects to the panel is always active and doesn't have to be started up manually on panel reboot. Also, if required, an additional SSH server/connection can be configured to be enabled on panel reboot directly on the panel (refer to the manual of the appropriate panel).
- VNC server: The automatic startup of the VNC server can be configured directly on the panel (refer to the manual of the appropriate panel).

#### 2.2.3.2 Settings configuration

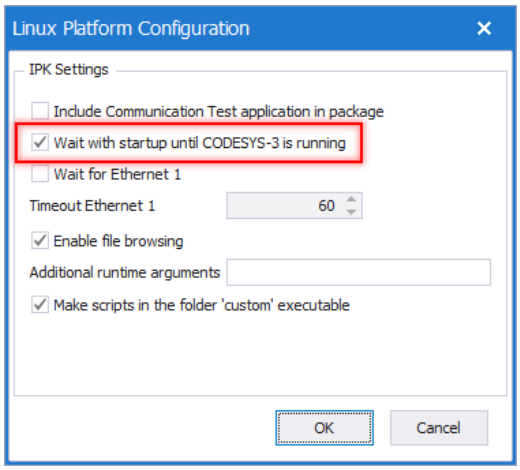
The settings of the panel can be configured using the Config Tool on the panel or the Web Config Tool (https://<panel_ip>:8375). All the settings are permanently stored on the device and will be available also after a reboot.

#### 2.2.3.3 Waiting for local CODESYS startup

In order to wait with the Galileo startup until the local CODESYS application has started up, there is a new option available in the "Linux Platform Configuration" which you will find on the ribbon "Project Configuration":

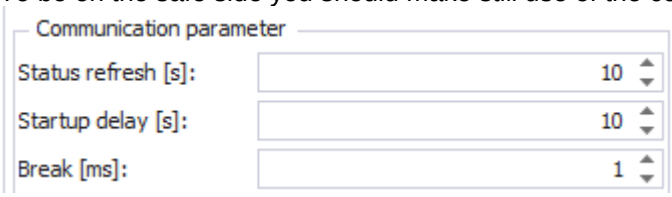



There you can find the option "Wait with startup until CODESYS-3 is running":



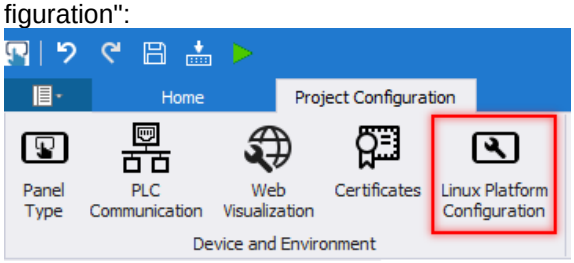

The Galileo application on panel boot will wait until the CODESYS-3 application is started up and ready to operate.

Attention: Even though the CODESYS application signaled a successful startup, this doesn't imply that also the external communication channels are already ready for operation. This state of readiness depends on further conditions and parameters on the CODESYS side, for example if there is a certificate re-generation on startup of CODESYS in place or not.
To be on the safe side you should make still use of the communication setting parameter "Startup delay":

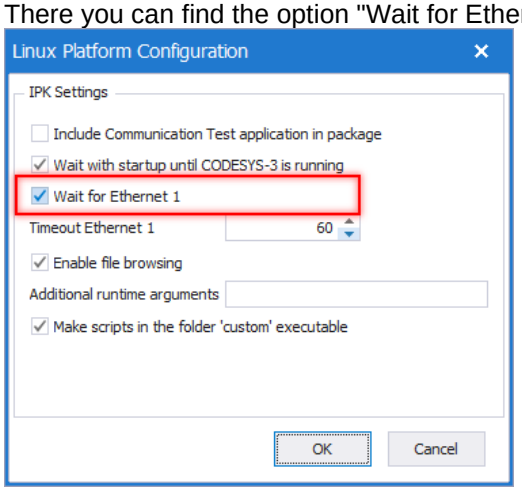



### 2.2.3.4 Waiting for Ethernet connection

In case you have to wait for some attached network infrastructure to become ready for operation, there is a new option available in the "Linux Platform Configuration" which you will find on the ribbon "Project Configuration":




There you can find the option "Wait for Ethernet 1" (including an optional timeout in seconds):



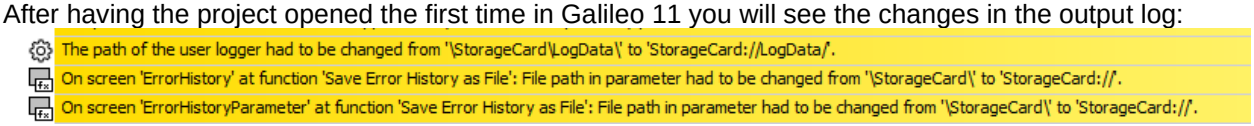
If you have a panel selected which supports 2 ethernet interfaces, you will see also an option "Wait for Ethernet 2".
This allows you to make Galileo wait to startup until it receives a proper Ethernet carrier signal.

#### 2.2.3.5 Custom use cases

If you executed some custom operations in the "autoexec.bat" or "hmi.bat" file, please contact our support team to find a suitable solution together (automation@eaton.com)

### 2.2.4 Changed file paths for storing data

On the Linux-based panels, the file system has changed compared to the Windows CE5 resp. Windows Embedded Compact 7 based panels.
Galileo 11 will try to take care of automatically convert the file and folder paths accordingly.
After having the project opened the first time in Galileo 11 you will see the changes in the output log:




Double-click on an output entry to jump directly to the location where the file/folder path was used and replaced.

For storing data on external media (storage card, USB drive) the file path syntax has changed as follows:

| Windows CE5/WEC7 path prefix | Linux-based panels prefix |
|---|---|
| /StorageCard | StorageCard:// |
| /StorageCard2 | StorageCard2:// |
| … | … |
| /UsbStorage | UsbStorage:// |
| /UsbStorage2 | UsbStorage2:// |
| … | … |

The following areas will be checked on project opening and paths get tried to be converted automatically:

- Graph archive storage location
- Recipe file path
- User Logger file path
- General data directory
- Special functions:
  - Save Error History as File
  - Save System History as File
  - Save History to CSV
  - Save History of Alarm Window to CSV
  - Save History of previous day to CSV
  - Save Parameter List to CSV
  - Image Load
  - Image Select
  - Capture Image (Webcam)
  - Capture Image as (Webcam)

File paths used in scripts at appropriate functions have to be checked manually. The places will be listed in the output window. You can jump to the location by double-clicking the entry.

If you passed a file path from the PLC via a string tag, please check your PLC application accordingly.

### 2.2.5 Case-sensitivity for file names and folder paths

**Please be aware that on Linux-based systems, unlike Windows, the file names and folder paths are case-sensitive!** This means, the file "StorageCard://myfile.txt" is a different one than "StorageCard://MyFile.txt".
Please make sure you use a consistent notation.

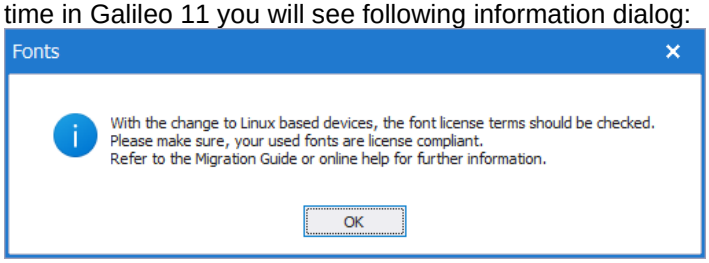
If you used the "ignorefunc.txt" to exclude functions from the User Logger, please ensure the file name is written exactly like this (using lowercase letters only).

Use the "/" sign as path separator (and not the Windows-like "\").

### 2.2.6 Fonts

With the change from a Windows-based system to a Linux-based system you should check the license terms for the font files used in your project, whether they do allow such a usage scenario or not.

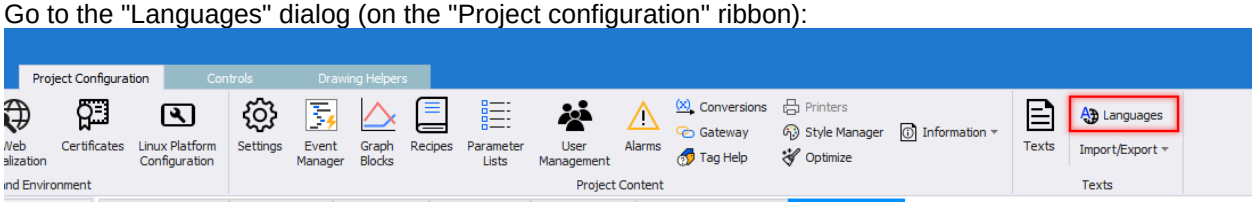
Therefore, after opening a Windows CE5 resp. Windows Embedded Compact 7 targeted project for the first time in Galileo 11 you will see following information dialog:



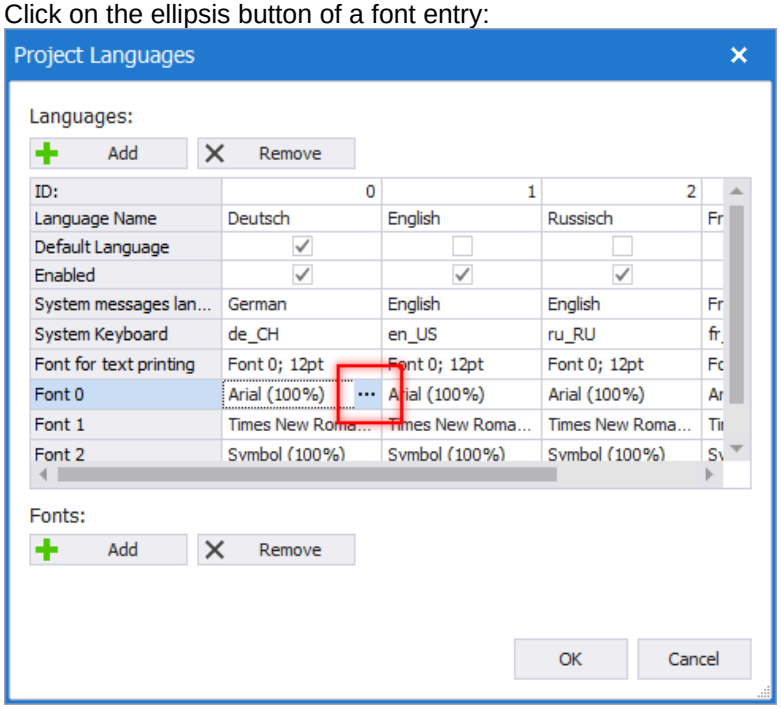

In case you are unsure, Galileo 11 includes some fonts in the setup package which are allowed to be used.

You have to do the changes in the project yourself. There is no automatic conversion available by now.

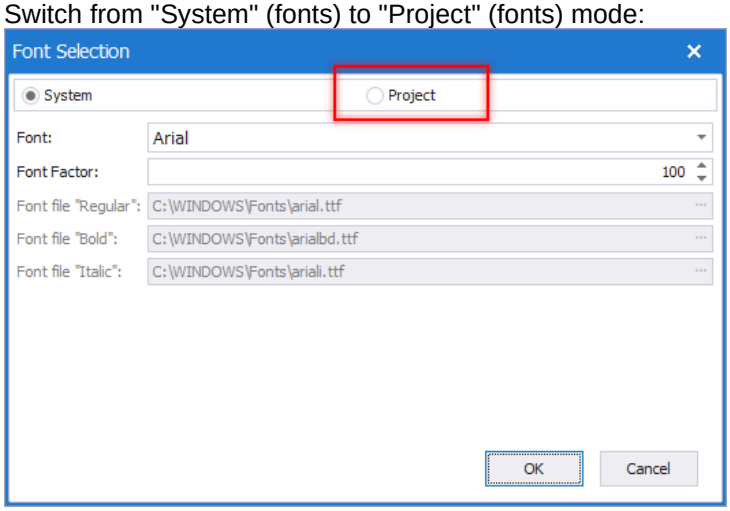
Go to the "Languages" dialog (on the "Project configuration" ribbon):



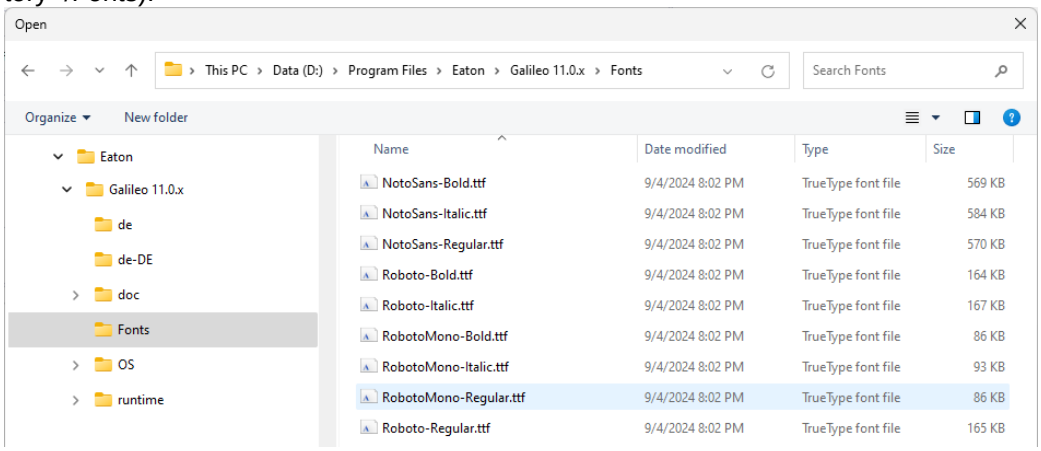

Click on the ellipsis button of a font entry:

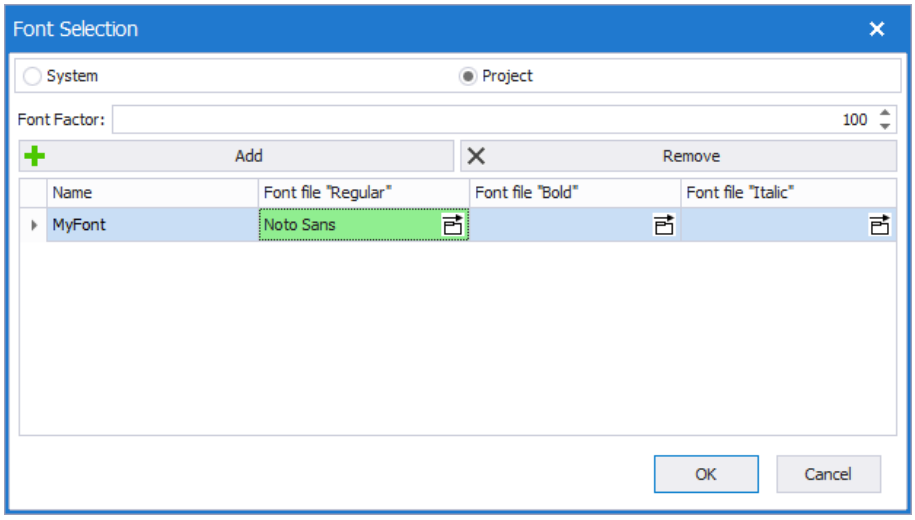


Switch from "System" (fonts) to "Project" (fonts) mode:




Add a font entry and select a font file from the Galileo installation directory (<Galileo_install_directory>/Fonts):




Do the same for the variants "Bold" and "Italic" if required.

