Variant selection with application parameters

PILZ
THE SPIRIT OF SAFETY

Product
Type: FS_SafetyGate_PLC
Name: PSS4000, Blocks, PAS4000, PLC, STL
Manufacturer: Pilz GmbH & Co. KG, Safe Automation

Document
Release Number: 02
Release Date: 13 September 2017

Document Revision History

Release	Date	Changes	Chapter
01	-	-	all
02	2017-09-13	Creation	all

Validity of Application Note

This present Application Note is valid until a new version of the document is published.
This and other Application Notes can be downloaded in the latest version and for free from www.pilz.com.
For a simple search, use our content document (1002400) or the direct search function in the download area.

Exclusion of liability

We have taken great care in compiling our application note. It contains information about our company and our products. All statements are made in accordance with the current status of technology and to the best of our knowledge and belief.
However, we cannot accept liability for the accuracy and entirety of the information provided, except in the case of gross negligence. In particular it should be noted that statements do not have the legal quality of assurances or assured properties.
We are grateful for any feedback on the contents.

September 2017

Contents

Abbreviations

PSS	Programmable control system (DE: **P**rogrammierbares **S**teuerungs**s**ystem)
POU	**P**rogram **O**rganisation **U**nit
Plant variant	Variation of the system according to their hardware-side characteristics.
Project variant	PAS-Project specific implementation of configuration and program on the individual plant variants.

1. Useful documentation

Reading the documentation listed below is necessary for understanding this application note. The availability of the indicated tools and safe handling are also presupposed with the user.

1.1. Documentation from Pilz GmbH & Co. KG

No.	Description	Item No.
1	Pilz international homepage, download section	www.pilz.com
2	Operating Manual PSSu H PLC1 FS SN SD(-T)(-R)	21939-EN-xx
3	Operating Manual PSSu E F 4DI(-T)(-R)	21310-EN-xx
4	Operating Manual PSSu E F DI OZ 2(-T)(-R)	21328-EN-xx
5	Operating Manual PSSU E F 4DO0.5(-T)(-R)	21316-EN-xx
6	Operating Manual PSSu E S 4DI(-T)	21340-EN-xx
7	Operating Manual PSSu E S 4DO0.5(-T)	21346-EN-xx
8	Operating Manual PSEN cs3.1a	1003295-EN-xx

1.2. Documentation from other sources of information

No.	Description	Item No.
1		
2		
3		
4		

2. Hardware configuration

2.1. Pilz products

No.	Descriptions	Order number	Version	Number
1	PSSu H PLC1 FS SN SD	312070	2	1
2	PSSu E F 4DI	312200	4	1
3	PSSu E F DI OZ 2	312220	2	1
4	PSSU E F 4DO0.5	312210	4	2
5	PSSu E S 4DI	312400	4	1
6	PSSu E S 4DO0.5	312405	3	1
7	PSEN cs3.1a	541061	-	3
8	PAS4000	-	v1.17.0	-

2.2. Hardware configuration

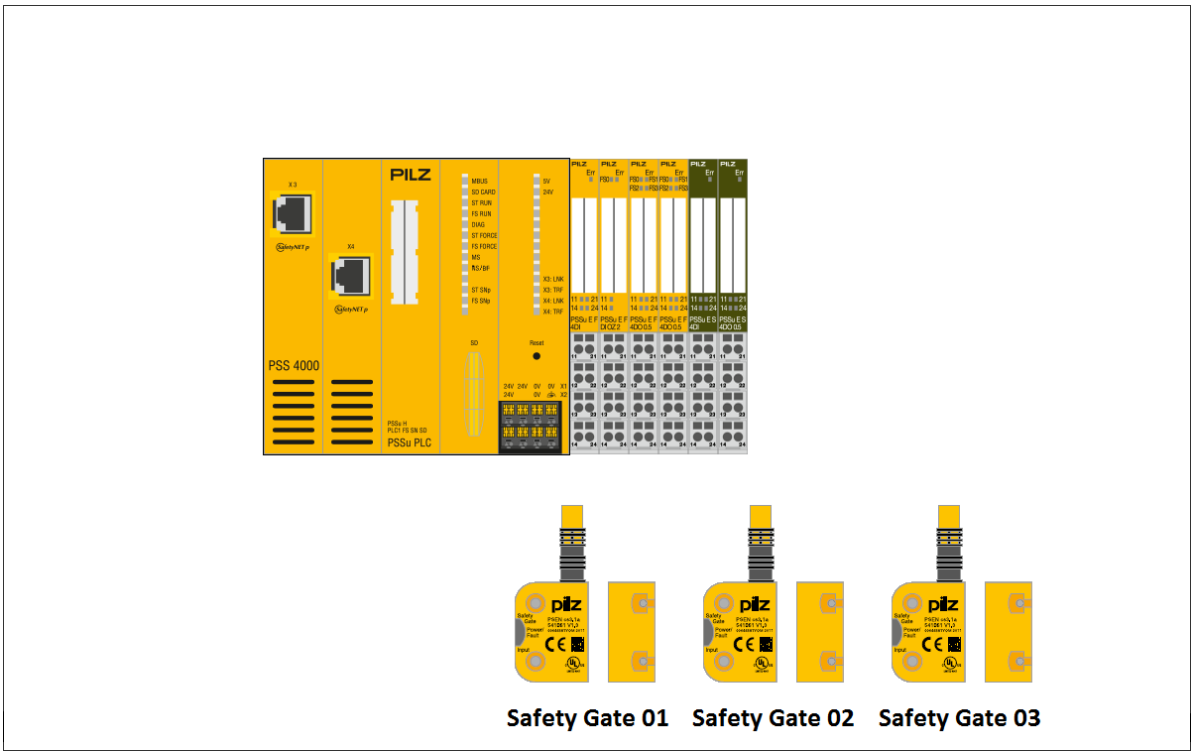


Fig. 1: Hardware configuration

3. Application task

This application note describes the application of a variant-segmented series machine/plant where application parameters are used to select variants.
The hardware structure of the control system is identical in all the variants.

Implementation is shown here only for the PAS4000 section.

Note
The terms *machine* or *plant* in this example are considered as equivalent, we will be using the term *plant* as a synonym for both forms.

3.1. Application parameters

Application parameters are used for the special requirement of adapting plants according to their expansion phase without having to change the user program.
If a plant is to be provided in a different variant, it is sufficient to adapt the application parameters accordingly when there is a corresponding programming. This way, the project has to be created only once for a plant range, and it can afterwards be adjusted to the required variant without a new change of the project.

The initially high amount of programming gives the option to use an already approved project for several variants. This is enabled by a uniform check sum. In doing so, it is created over the entire project excluding the changing application parameters.
The individual variants of the plant must be selected, if possible, so that the overlapping of the individual functions can be used to create a plant that has a similar basic function.

3.2. Project configuration of the variant management

To implement the variant selection using the application parameters, we show the sample process below.

- Procedure when planning the plant
  - Define plant variants.
  - Specify hardware configurations for the plant.
  - Create variant variable (*The variable must be used in the user program read-only*) and define it as application parameter.
  - Integrate variant handling in the user program.
  - Program out the user program.
  - Ensure that variant selection has to take place (e.g. pre-assignment with value 0 and locking the safety functions to this value).
  - Inclusion of the check sums of the project.

- Procedure of variant selection during commissioning
  - Prerequisites:
    - The entire required hardware is available and operational (e.g. naming data are correct, etc.).
    - The commissioning engineer knows the name and the value to be set of the application parameter.
    - PC with PAS4000 installation (in the required version).
  - Start PAS4000 with the valid project
  - Log in as an operator who can change application parameters.
  - Set application parameters of the variant selection in accordance with the plant variant.
  - Build changes and check the result for successful execution.
  - Control of the project check sums for correctness.
  - Download the project.
  - Commission the overall plant in accordance with the set variant.

3.3. Planning the plant

3.3.1. Design of plant variants in the example

The following variants have been provided for the plant range used in the example.

Variant 1:
PSSu PLC with one safety function "Safety Gate 01"

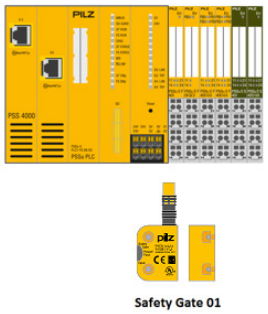


Fig. 2: Variant 1

Variant 2:
PSSu PLC with two safety functions "Safety Gate 01" and "Safety Gate 02"

Fig. 3: Variant 2

Variant 3:
PSSu PLC with three safety functions "Safety Gate 01", "Safety Gate 02" and "Safety Gate 03"

Fig. 4: Variant 3

3.3.2. Programming

3.3.2.1. Definition of the variant variable as application parameter

Create a local variable in the declaration section of the Editor, and right-click to select "*Add Variable to Application Parameters…*".