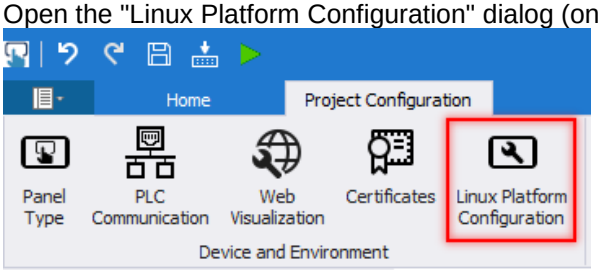
Note: You can find the license files in the same directory as the font files (<Galileo_install_directory>/Fonts) for additional information.

### 2.2.7 Executing external applications or batch files

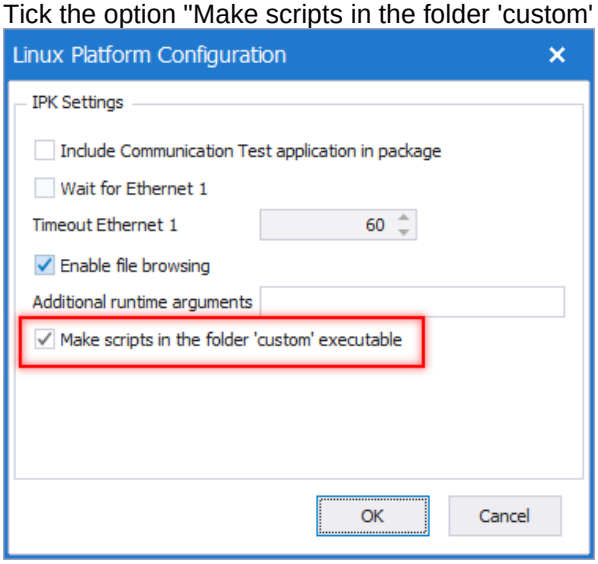
Executables compiled for Windows CE5 resp. Windows Embedded Compact 7 are not compatible with Linux-based panels. They cannot be executed.
If you have used such executable, you have to compile them for the Linux platform again. Please contact our support team in this case (automation@eaton.com) to get additional information.

The same applies for Windows batch files: The batch commands are not compatible with Linux-based systems. They cannot be executed.
If you have used such batch files, you have to rewrite the logic in bash, the shell program and command language for Linux-based systems. It is recommended to put such bash files (file ending ".sh") into the Galileo project folder called "custom".
To be able to execute such scripts on the panel, following additional configuration steps are required for security reasons:

- Open the "Linux Platform Configuration" dialog (on ribbon "Project Configuration"):




- Tick the option "Make scripts in the folder 'custom' executable":



### 2.2.8 Not yet supported functionalities and communication protocols

The following functionalities are not yet supported by Galileo 11.x:

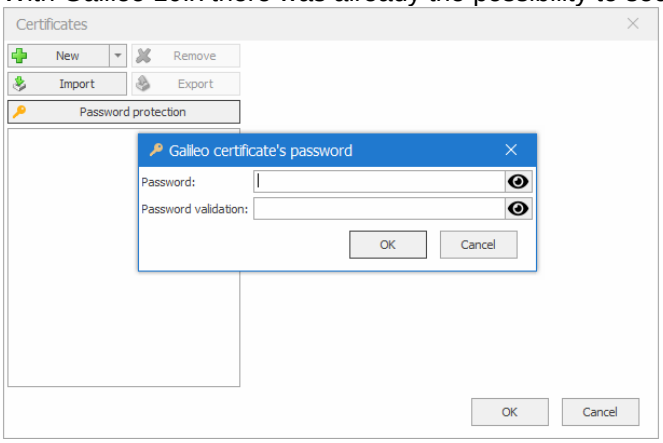
- Printing
- HTML Viewer
- Video Player
- System Tags and special functions:
  - System Tags: RemoteClientActive, RemoteClientInputEnable, RemoteClientInputEnabled
  - Webcam: "Capture Image As" and "Capture Image As Var"

The following communication protocols are not yet supported by Galileo 11.x:

- A. Bradley – Logix – DF1 (will follow in a future release)
- CANopen (will follow in a future release)
- Omron Host-Link C-mode
- Siemens MPI
- Siemens - S7 Profibus Standard Profile
- Sucom-A
- Universal Protocol TP3

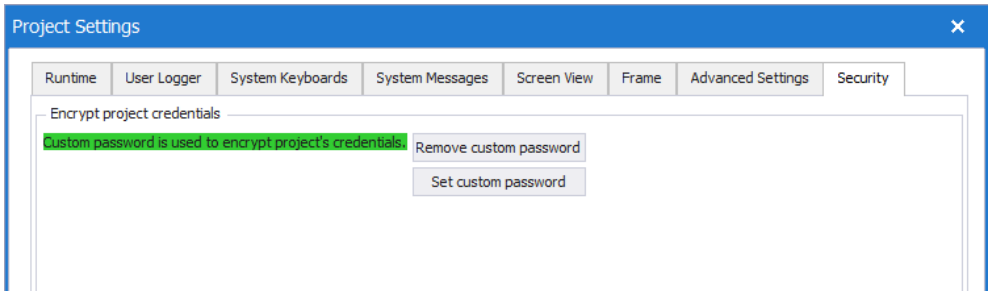
## 2.3 Encrypting credentials of the project

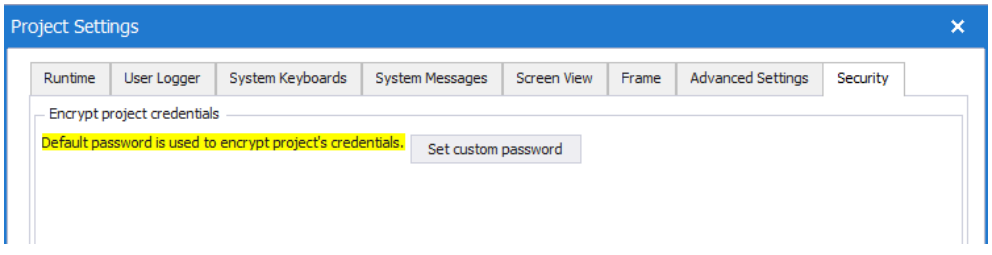
With Galileo 10.x there was already the possibility to secure your certificates with an own password:




Other credentials, like for example FTP login passwords, were encrypted with default mechanisms of Galileo so they would not be stored in clear text.

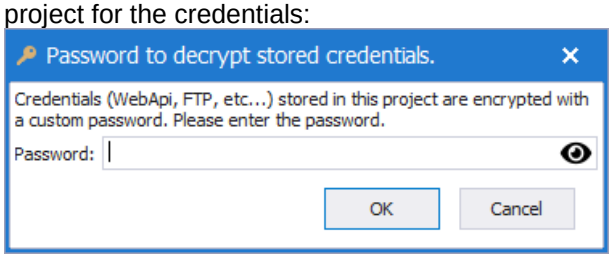
With Galileo 11, the securing of the credentials and certificates is now unified. You can either define your own password to encrypt the data or use the default mechanisms of Galileo. You will find the settings within the "Project Settings" dialog on the new tab "Security":



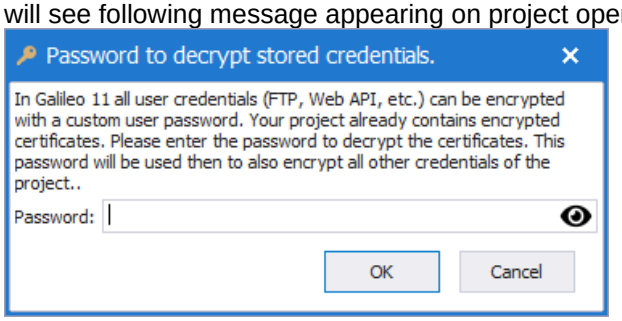



If the "Default password" mechanism is used, Galileo handles the encryption and decryption itself. Be aware, that by using this mechanism it is possible for anybody to open the Galileo project and work the stored credentials. (Note: The credentials are nowhere stored or visible in clear text.)

If a custom password is used to encrypt the project's credentials, you will get asked on opening of the project for the credentials:




If you open a Galileo 10.x project where you have encrypted your certificates with a password already, you will see following message appearing on project opening:




You have to enter your password, which you used for encrypting the certificates. If the correct password was entered, this password will now be used to encrypt all the project credentials. In other words: The mode "Custom Password" (from the screenshots before) is active.

*Note: If you have forgotten your password, you are not able anymore to open the project with the stored credentials. The only option then is to delete the file "credentials.json" in your Galileo project directory. If you do this, on next open of the Galileo project, all the stored credentials get removed from the project which means, you must configure these settings again (e.g., Web API connection passwords). In addition to that, you also have to remove all the certificates from Galileo's Certificate Manager manually and define them again.*

## 2.4 Breaking changes

### 2.4.1 User logger file "ignorefunc.txt"

If you used the file "ignorefunc.txt" to exclude specific functions from the User Logger, this file has to be placed now in directory "custom" instead of the "data" directory.

# 3 Migrating from Galileo 8.1.x to Galileo 10.x/11.x

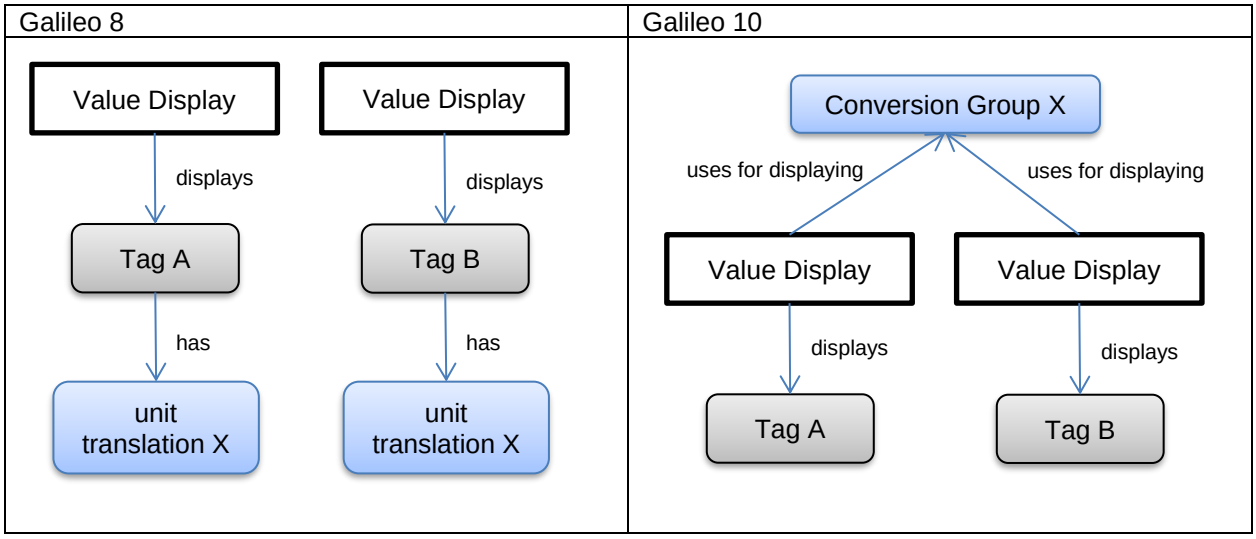
This chapter covers some major conceptual changes in Galileo 10/11 compared to Galileo 8. For further explanations about using these new concepts/features, please take a look at the help documentation.

## 3.1 Conceptual Changes

### 3.1.1 "Unit Translations" replaced by "Conversion Manager"

The formerly known "Unit Translations" in Galileo 8 at each tag are replaced by a new, centralized "Conversion Manager". The "Conversion Manager" gives you the possibility for managing the different conversions in a better, clearer way. It also allows you using a specific conversion for multiple tags. This makes it easier to maintain the conversions – if you need to change a conversion you just have to do it at one place not for each tag. The "Conversion Manager" gives you also more flexibility for displaying values at the same time with different conversions. This flexibility is given through the fact, that the "conversions" are now more a displaying feature rather than directly connected to a tag.

Conceptual comparison between Galileo 8 and Galileo 10:




As you can see, the "Conversion" is now assigned to a displayable control (here a "Value Display"). If you now want to modify the "Conversion Group X", you have to do it only once and not several times.
It would be also possible, to display the "Tag B" with another "Value Display", which is connected to another conversion.

While importing a Galileo 8 project into Galileo 10 the conversion groups and conversions will be automatically created, so the project (here especially: the displaying style and conversion of the values) is fully compatible with the one of Galileo 8.

#### 3.1.1.1 Behavior when used in parameterized sub screens together with type instances

In following situation, an inconsistency could occur which you have to solve on yourself because of the conceptual change.