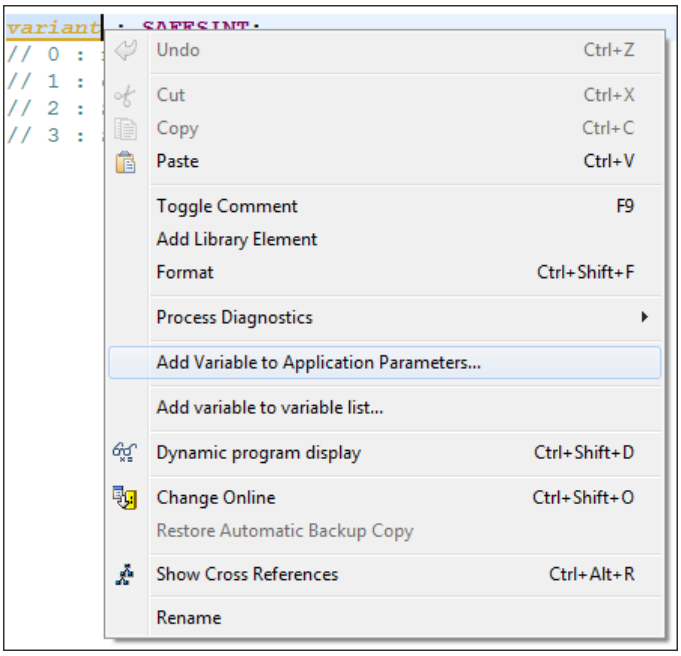


Fig. 5: Add variable to application parameters …

The variable can be assigned to only one group in the application parameters editor.

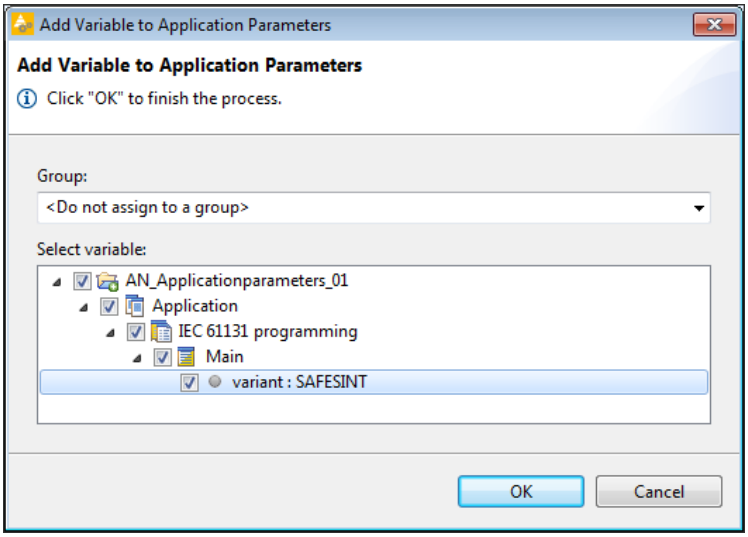


Fig. 6: Add variable to application parameters group

To mark it as an application parameter, the variable will be underlined from now on.

```
variant : SAFESINT;
// 0 : no adjustment made still yet
// 1 : only SG1
// 2 : SG1 and SG2
// 3 : SG1 and SG2 and SG3
```

Fig. 7: Application parameters created

3.3.2.2. Program cycle / program design

The three safety gate blocks of this example are called up and run together in the program code. The complete programming is run as if it were a plant in full expansion stage. The version to another variant than the full expansion is run by reducing individual functions.
Activation of the individual functions is shown as an example for safety gate 01 as follows (The two other safety gate blocks are treated logically as equal.).

```
//XXX SAFETY GATE 01 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
SafetyGate_01(
       SwitchType := SwitchType_01_Replace,
       Input1 := Input1_01_Replace,
       Input1_Valid := Input1_Valid_01_Replace,
       Input2 := Input2_01_Replace,
       Input2_Valid := Input2_Valid_01_Replace,
       Input3 := Input3_01_Replace,
       Input3_Valid := Input3_Valid_01_Replace,
       AutoStart := AutoStart_01_Replace,
       AutoReset := AutoReset_01_Replace,
       MonitoredReset := MonitoredReset_01_Replace,
       StartupTest := StartupTest_01_Replace,
       SimultaneityTime := SimultaneityTime_01_Replace,
       DelayTime := DelayTime_01_Replace,
       Reset := Reset_SG,
       Enable => Enable_SG01,
       DiagOperated => DiagOperated_SG01,
       DiagReadyForReset => DiagReadyForReset_SG01,
       DiagReadyForTest => DiagReadyForTest_SG01,
       DiagSwitchError => DiagSwitchError_SG01,
       DiagInputNotValid => DiagInputNotValid_SG01
       );

//XXX SAFETY GATE 02 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
SafetyGate_02(
       ...
       );

//XXX SAFETY GATE 03 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
SafetyGate_03(
       ...
       );
```