- Data type with members defined (here in this example: type called "T1", member of type "Word" called "w1")
- Two instances of type "T1" defined, called "Instance1" and "Instance2"
  - Instance1.w1 has a different unit translation than Instance2.w1
- Sub screen defined with a "Value Entry" control. Sub screen takes instances of "T1" as parameter. "T1.w1" assigned to the "Value Entry" control.
- Two sub screens placed on a screen, one with parameter "Instance1", the other with "Instance2"

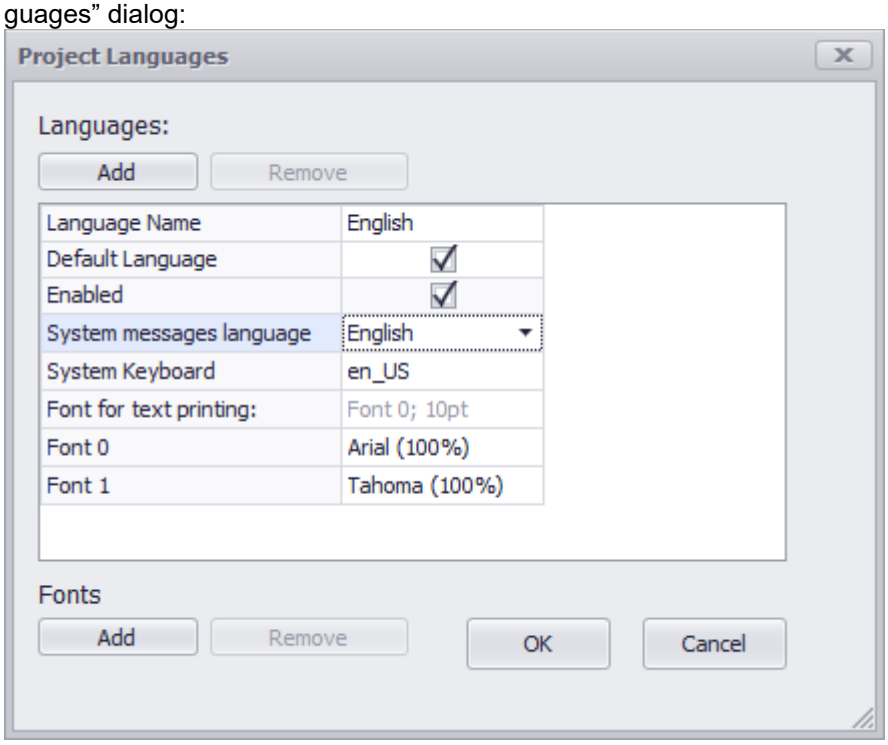
In Galileo 8, each “Value Entry” control now took the unit translation of the assigned tag (via parameters), which means, the displayed values were different.

In Galileo 10, in the sub screen, the “Value Entry” control gets a conversion assigned during the import (the first found conversion of an instance, which in this case would be the conversion created from tag “Instance1.w1”). The two “Value Entry” controls now display the same value.

### 3.1.2 System message languages

In Galileo 8 it was possible to change the system language in the runtime independent from the project language using the special functions “SYS Language …”.
In Galileo 10 you can assign a specific system language to each project language. Therefore, when changing the project language, the system language also changes to the specified one. You don’t have to execute two language changes (project language and system message language).

The coupling of a project language to a system message language can be configured in the “Project Languages” dialog:




### 3.1.3 Function “Last Mask” in combination with user rights

The function “Last Screen” (formerly called “Last Mask”) doesn’t check anymore accessibility rights at all. This concept wasn’t even reliable in Galileo 8.

It concerns following situation:

- Screen “A” contains:
  - Control “Screen Change” to target screen “B”. Its accessibility depends on a bit, which is set by the user rights.
  - Control “Function Key” with function “Last Screen”.
- Screen “B” contains:
  - Control “Screen Change” to target screen “A”.
- The current logged in user has the rights to enter screen “B”. He does that and returns to screen “A”.
- Now the user will be logged out (either manually or automatically)
- The control “Screen Change” to target screen “B” is now disabled, which is correct.

- However, by clicking on the function key with function “Last Screen” assigned, it’s still possible to enter the screen “B”.

To circumvent such situations, you have to write your own script, where you query the access rights and the last called screen. You can use the new introduced “System Tag” called “LastScreenID” to determine the last called screen ID and control the behavior based on this number.

### 3.1.4 Relative path for user logger files

If in Galileo 8 the directory for the user logger has been given as relative path (f.e. “data\”) the folder has been created in the root of the device.

Sample:
Directory: data\

Desktop Computer: User logger folder “\data” created in root of current drive
Panel device: User logger folder “\data” created in root of the device which is in RAM !

From Galileo 10 the folder is created in “General Data Directory” if the given name is a relative path.

### 3.1.5 Communication

In Galileo 8 both the visualization and the communication part are contained in the same global loop. Within Galileo 10 the visualization and the communication part are more separated. This leads to that both parts have their own loop within they can operate.

Due to this stronger separation, the communication sub system has more freedom with respect to when actual IO operations shall be performed. To give the user a better handle to define how often IO operations shall be performed, there exist a new communication parameter called *minimum cycle time*. This parameter is abbreviated in the “PLC Selection & Configuration” dialog as “Min. Cycle Time”. For certain communications also the “Break” communication parameter still exists.

The minimum cycle time $T_{ms}$ defines the interval time after which a new polling operation is started. In Figure 1 this behavior is shown in the case that the time required to read the tags values $T_R$ is smaller than the minimum cycle time. The sleep time $T_S$ is defined as the difference between the minimum cycle time and the time to read the tags, i.e. $T_S = T_{ms} - T_R$. Is the sleep time positive, as shown in Figure 1, then for the time of $T_S$ there will be no IO operations performed, hence the name sleep time.

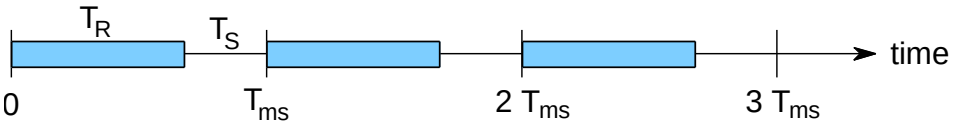



**Figure 1: Representation of the minimum cycle time ($T_{ms}$). The blue bars represent the time required to perform the read operation ($T_R$). The sleep time is denoted as $T_S$.**

It is however possible that TS is larger than Tms. In such a case there will be no additional sleep time. I.e. after finishing the read operation, the next read operation will start immediately.

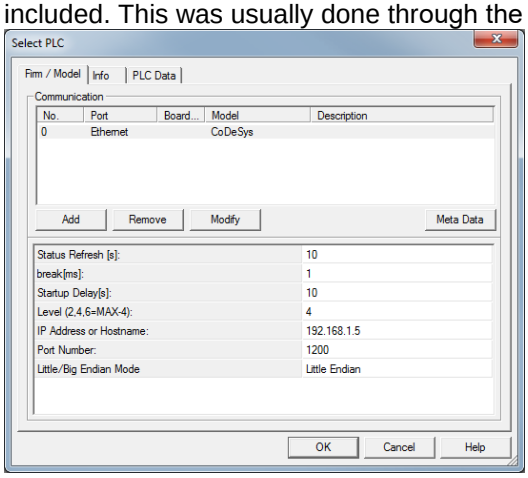
The default value of the minimum cycle time is set to 50 ms. In case GRS creates too much load on the PLC due to sending too many requests within a given time to the PLC, it might be required to increase this value.

For certain communications also the “Break” parameter exists. It is independent from the minimum cycle time parameter. In contrast to the minimum cycle time, the break parameter adds a sleep period after a read cycle, regardless of how long the read cycle actually took. Sometimes this may be the better parameter to reduce the load, put on a PLC due to the IO request sent from GRS.

Both parameters, the break time and the minimum cycle time, can be combined, resulting in the behavior that first there will be a sleep period applied, due to the minimum cycle time, if the read operations did not take longer than the minimum cycle time. And afterwards the break time will cause an additional sleep period of the defined size.

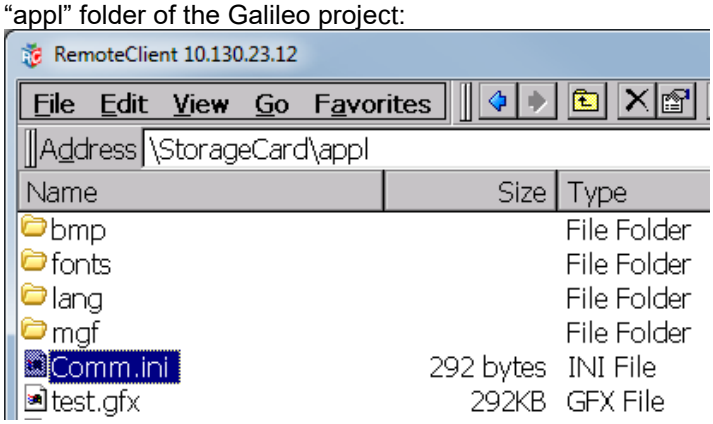
### 3.1.6 Replacement for autoexec.reg

Galileo 8.x offered the possibility to create an autoexec.reg file, where the communication parameters were included. This was usually done through the “Meta Data” button in the PLC selection dialog:




Having this autoexec.reg file, it was easy to compile the project once, download it to multiple panels and adapt just the autoexec.reg file on the panels for e.g. changing the IP address of a PLC.

Galileo 10 doesn’t support that concept of an autoexec.reg file anymore. But there is another way to achieve this.
When compiling a Galileo 10 project and downloading it to a panel, there exists a file “Comm.ini” in the “appl” folder of the Galileo project:




In this file, the communication settings of all PLCs are stored. The content of such a file could look like this:

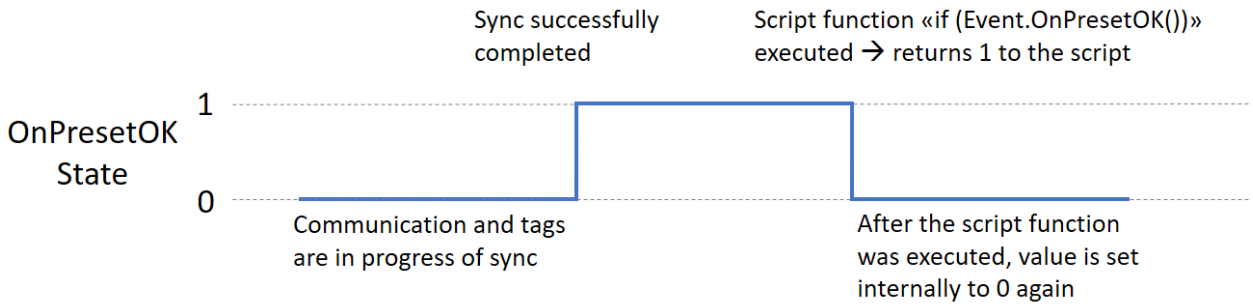
```
; CODESYS V2
[Communication0]
Type=16
Group=1
Model=1
Board=0
Endianness=0
TwistedOrder=0
Alignment=5
SlotOrCom=0
LifeTimeCycle=10
Parameter1=1200          ; Port Number
Parameter4=4             ; Level (2,4)
String1=192.168.1.5           ; IP Address or Hostname
Break=1
StartupDelay=10
MinCycleTime=50
```

You now can change any setting in this file and after restarting the Galileo Runtime System, the settings will be applied.
The advantages of this solution compared to the former autoexec.reg file are:

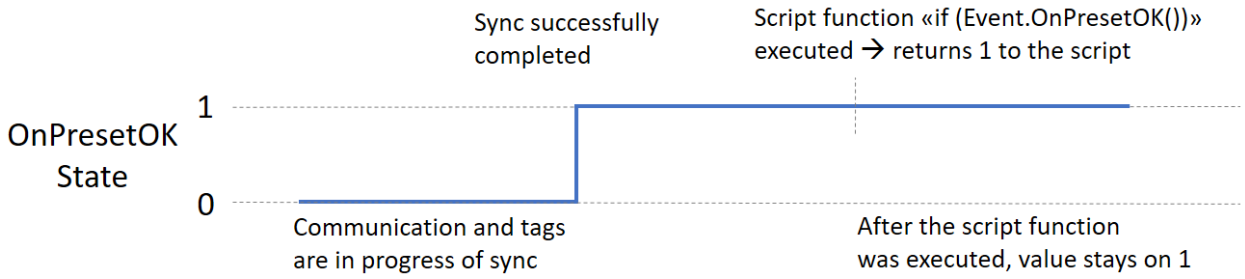
- All communication parameters are available for changing
- Parameters can be defined per project and not per panel
- A change of a parameter just needs a restart of the Galileo Runtime System and not a complete system reboot

### 3.1.7 OnPresetOK

The script function “OnPresetOK” is used for determining the state, when all the communications and tags are successfully set up.
On Galileo 8, the “OnPresetOK” function call was done as an edge. Meaning, once the “OnPresetOK” was called and returned “1”, the internal state of “OnPresetOK” was reset to “0”.




In Galileo 10 this behavior is changed, so once the “OnPresetOK” state is set, the state remains on “1”:




In certain circumstances, this can lead to problems to project behavior. Especially when using a code like this one (in a loop script):

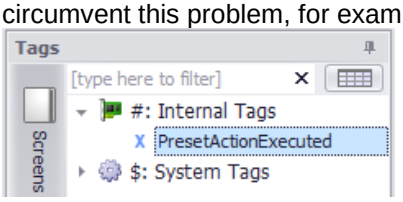
Loop script:

```
if (Event.OnPresetOK())

    Screen.ScreenChange(StartScreen);

endif
```

In Galileo 8 this code forced a single screen change to “StartScreen” because “OnPresetOK” after the first call always returned “0”.
In Galileo 10 this code would lead to a continuously screen change to “StartScreen” again and again. To circumvent this problem, for example an internal tag could be used as a helper variable:




Loop script:

```
if (Event.OnPresetOK() AND NOT PresetActionExecuted)

    Screen.ScreenChange(Screen);
    PresetActionExecuted := 1;

endif
```

### 3.1.8 Transparent standard screens

Standard screens in Galileo 10, compared to Galileo 8.1.x, are no longer allowed to have the background set to “Transparent”. There were a few common known use-cases for using “Transparent” as the screen background. Following sub chapters explain each use-case and how they can be handled in Galileo 10 in a more proper way.

#### 3.1.8.1 Transparent screen as workaround to execute scripts from PLC

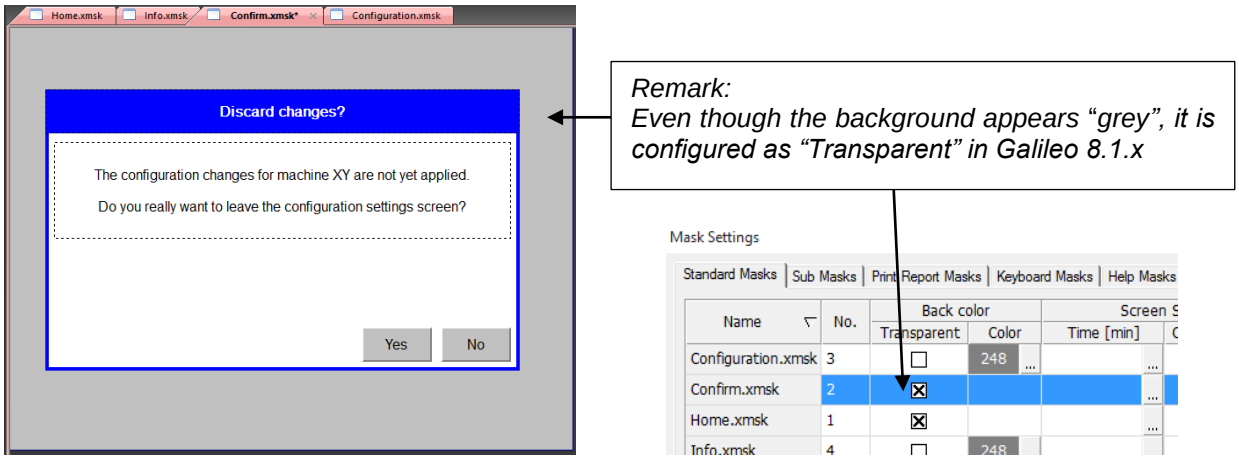
A common use case to use a completely transparent screen in Galileo 8 was the following situation:

- A Galileo script (let’s name it “init”) should be called from the PLC
- Galileo 8.1.x didn’t support calling scripts from the PLC but allowed executing a screen change from the PLC
- An empty screen with background set to “Transparent” was created and the script “init” was assigned as an entry script.
- The screen change to this empty, transparent screen was performed from the PLC. The PLC (or the Galileo script) handled the jumping back to the initial screen after the change (and therefore the entry script execution) were done.

Galileo 10 now offers the possibility to execute Galileo scripts directly from the PLC. Please take a look at the script property “Use script ID” and refer to the help content of the special function “Call Script Function”.

#### 3.1.8.2 Use transparency to build a kind of “Dialog”

Another use case for transparent standard screens was to use them as modal dialogs (window on top of the application, interaction with the application is not possible until the window gets closed by the user).

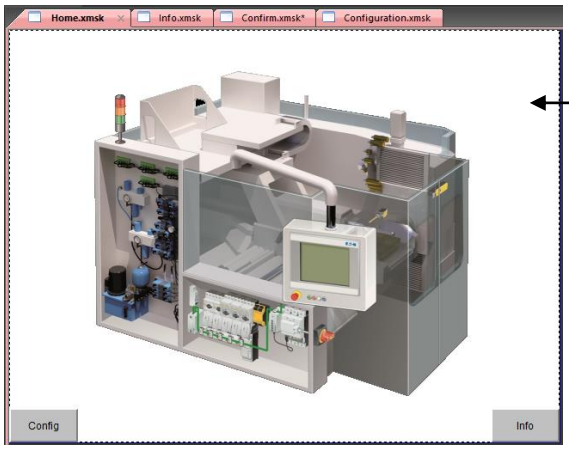



Galileo 10.4 now offers such “Dialog” behavior in a cleaner way. There’s the new control “Dialog button” available which is intended for such use cases.

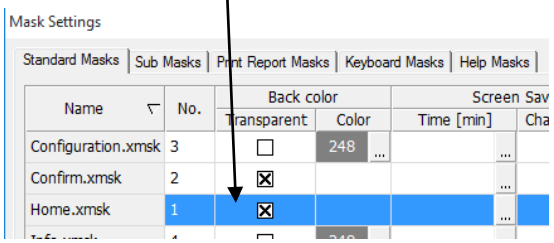
On import of Galileo 8.1.x projects, a wizard will help you to convert screens with transparent background into modal dialogs. To explain the way of conversion better, please read chapter 3.1.8.4.

#### 3.1.8.3 Transparent background by accident

It could happen, that the background of a screen was set accidentally to “Transparent” and it was never noticed on runtime because for example, an image or a control (e.g. filled rectangle) placed as background filled the whole screen.




*Remark:*
*The image size is equal to the screen size. The background is set to "Transparent" but had never an affect during runtime.*



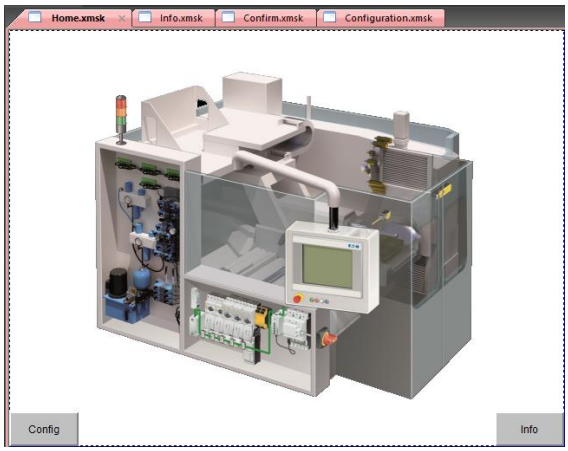

On import of Galileo 8.1.x projects, a wizard will help you to decide whether such screens with transparent background should be converted to modal dialogs or stay as "Standard Screens" with a default background color set. To explain the way of conversion better, please read chapter 3.1.8.4.

### 3.1.8.4 Import wizard for “Modal Dialogs”

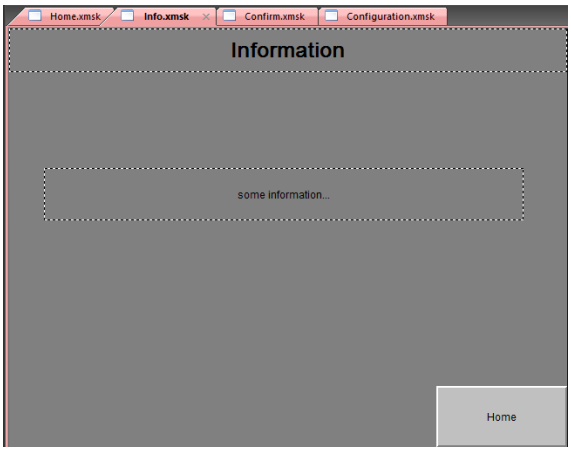
To illustrate the working principles and the effects of the import wizard better, we take following Galileo 8.1.x project as an example.

We have a project containing 4 screens:

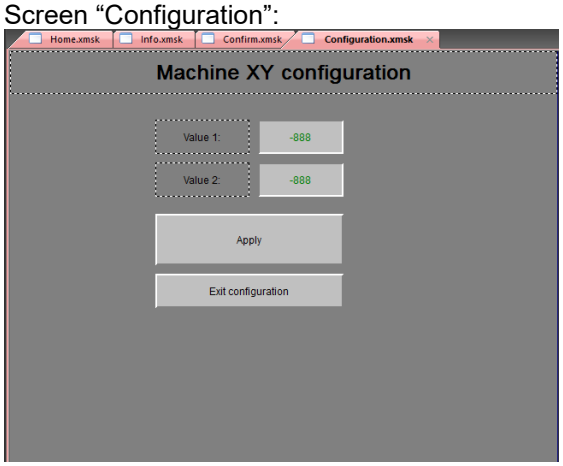
Screen “Home”:



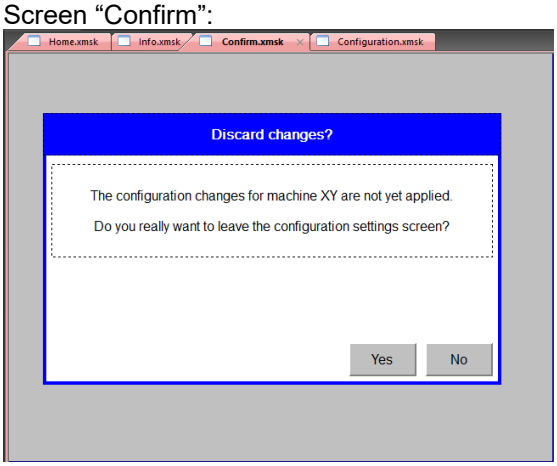

Screen “Info”:



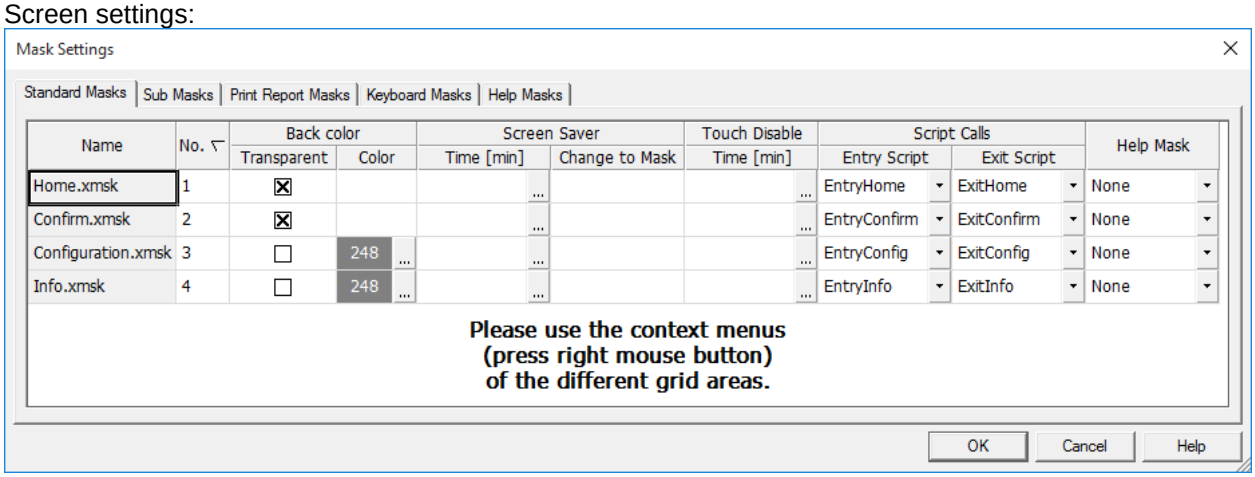

Screen “Configuration”:




Screen “Confirm”:

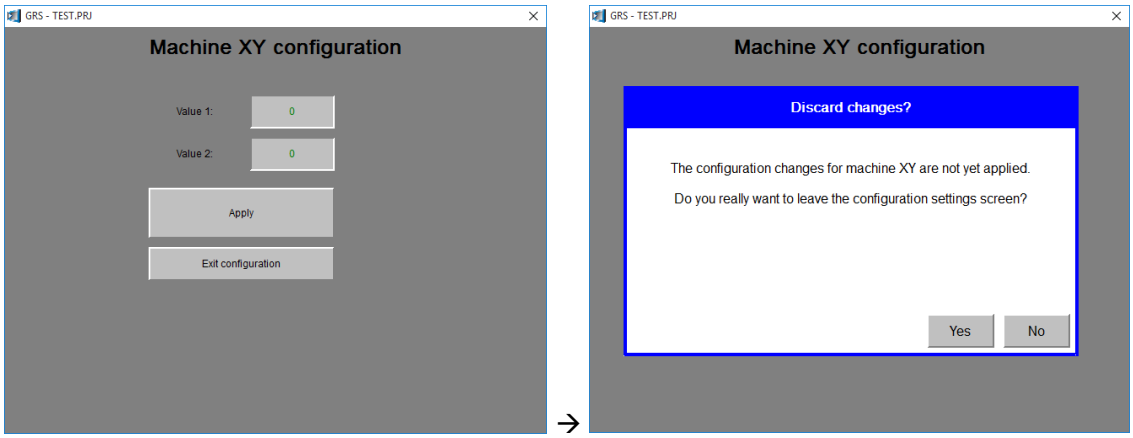



Screen settings:

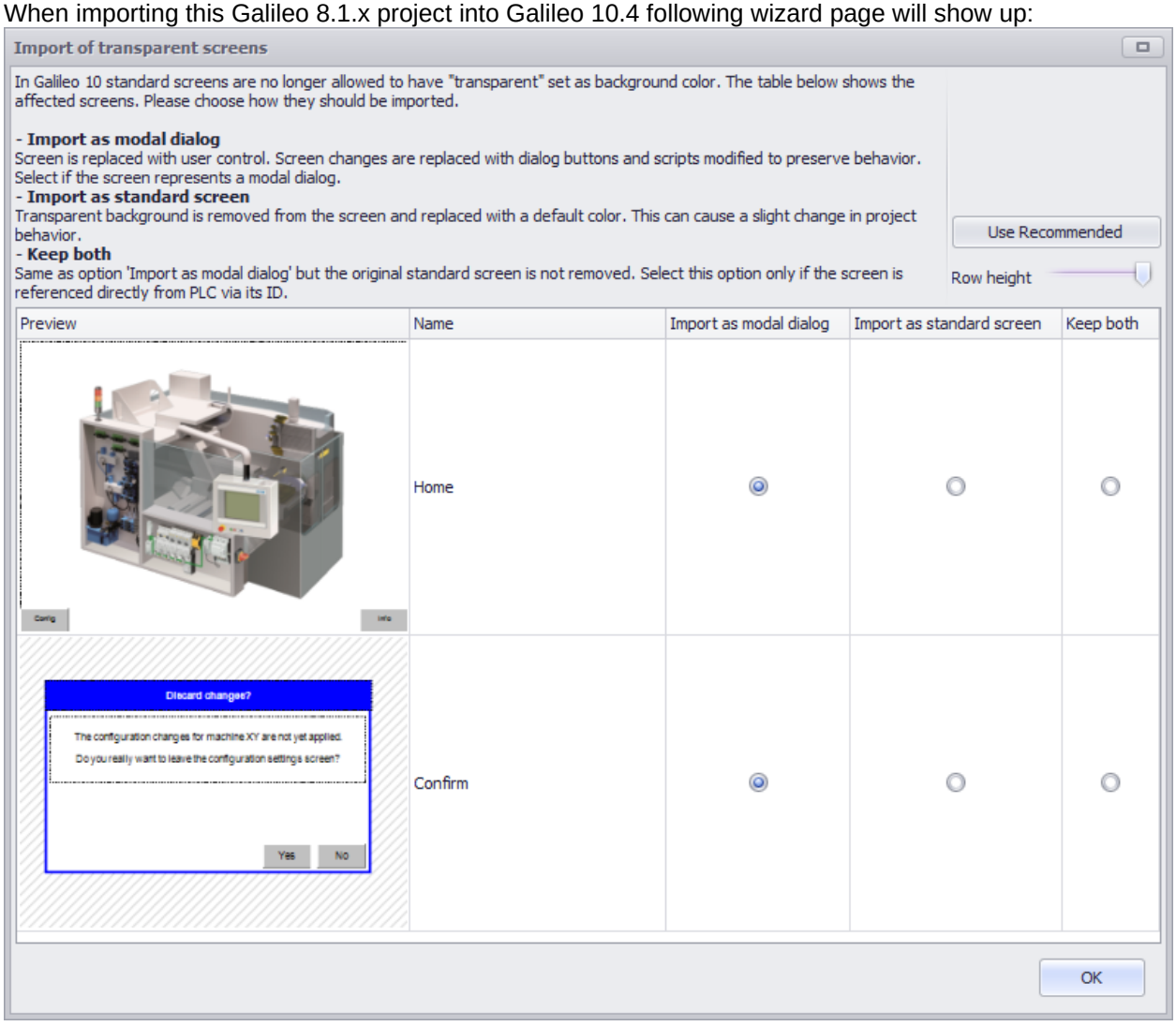



(“Home” and “Confirm” use a transparent background. Note also the assigned “Entry” and “Exit” scripts.)