The input variables of the block are not assigned directly with the variables assigned to the hardware inputs, but with alternative variables (xx_xx_Replace) that are set in accordance with the variant selection.

3.3.2.3. Editing the variant-related parameter setting

To set the parameters of the safety-related blocks according to the variant selection, the relevant blocks are only supplied with the real data of the inputs when they are also required in the used variant.
To do this, the variant variable is polled as a selector in a multiselection request *CASE*.

(*You can find more detailed information on this in the PAS4000 0nline help under "Multiselection CASE"*)

Description of the design using the example of variant 1:
When using *Variant 1*, the variant variable has to be initialised with the value "1".
The *CASE* instruction therefore jumps to the statement block with the value "1". Here, the variable mapping of all three safety gate blocks is performed for *Variant 1*.
The safety-related blocks that are not required in the variant are pre-assigned in the selected statement block via the alternative variables with defined substitute values for error-free functioning. This prevents the output of error messages which otherwise would be initiated by these blocks, and it permits to reuse the overall function in accordance with the full configuration.

Attention
A wrong setting of the variant selection relating to the existing plant variant has an effect on safety. A wrong project variant can result in the failure of individual safety functions.
The correct function of the overall plant must be checked during commissioning (see 3.4.5).

```
//XXX Initialising of the FS-Functions SG01, SG02 and SG03 related to the variants XXX
CASE variant OF

        USINT#1: // Only SG01

                //Block 01 - Initialisation by input variables
                SwitchType_01_Replace := USINT#3;
                Input1_01_Replace := Input1_01_IN;
                Input1_Valid_01_Replace := Input1_Valid_01_IN;
                Input2_01_Replace := Input2_01_IN;
                Input2_Valid_01_Replace := Input2_Valid_01_IN;
                Input3_01_Replace := Input3_01_IN;
                Input3_Valid_01_Replace := Input3_Valid_01_IN;
                AutoStart_01_Replace := TRUE;
                AutoReset_01_Replace := FALSE;
                MonitoredReset_01_Replace := TRUE;
                StartupTest_01_Replace := FALSE;
                SimultaneityTime_01_Replace := SimultaneityTime;
                DelayTime_01_Replace := DelayTime;

                //Block 02 - Initialisation by alternative variables
                SwitchType_02_Replace := USINT#3;
                Input1_02_Replace := TRUE;
                Input1_Valid_02_Replace := TRUE;
                Input2_02_Replace := TRUE;
                Input2_Valid_02_Replace := TRUE;
                Input3_02_Replace := TRUE;
                Input3_Valid_02_Replace := TRUE;
                AutoStart_02_Replace := TRUE;
                AutoReset_02_Replace := TRUE;
                MonitoredReset_02_Replace := TRUE;
                StartupTest_02_Replace := FALSE;
                SimultaneityTime_02_Replace := SimultaneityTime;
                DelayTime_02_Replace := DelayTime;
```

```
                //Block 03 - Initialisation by alternative variables
                SwitchType_03_Replace := USINT#3;
                Input1_03_Replace := TRUE;
                Input1_Valid_03_Replace := TRUE;
                Input2_03_Replace := TRUE;
                Input2_Valid_03_Replace := TRUE;
                Input3_03_Replace := TRUE;
                Input3_Valid_03_Replace := TRUE;
                AutoStart_03_Replace := TRUE;
                AutoReset_03_Replace := TRUE;
                MonitoredReset_03_Replace := TRUE;
                StartupTest_03_Replace := FALSE;
                SimultaneityTime_03_Replace := SimultaneityTime;
                DelayTime_03_Replace := DelayTime;

        USINT#2: // Only SG01 + SG02

                //Block 01 - Initialisation by input variables
                SwitchType_01_Replace := USINT#3;
                Input1_01_Replace := Input1_01_IN;
                Input1_Valid_01_Replace := Input1_Valid_01_IN;
                Input2_01_Replace := Input2_01_IN;
                ...
                //Block 02 - Initialisation by input variables
                ...
                //Block 03 - Initialisation by alternative variables
                ...
        USINT#3: // SG01 + SG02 + SG03

                //Block 01 - Initialisation by input variables
                ...
                //Block 02 - Initialisation by input variables
                ...
                //Block 03 - Initialisation by input variables
                ...
        ELSE    // No variant - close of all function
                // Important part!!! -> described in next chapter.

END_CASE;
```