On runtime, when executing “Exit configuration” on the “Configuration” screen without applying the changes before, the “Confirm” screen appears and overlays the screen in a kind of “Dialog” way:




When importing this Galileo 8.1.x project into Galileo 10.4 following wizard page will show up:




All screens with a former “transparent” background will be listed here. For every screen, it must be decided how they should be imported into Galileo 10.4.

The options are:

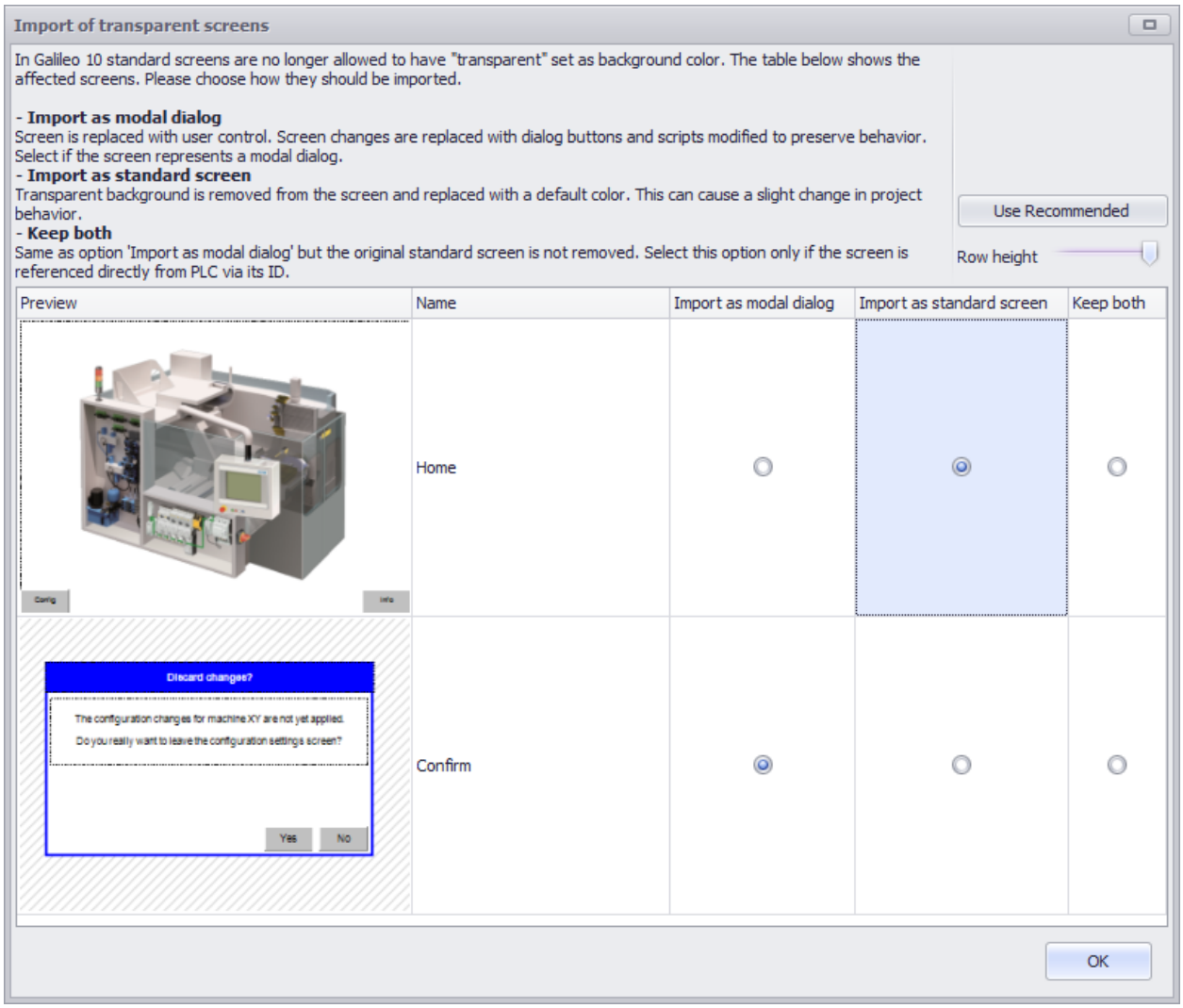
- Import as modal dialog:
  The screen will be replaced with a “User Control”, containing all the same content as the screen. If in Galileo 8.1.x a “Screen Change” was used to call the screen, this control will be replaced with the new control “Dialog Button” having the created “User Control”. The involved scripts will get reassigned so the behavior on runtime stays the same. This option should be chosen for every

screen representing a kind of “Dialog” AND was never directly called from a PLC for a screen change.

- Import as standard screen:
  The “transparent” background of the screen will be replaced with a default color. This can cause a slight change in project behavior in case there isn’t a control placed on the screen which fully covers the screen.
- Keep both:
  Imports the screen as both, “User Control” and “Standard Screen”. However, all the places where the screen was referenced in the project get replaced by “Dialog button” controls (same as option “Import as modal dialog”). The separate copy as “Standard Screen” remains in the project in case a screen change was performed from the PLC to this screen (because a screen change to a “User Control” is not possible).

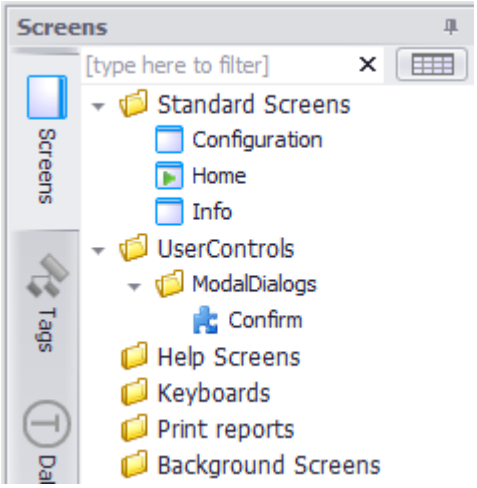
For our example we would choose following options:

- Import screen “Home” as “Standard Screen” because we know, the background was accidentally set to “transparent” in Galileo 8.1.x and the screen is always intended to be used as a “Standard Screen”
- Import screen “Confirm” as “Modal Dialog” because we know, we used it as a kind of “Dialog” in Galileo 8.1.x

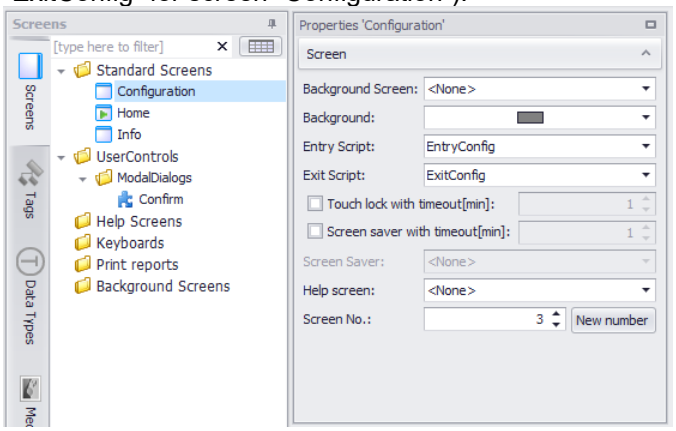



What happens after clicking “OK” is following:

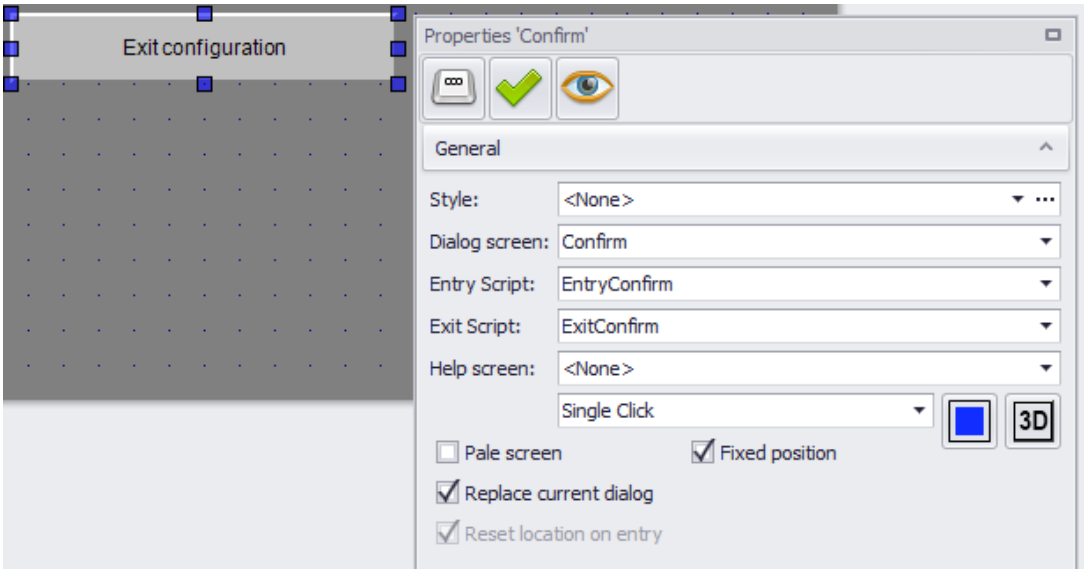
- The screens “Home”, “Info” and “Configuration” are available as “Standard Screens”. The former screen “Confirm” is now available as a “User Control”:




- The “Entry” and “Exit” scripts of the “Standard Screens” remain assigned (e.g. “EntryConfig” and “ExitConfig” for screen “Configuration”):




- The “User Control” “Confirm” is no longer able to have “Entry” and “Exit” scripts assigned. In order to keep the runtime behavior of Galileo 8.1.x the same (which had these scripts assigned), the scripts are now assigned to the “Dialog Button” control, which calls the “User Control” “Confirm”:



#### 3.1.8.5 Entry and Exit scripts and “Modal Dialogs”

To illustrate the working principles and the effects on the entry and exit scripts let’s consider following example.

We have screen “Configuration”.
Entry script: “EntryConfig”
Exit script: “ExitConfig”

Screen with transparent background called “Confirm”.
Entry script: “EntryConfirm”
Exit script: “ExitConfirm”

We have following situation which represents opening and closing of a dialog:

1. Open screen “Configuration”
2. Open screen “Confirm”
3. Return back to screen “Configuration” (close the “Confirm” dialog)

In Galileo 8 the following scripts will be called:

1. “EntryConfig”
2. “ExitConfig”
3. “EntryConfirm”
4. “ExitConfirm”
5. “EntryConfig”

In Galileo 10 after the conversion of screen “Confirm” to a modal dialog the following scripts will be called:

1. “EntryConfig”
2. “EntryConfirm”
3. “ExitConfirm”

This is because in Galileo 10 opening and closing of dialog is not a screen change anymore. In some situations scripts “EntryConfig” a “ExitConfig” may not be needed anymore or they may need to be revised.

### 3.1.9 CANopen communications

Galileo 8 offered multiple "CAN Open" communications if the selected panel supported a CAN interface. There were so called "polling" (CAN SDO) and "event" (CAN PDO) variations available. So, if both types, SDO and PDO, had to be handled, two communications had to be added to the Galileo project.

Galileo 10 supports, starting with version 10.4.5, also a CANopen communication. But there are some major differences:

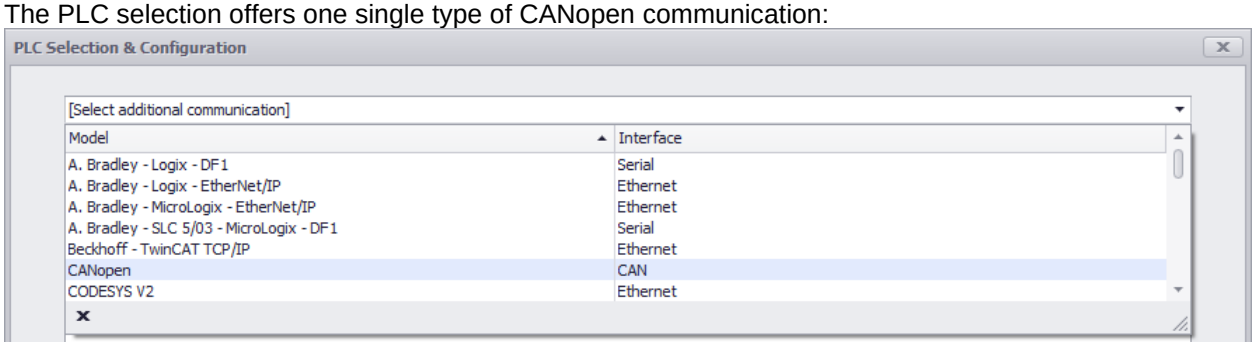
- **It is NOT possible to have Galileo and another application using the CAN interface (e.g. a CODESYS application which also operates on some CAN nodes) running on the same device. The first starting application will occupy the interface and the second (and following) application will report an error.**
- The Galileo 10 CANopen communication supports both types, SDO and PDO, on the same communication
- Only 1 CANopen communication is supported per Galileo project

These differences/limitations, under circumstances, have certain impact on how Galileo 8 projects are imported/converted when opening with Galileo 10. Especially on the tag addressing, communication settings and possible used functions of type "Communication x On/Off". The chapter xxx gives an overview of possible usage scenarios and what you have to consider in each case.

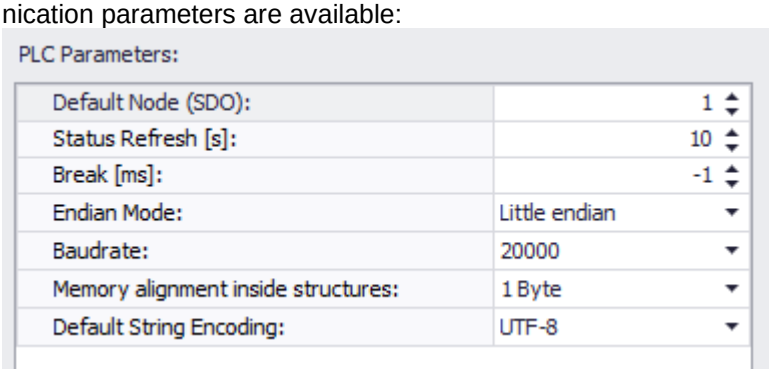
The chapter 3.1.9.1 shows all the general changes to provide a quick overview.

#### 3.1.9.1 General changes

The PLC selection offers one single type of CANopen communication:

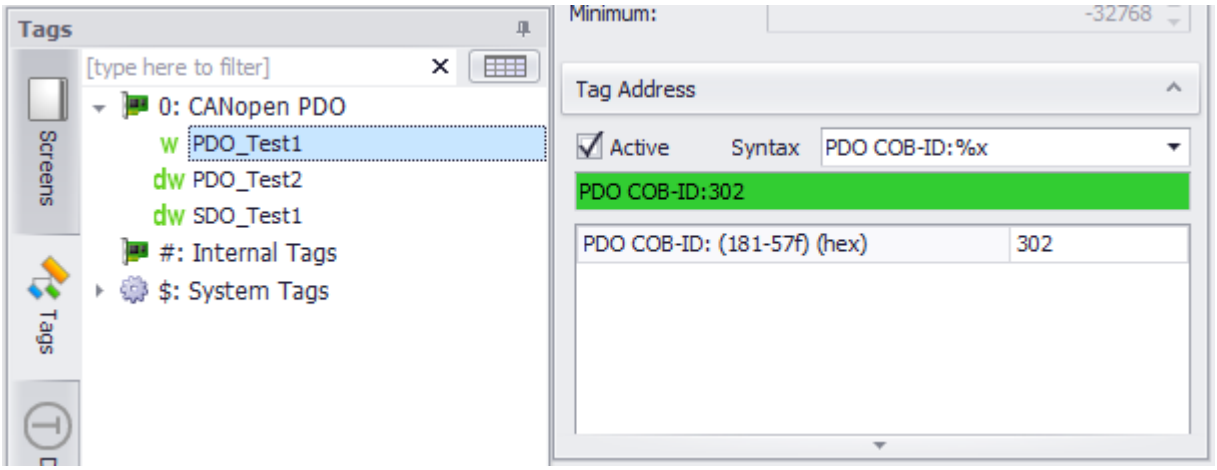



The CANopen communication provides (and combines) the functionality of the Galileo 8 communications "CAN Open – polling" and "CAN Open – event". Because of this combining, all relevant CANopen communication parameters are available:



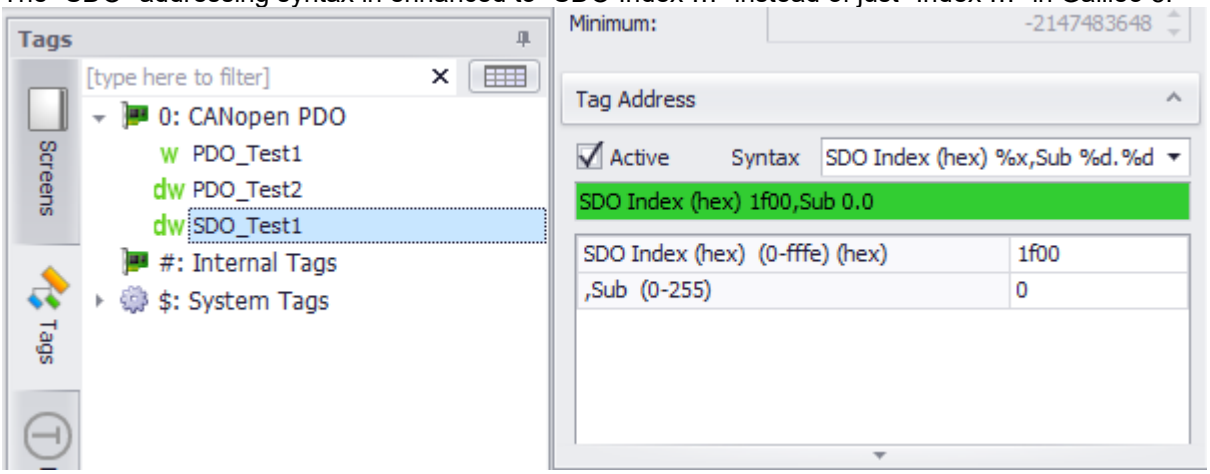

Not all parameters take effect on the "PDO" communication. For example, the "Default Node" and also the "Break" time will be ignored.

The "PDO" addressing syntax is enhanced to "PDO COB-ID:%x" instead of just "%x" in Galileo 8:



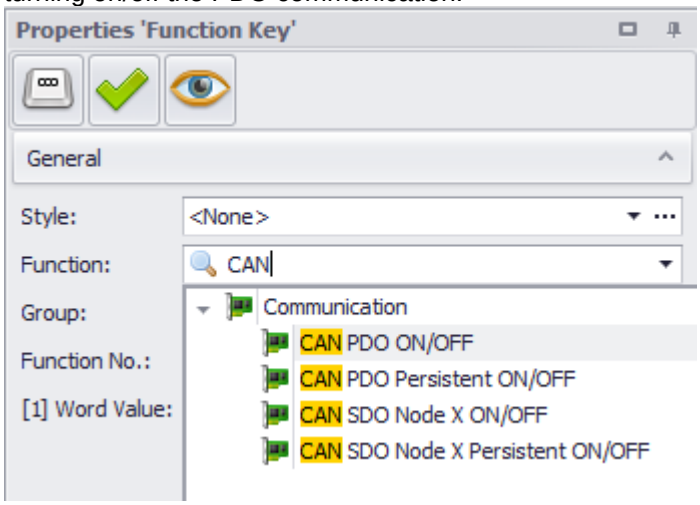

The entering of the address is identical to Galileo 8. Simply enter the COB-ID you want to operate on.

The "SDO" addressing syntax in enhanced to "SDO Index …" instead of just "Index …" in Galileo 8:




The entering of the address is identical to Galileo 8. Simply enter the SDO Index number and other required information.

There are new functions available for turning the communication to single SDO nodes on/off as well as turning on/off the PDO communication:




These functions can be used to enable/disable communication to certain parts of the machine during operation time. The "Persistent" versions of the functions remain the state after restarting the Galileo application.

For tags of type "String", in Galileo 8 it was possible to select the type of "String Termination". This doesn't make a lot of sense, because CANopen doesn't knows the concept of "Strings". It is just a sequence of

bytes which, under circumstances, are interpreted as a character array on the HMI side. Therefore, it is no longer possible to select a string termination.

#### 3.1.9.2 Usage scenarios

The sub-chapters give an overview of the different usage scenarios and what actions/changes were done during import of the project. They also give a hint of possible compatibility problems or which settings have to be checked manually.

##### 3.2.1.9.1 Galileo 8 handles only PDOs

Use case:
- Galileo 8 project contains only one "CAN Open – event" communication which is used to handle CAN PDO communication
- The Galileo 8 project can contain other communications (e.g. CODESYS-2) as well

Actions/Changes during import:
- Communication settings are taken over identically
- Tag addresses (PDO COB-IDs) are taken over identically
- Special functions and script functions like "Comm. on Connection 0 ON/OFF" are taken over identically

Settings to check manually:
- Nothing special. A quick check of the settings and addresses should be performed.

Possible known compatibility problems:
- None

##### 3.2.1.9.2 Galileo 8 contains only 1 SDO communication

Use case:
- Galileo 8 project contains only one "CAN Open – polling" communication which is used to handle CAN SDO communication
- The Galileo 8 project can contain other communications (e.g. CODESYS-2) as well

Actions/Changes during import:
- Communication settings are taken over identically
- Tag addresses (SDO Index, Subindex) are taken over identically
- Special functions and script functions like "Comm. on Connection 0 ON/OFF" are taken over identically

Settings to check manually:
- Nothing special. A quick check of the settings and addresses should be performed.

Possible known compatibility problems:
- None

##### 3.2.1.9.3 Galileo 8 handles one PDO and one SDO communication

Use case:
- Galileo 8 project contains one "CAN Open – event" and one "CAN Open – polling" communication
- The Galileo 8 project can contain other communications (e.g. CODESYS-2) as well

Actions/Changes during the import:
- The two CANopen communications are unified in a single CANopen communication
- Communication settings are in general taken
- Tag addresses are taken over identically

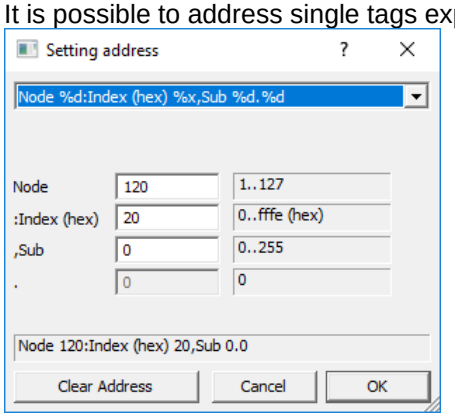
- Special functions and script functions like "Comm. on Connection 0 ON/OFF" are replaced with appropriate functions "CAN SDO Node X On/Off" and/or "CAN PDO On/Off"

Settings to check manually:

- Because the communications could have had different "Baud rate" and "Status Refresh" settings in Galileo 8, the settings have to be checked. Only the settings from the first found communication in the project are applied.

Possible known compatibility problems:

- Turning on/off communication to multiple CAN SDO Nodes with one function call is no longer possible. Explanation
  - In Galileo the CAN SDO communication has a setting for "Default Node" (e.g. Node 5)
  - Any tag addressed on this communication by default operates on this node
  - It is possible to address single tags explicitly on another node:




  - When executing the function "Comm. on Connection 0 ON/OFF", it affected the communication to CAN Node 5 (Default Node) as well as CAN Node 120 (specific tag)
  - Any existing "Comm. on Connection 0 ON/OFF" function on import just gets replaced by the function "CAN SDO Node 5 On/Off" (the default node ID).

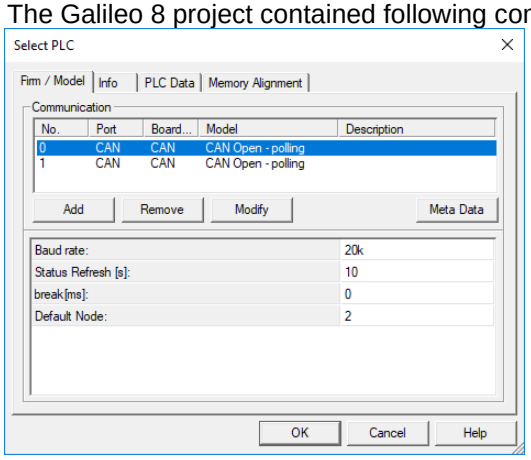
### 3.2.1.9.4 Galileo 8 handles multiple SDO communications

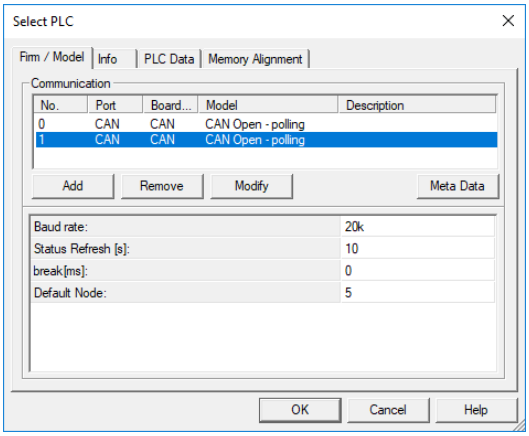
Use case:

- Galileo 8 project contains multiple "CAN Open – polling" communications
- The Galileo 8 project can contain other communications (e.g. CODESYS-2, CAN PDO) as well

Actions/Changes during the import:

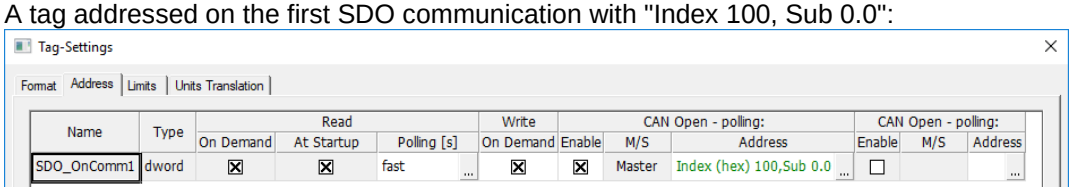
- The multiple CANopen communications are unified in a single CANopen communication
- Communication settings are in general taken
- Tag addresses will be adopted where needed:
  - Because the different SDO communications had different "Default Node" settings but we offer now just on "Default Node" setting, it is needed to adopt tags to contain the Node ID as part of the address.
  - Example:
    The Galileo 8 project contained following communication definitions:



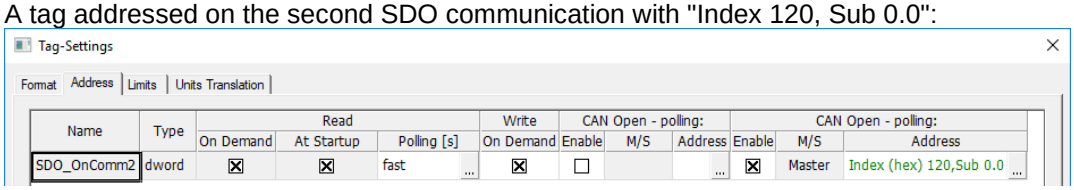
One CAN SDO communication with "Default Node" 2, one with 5

A tag addressed on the first SDO communication with "Index 100, Sub 0.0":



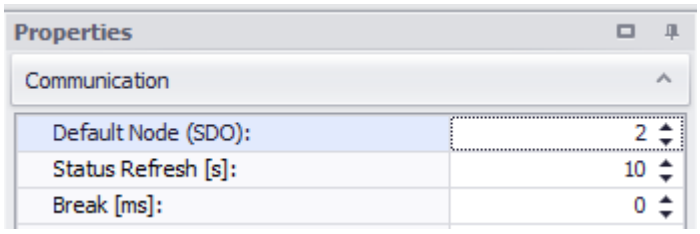

So, in the end, it operates on: CAN Node 2, Index 100, Sub 0.0

A tag addressed on the second SDO communication with "Index 120, Sub 0.0":

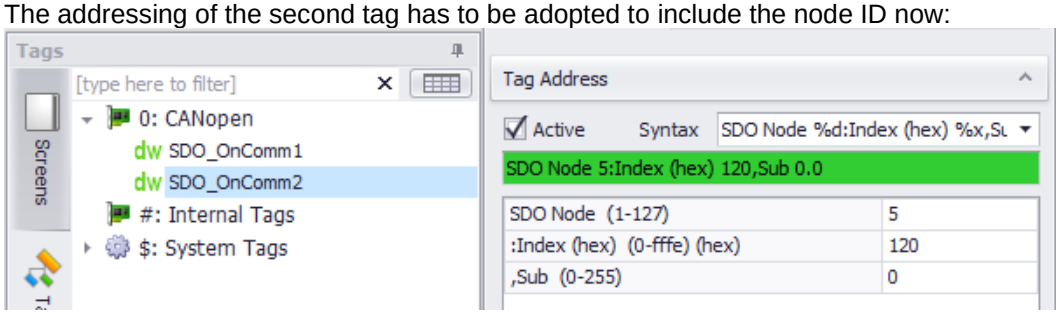



So, in the end, it operates on: CAN Node 5, Index 120, Sub 0.0

When importing in Galileo 10, the "Default Node" of the only CANopen communication will be set to 2:




This means, while the addressing of the first tag can remain simply "Index 100, Sub 0.0". The addressing of the second tag has to be adopted to include the node ID now:




- Special functions and script functions like "Comm. on Connection 0 ON/OFF" are replaced with appropriate functions "CAN SDO Node X On/Off" and/or "CAN PDO On/Off"

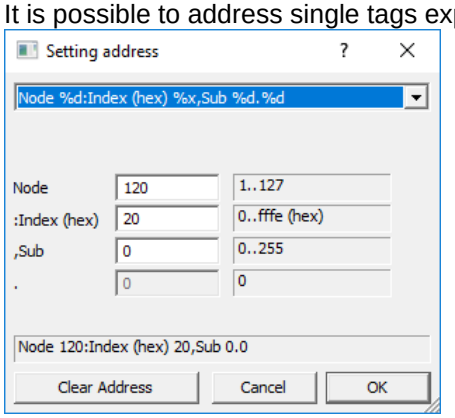
Settings to check manually:

- Because the different CAN SDO communications could have had different settings in Galileo 8, the settings have to be checked. Only the settings from the first found communication in the project are applied.
- Possible "Comm. on Connection x ON/OFF" functions have to be checked, especially if other communications are used as well. Example:
  - Galileo 8 project contains following communication in exactly this order:
    - CAN – polling (0)
    - CAN – polling (1)
    - CAN – polling (2)
    - CODESYS-2 (3)
  - A special function "Comm. on Connection 3 ON/OFF" would have operated on the CODESYS-2 communication
  - When importing the project to Galileo 10, the first three communications are consolidated to one, leaving the project with following communications:
    - CANopen (0)
    - CODESYS-2 (1)

- Therefore, all the special functions "Comm. on Connection 3 ON/OFF" are now replaced with "Comm. on Connection 1 ON/OFF"

Possible known compatibility problems:

- Turning on/off communication to multiple CAN SDO Nodes with one function call is no longer possible. Explanation
  - In Galileo the CAN SDO communication has a setting for "Default Node" (e.g. Node 5)
  - Any tag addressed on this communication by default operates on this node
  - It is possible to address single tags explicitly on another node:




  - When executing the function "Comm. on Connection 0 ON/OFF", it affected the communication to CAN Node 5 (Default Node) as well as CAN Node 120 (specific tag)
  - Any existing "Comm. on Connection 0 ON/OFF" function on import just gets replaced by the function "CAN SDO Node 5 On/Off" (the default node ID).

#### 3.1.9.3 Gateway of tags

In Galileo 8 it was possible to define a gateway between multiple CANopen communications. For example, having tag "A" addressed on a PDO (as "Master") and addressed on an SDO (as "Slave"). The value received from the PDO than would have been written to the SDO.

Because Galileo 10 merged the different CANopen communications into one, there is no longer the possibility to configure such a gateway functionality.
If such a functionality was used, as a possible workaround, the loop script should be considered to fulfill this approach now:

- Create a tag "A" addressed on a PDO
- Create a tag "B" addressed on an SDO
- Create a loop script and copy the value of "A" to "B"

## 3.2 Importing Issues

This chapter covers issues, that can occur during the import of a Galileo 8 project into Galileo 10.

### 3.2.1 Issue1: No longer supported panels

Galileo 10.0 doesn't support all the panels available in Galileo 8.1.x. Following panel series are supported:

- XV100: all panels except of the monochrome types
- XC100: all panels
- XV400: all panels with support of more than 256 colors
- XV300: all panels
- XP500: all panels
- Galileo Open

Projects based on a no longer supported panels can't be imported into Galileo 10.

Solution:

- Open the original project again in Galileo 8.1.x
- Change the panel type to a compatible one from the list above
- Save the project
- Import the project again with Galileo 10

### 3.2.2 Issue2: No longer supported communications

Galileo 10.0 doesn't support all the communications available in Galileo 8.1.x. Following communications are supported:

- A. Bradley - Logix - DF1
- A. Bradley - Logix - EtherNet/IP
- A. Bradley - MicroLogix - EtherNet/IP
- A. Bradley - SLC 5/03 - MicroLogix - DF1
- Beckhoff TwinCAT TCP/IP
- CANopen
- CODESYS V2
- CODESYS V3
- ELC (various)
- Modbus ASCII
- Modbus RTU
- Modbus TCP
- Omron Host-Link (C-mode and FINS)
- Omron FINS over UDP
- Persistent Tag Storage
- Sucom A (various PLCs)
- Siemens - Industrial Ethernet
- Siemens - MPI
- Siemens - PPI
- Siemens - S7 Profibus Standard Profile

Communications which are not supported are removed from the project including all the tags, which were addressed on this communication.

Solution:
If you want to keep the tags, which are addressed on such a no longer supported communication, do following:

- Open the original project again in Galileo 8.1.x
- Change the communication type of the no longer supported communication to one (compatible) from the list above
- Save the project
- Import the project again with Galileo 10

If the list above doesn't contain a communication, which suits your needs, you are not able to upgrade the project to Galileo 10 at the moment without losing the tags addressed on the appropriate communication.

We will continuously add some more communications to Galileo 10, so please be patient and look forward to new versions and their communication compatibility list.

### 3.2.3 Issue3: Users/User Groups were renamed

In Galileo 10 the rules for names of the users and user groups are stricter than in Galileo 8. The names must start with a letter (Latin) or an underscore. The rest of the name must contain only letters, numbers and underscores. For example **User A** is a valid name in Galileo 8 but no longer in Galileo 10. When importing a project with this user name, the user gets automatically renamed to **User_A.**

This can cause a problem in following situation:

- The project and its user management was imported/converted to Galileo 10
- You download the project without selecting the user management for downloading
- ➔ Now you will encounter the fact, that the data in the user management file on the panel is no longer compatible with the data in the downloaded project

Solution 1:
Apply this solution, if your project doesn't allow changes on the users or user groups on the panel itself. The user management is always set up in the design tool and only these data are relevant.

- Open the project in Galileo 10
- Let Galileo 10 do the renaming of the affected users and user groups
- In the "Build and Deploy Settings" of Galileo 10 choose "Download user management data"
- Download the project on the panel

Solution 2:
Apply this solution, if your project supports changes on the users or user groups on the panel itself and you are sure, that modifications have been done on the panel. This includes operations on the panel like "adding a new user", "deleting a user", "changing password", …

- Open the original project again in Galileo 8.1.x
- Upload the user management data from the panel into the project using the "Upload Data" function
- Save the project (with the now updated user management data)
- Import the project again with Galileo 10
- Let Galileo 10 do the renaming of the affected users and user groups
- In the "Build and Deploy Settings" of Galileo 10 choose "Download user management data"
- Download the project on the panel

### 3.2.4 Issue4: Unit translation replacement in script

Galileo 10 no longer supports unit translations. They were replaced by the "Conversion Manager" (see also chapter 3.1.1 for further information).
This change also effects scripts, which used the special function call "Unit group Mode A/B" to switch between the two modes of a unit group. Galileo 10 converts this function automatically to a script calling function. The called script should do all the conversion changes to be compatible with the Galileo 8 approach. However, some manual verification is recommended to ensure, that all the conversions are functionally compatible with Galileo 8.

### 3.2.5 Issue5: Camera control

The set of functions for the camera control has changed compared to Galileo 8 and the old functions are not converted automatically on import. They must be created again.
The former functionality to load an image to the camera control is no longer available but can be replaced with:

- A control of type "Image", setting the property "Image can be loaded dynamically"
- Using the special function "Image Load" or "Image Load Var"

### 3.2.6 Issue6: Revision of system message languages

During the import, Galileo 10 tries to match some project languages to the available system languages (English, German, Italian, Spanish, French) e.g. if you have a project language called “Spanish”, Galileo 10 will assign the system message language “Spanish” to this project language. However, this is not possible in any case so it’s recommended, that you verify the assignment of system message language to project language manually. You can do this in the “Project Languages” dialog. (See also chapter 3.1.2 for further information.)

### 3.2.7 Issue7: Invalid characters in name

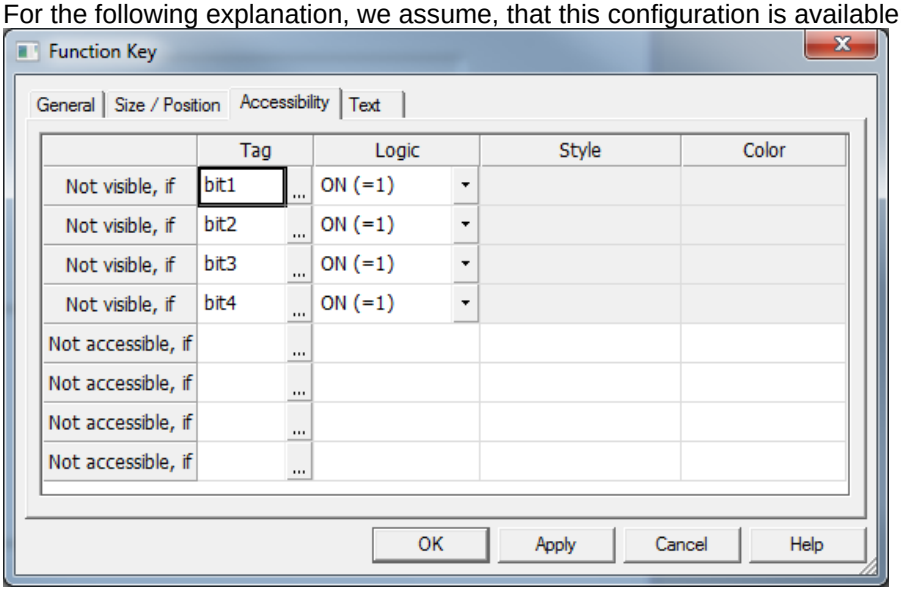
In Galileo 10 the rules for names for objects are stricter than in Galileo 8. The names must start with a letter (Latin) or an underscore. The rest of the name must contain only letters, numbers and underscores. During the import, Galileo 10 fixes such (now) invalid names automatically and informs you about the changes. No further, manual actions are needed.

### 3.2.8 Issue8: Visibility limitation reached

This problem can occur during the conversion of the formerly known “Unit translations” to the new “Conversion Manager” (see also chapter 3.1.1 for further information).
During the import every function key with the special function “Unit group Mode A/B” will be converted to two function keys, which call then the appropriate scripts to perform the conversion change. Each of these two function keys becomes an additional visibility setting. If your function key in Galileo 8 already had four visibility settings, then the limit is already reached and the new visibility can’t be added.

Solution:
There’s no general solution to this problem. Anyhow, a solution must involve the elimination of one of the used visibility settings. This can be achieved with following solution:

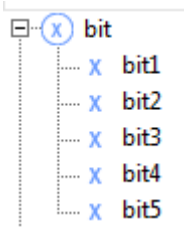
- Open the original project again in Galileo 8.1.x
- Select the appropriate function key with the special function “Unit group Mode A/B”, which has already four visibility settings
- For the following explanation, we assume, that this configuration is available:




We now need to eliminate one of these entries.

- If tags are addressed on a PLC:
  The easiest way would be, to create a new bit tag (let’s say “bit5”) in the appropriate PLC and assign always the logical “OR” operation result of the two tags “bit3” and “bit4”:
  bit5 := bit3 OR bit4
  Then eliminate the “bit3” and “bit4” entries and use the tag “bit5” as a visibility setting.

- If tags are not addressed (internal Galileo tags)
  Create a new tag of type “Bit” in your project. We name it here “Bit5”.




Open (or create) a loop script and add following assignment to the loop script:
bit5:=bit3 OR bit4;

Then eliminate the “bit3” and “bit4” entries and use the tag “bit5” as a visibility setting.

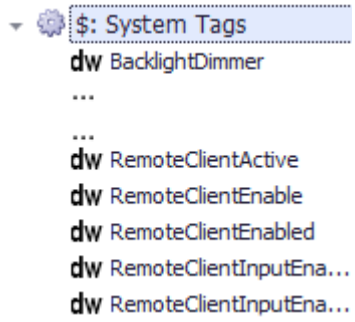
- Save the project
- Import the project again with Galileo 10

### 3.2.9 Issue9: Error Tag Printing

The whole online printing functionality is no longer supported by Galileo 10. This affects also the “Printing” flag formerly available at error tags.
In Galileo 10, it’s possible to define, that error tags will be logged also within the “User Logger”

### 3.2.10 Issue10: Tags of no longer supported “System” communication

The special communication “System”, which allowed to get state information like e.g. “Remote Client Active” is no longer supported in Galileo 10.
The replacement for these functionalities are the newly introduced “System Tags”.




The behavior of the import depends on the tag addressing:

Situation 1: A tag was just addressed on the “System” communication
Controls, which used such a tag now automatically use the appropriate “System Tag”.

Situation 2: A tag was addressed on the “System” communication and another PLC (Bridging)
Controls, which used such a tag now automatically use the appropriate PLC-addressed tag.
The functionality of bridging such tags directly to/from any other PLC’s value is no longer available.

To achieve the same result, you have to do such value assignments in a loop script. This step will not be made automatically by the import.

### 3.2.11 Issue11: ESC Printers and Forms

Galileo 10 no longer supports ESC printers. Therefore, the function “Print ESC Sequence” is no longer supported. Also the functions “Print Form XY” and “Print FormFeed” are no longer supported.
The function “ChangeToPrinter” is no longer supported in scripts.

### 3.2.12 Issue12: Displaying special functions on keyboards

A few of the displaying special functions are no longer supported by Galileo 10. This includes:

- Display ‘Value/Text’ Password
- Display ‘Value’ in Hex
- Display ‘Minimum’ in Hex
- Display ‘Maximum’ in Hex

**For the function “Display ‘Value/Text’ Password” following conversion will be applied:**
Functions will be replaced with the special function “Display ‘Value/Text’”. The keyboard types, on which such functions were used, will be changed to the password supporting type (e.g. keyboard type “Numeric” will be changed to “Numeric Password”).

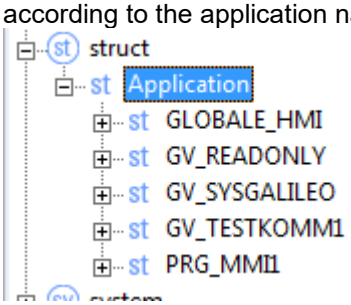
**For the functions regarding “Hex” displaying following conversion will be applied:**
Functions will be replaced with their “normal” equivalent, which means e.g. “Display ‘Minimum’” instead of “Display ‘Minimum’ in Hex”.
The decision, if the display should show the numeric value or the hexadecimal value, is made by the keyboard calling control and it’s displaying format. You can set the format on the control “Value Display/Entry”.

### 3.2.13 Issue13: Tags of type “Bit” for control “Bargraph”

Galileo 10 no longer supports assigning a tag of type “Bit” to a “Bargraph”. In case for just displaying a filled/empty area it is recommended to replace the “Bargraph” control with a “Flag Display” control.

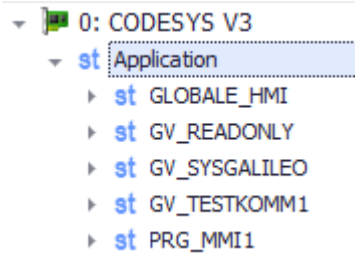
### 3.2.14 Issue14: “Application” tag no longer needed for CODESYS V3 communication

Importing a CODESYS V3 symbol file in Galileo 8.x lead to a structure, where the root tag was named according to the application name in CODESYS V3, e.g. “Application”:

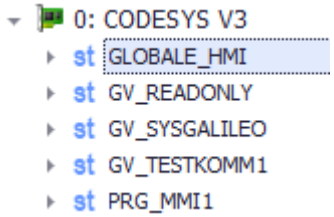



In Galileo 10 it no longer makes sense to have such a tag since the tags are split up by communications. For a better support on re-importing the symbol file later, you should move all tags from within this structure to the “root” level directly under the communication entry (and after that, remove the “Application” tag):

Current:



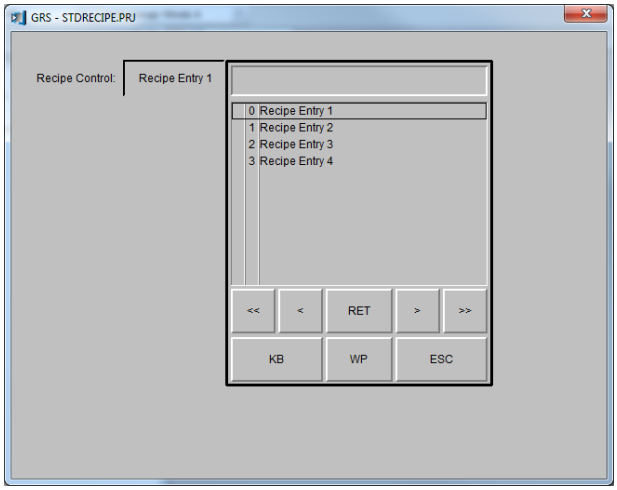

After manually moving:



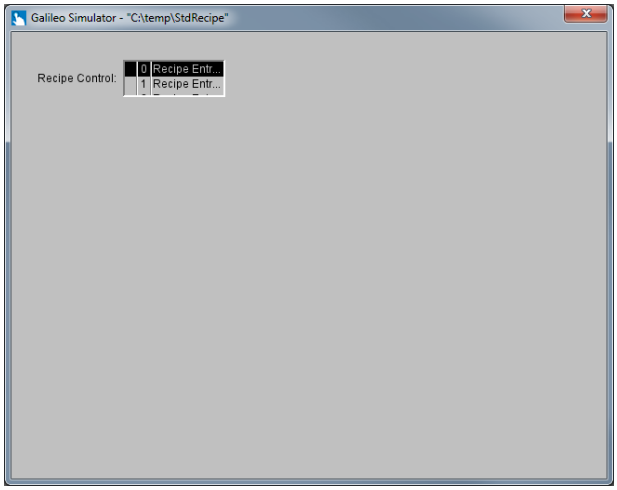

Remark: This step is only needed, if in the Galileo 8.x project more than one tag on root level was available – otherwise Galileo 10 fixes the above mentioned things automatically.

### 3.2.15 Issue15: Recipes of type “Standard” are no longer supported

In Galileo 8.x there were two types of recipes: “Standard” and “Enhanced”. The type “Standard” will be no longer supported.
A control of type “Recipe” in Galileo 8.x with a “Standard” recipe assigned, opened a dialog to load/rename entries




In Galileo 10 after importing such a project, the control of type “Recipe” represents the entries directly on the screen.

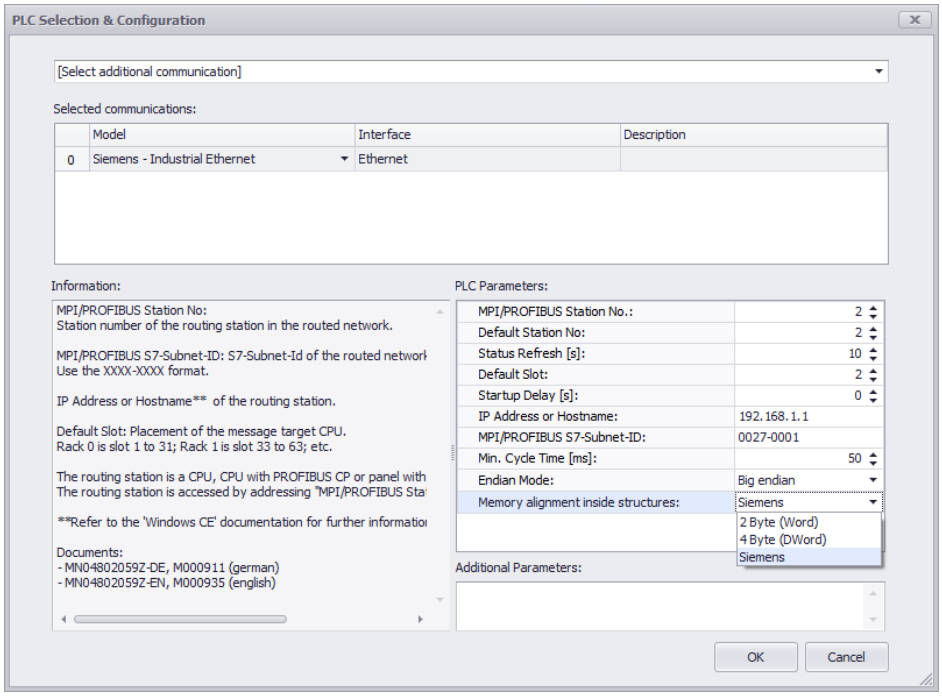



There are two possible solutions to solve this issues:

1. If you have enough space on your current screen: Enlarge the size of the “Recipe” control so the entry names are completely visible and embed “Function Key” controls to load and rename entries.
2. If there isn’t enough space or you want to control access to the recipe operations through a single control: Create a new screen dedicated for displaying the recipe entries and performing the operations like “Load” or “Change Recipe Entry Name”. Replace the formerly used “Recipe” control on the initial screen with a “Screen Change” control which calls the newly created screen. You can control the access by simply configure the accessibility of this “Screen Change” control.

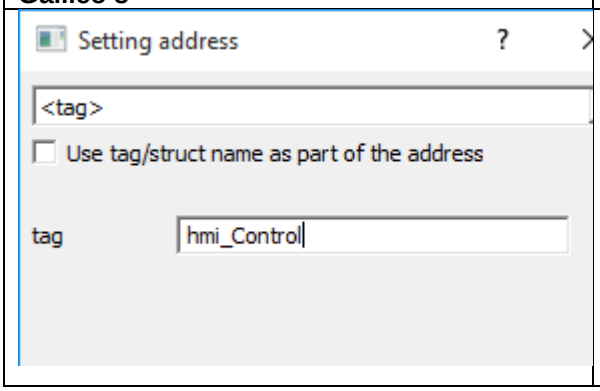
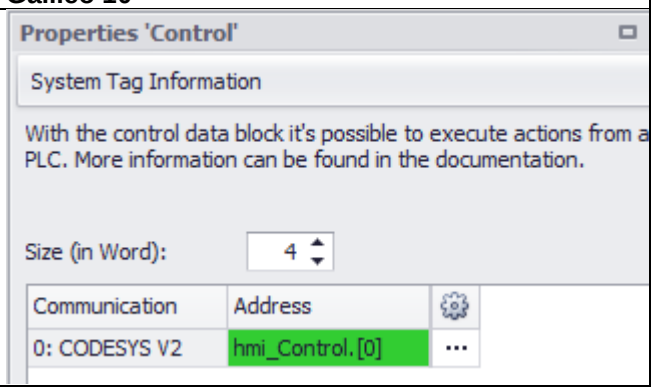
### 3.2.16 Issue16: Siemens memory alignment when using “String” tags

In Galileo 10 a new memory alignment type for Siemens communications called “Siemens” exist. When using “String” tags on a Siemens communication you should now use this alignment type to avoid communication errors.



### 3.2.17 Issue17: Missing array indexer

In Galileo 10 it is no longer possible to address tag array without using syntax with tag array indexer '[%d]'. For example if in Galileo 8 system tag Control is addressed as a 4 word long array called 'hmi_Control' on the PLC one can use address 'hmi_Control' without any errors. In Galileo 10 this has to be manually fixed to 'hmi_Control.[x]'.

| Galileo 8 | Galileo 10 |
|---|---|
|  |  |





## 3.3 Performance Issues

Galileo 10 has a different runtime behavior than Galileo 8, some use cases are faster and some are a bit slower. This chapter shall help you, if you are not satisfied with the performance and how you can positively influence it.

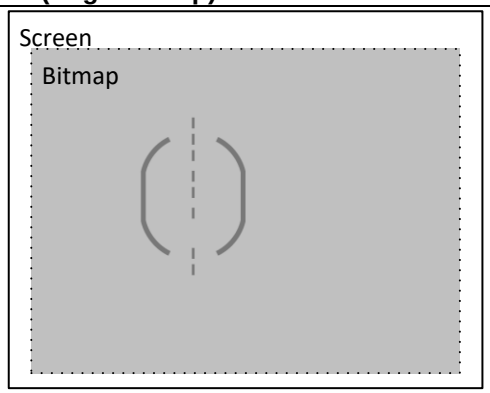
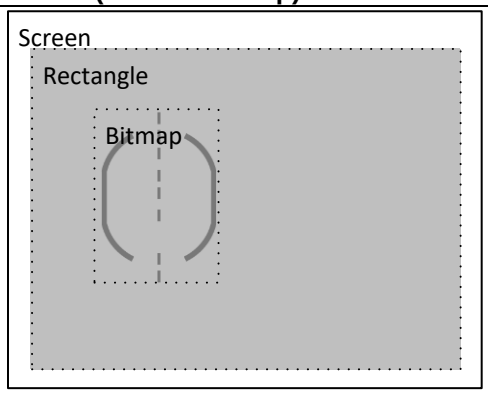
### 3.3.1 Bitmaps

Galileo 10 has an internal bitmap cache, which holds the most frequently used bitmaps (images) in memory (instead of loading them whenever they are needed). Of course, the size of the bitmap cache depends highly on the target device and the overall project size. The rule is easy: the more bitmaps fit into the bitmap cache, the better is the performance. This can be achieved by reducing the number of bitmaps and/or by making them smaller

#### 3.3.1.1 Size of a (background) bitmap

Many customer screens use a large bitmap as a background, which covers the whole screen. In some cases, it is possible to make the bitmap smaller and set a color rectangle behind.

Example:

| Before (large bitmap) | Afterwards (smaller bitmap) |
|---|---|
|  |  |



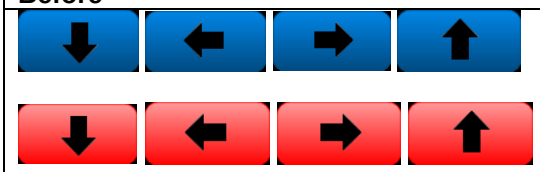


### 3.3.2 Reduce the number of bitmaps

Galileo 10 now allows to define a foreground type for buttons as well as background types. They will be combined at runtime. With this new feature, it is possible to reduce the number of bitmaps (see sample below, where we had eight bitmaps before and afterwards just 6). On top of that, we have less redundancy,

so changing for example backgrounds of several buttons can be done with adaption of one since bitmap (instead of touching all).

Example:

| Before | Afterwards |
|---|---|
|  | Backgrounds:  <br>Foregrounds:  |

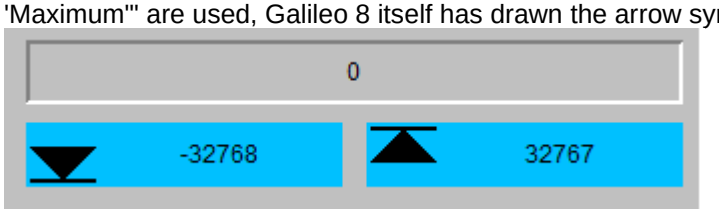
## 3.4 Graphical changes

### 3.4.1 Gauge drawing

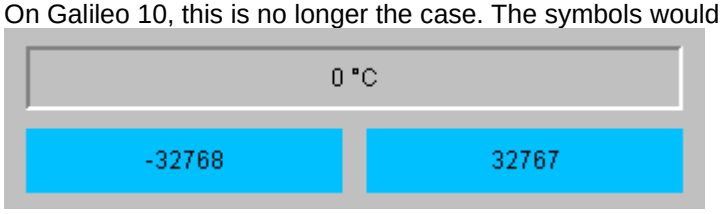
Because of the different graphic rendering engines used in Galileo 10 compared to Galileo 8, controls of type "Gauge" doesn't always look exactly identical compared to Galileo 8. There can be minor differences regarding some pixel shifting or scaling/sizing of the gauge. It's recommended to quickly check the gauge controls in the project manually to see if they still display the content properly.

### 3.4.2 Numeric keyboards

On customer created numeric keyboards where the display functions "Display 'Minimum'" and "Display 'Maximum'" are used, Galileo 8 itself has drawn the arrow symbols by default:




On Galileo 10, this is no longer the case. The symbols would have to be added manually if desired:




Another difference compared to Galileo 8 is, that the unit of a possibly assigned conversion will now be displayed as well in the keyboard while entering the value. This makes it easier for the user to identify, what kind of value he is currently entering.