When using the block (*Safety Gate 01*), the alternative variables (xx_xx_Replace) are initialised directly with the variables (xx_xx_IN) that are mapped to the hardware inputs.
The alternative variables (xx_xx_Replace) are initialised with substitute values that permanently set the blocks to Enable = TRUE, when the block (*Safety Gate 02* and *Safety Gate 03*) is not used.

3.3.2.4. Locking the variant selection

To force the commissioning engineer to explicitly select the variant, it is necessary to save the basic form of the project in a neutral version of the variant selection. In the example, this is selected by the variant value "0".
In addition, entering an invalid value for the variant selection in the program must be intercepted. To do this, all the variants in the project must be deselected, and running the plant function must be prevented.
It is also possible to make a corresponding note about an output medium, if available.
This functionality can be performed using the statement blocks *ELSE* within the *CASE* instruction.

```
        ELSE    // No variant - close of all function
                // Error-Message -> Wrong parameter value

                //Block 01 - Initialisation by alternative variables
                SwitchType_01_Replace := USINT#3;
                Input1_01_Replace := FALSE;
                Input1_Valid_01_Replace := FALSE;
                Input2_01_Replace := FALSE;
                Input2_Valid_01_Replace := FALSE;
                Input3_01_Replace := FALSE;
                Input3_Valid_01_Replace := FALSE;
                AutoStart_01_Replace := FALSE;
                AutoReset_01_Replace := FALSE;
                MonitoredReset_01_Replace := TRUE;
                StartupTest_01_Replace := FALSE;
                SimultaneityTime_01_Replace := SimultaneityTime;
                DelayTime_01_Replace := DelayTime;

                //Block 02 - Initialisation by alternative variables
                SwitchType_02_Replace := USINT#3;
                Input1_02_Replace := FALSE;
                Input1_Valid_02_Replace := FALSE;
                Input2_02_Replace := FALSE;
                Input2_Valid_02_Replace := FALSE;
                Input3_02_Replace := FALSE;
                Input3_Valid_02_Replace := FALSE;
                AutoStart_02_Replace := FALSE;
                AutoReset_02_Replace := FALSE;
                MonitoredReset_02_Replace := TRUE;
                StartupTest_02_Replace := FALSE;
                SimultaneityTime_02_Replace := SimultaneityTime;
                DelayTime_02_Replace := DelayTime;

                //Block 03 - Initialisation by alternative variables
                SwitchType_03_Replace := USINT#3;
                Input1_03_Replace := FALSE;
                Input1_Valid_03_Replace := FALSE;
                Input2_03_Replace := FALSE;
                Input2_Valid_03_Replace := FALSE;
                Input3_03_Replace := FALSE;
                Input3_Valid_03_Replace := FALSE;
                AutoStart_03_Replace := FALSE;
                AutoReset_03_Replace := FALSE;
                MonitoredReset_03_Replace := TRUE;
                StartupTest_03_Replace := FALSE;
                SimultaneityTime_03_Replace := SimultaneityTime;
                DelayTime_03_Replace := DelayTime;
END_CASE;
```

The alternative variables (xx_xx_Replace) are initialised with substitute values when the block (*Safety Gate 01*, *Safety Gate 02* and *Safety Gate 03*) is not used. The substitute values set the blocks permanently to Enable = FALSE.

3.3.3. Expanded use of the hardware variation

The implementation presented in the example only shows the basic procedure in a simple application.

Of course, this procedure can also be used for more complex variants.

Not only can substitute values be placed in via the alternative variables, but also entirely different hardware inputs and outputs can be mapped.
Depending on the variant, alternative variables can be used to e.g. map hardware inputs to the input values of various safety functions.
In addition, output values of safety functions can be mapped via alternative variables to different hardware outputs.

This increases the variability of the used hardware, as the available hardware can be used with changed (software) mapping (as part of the real inputs and outputs).
It is possible, for example, that a hardware input in a variant is mapped to a safety gate block, and in the other variant to an E-STOP block.

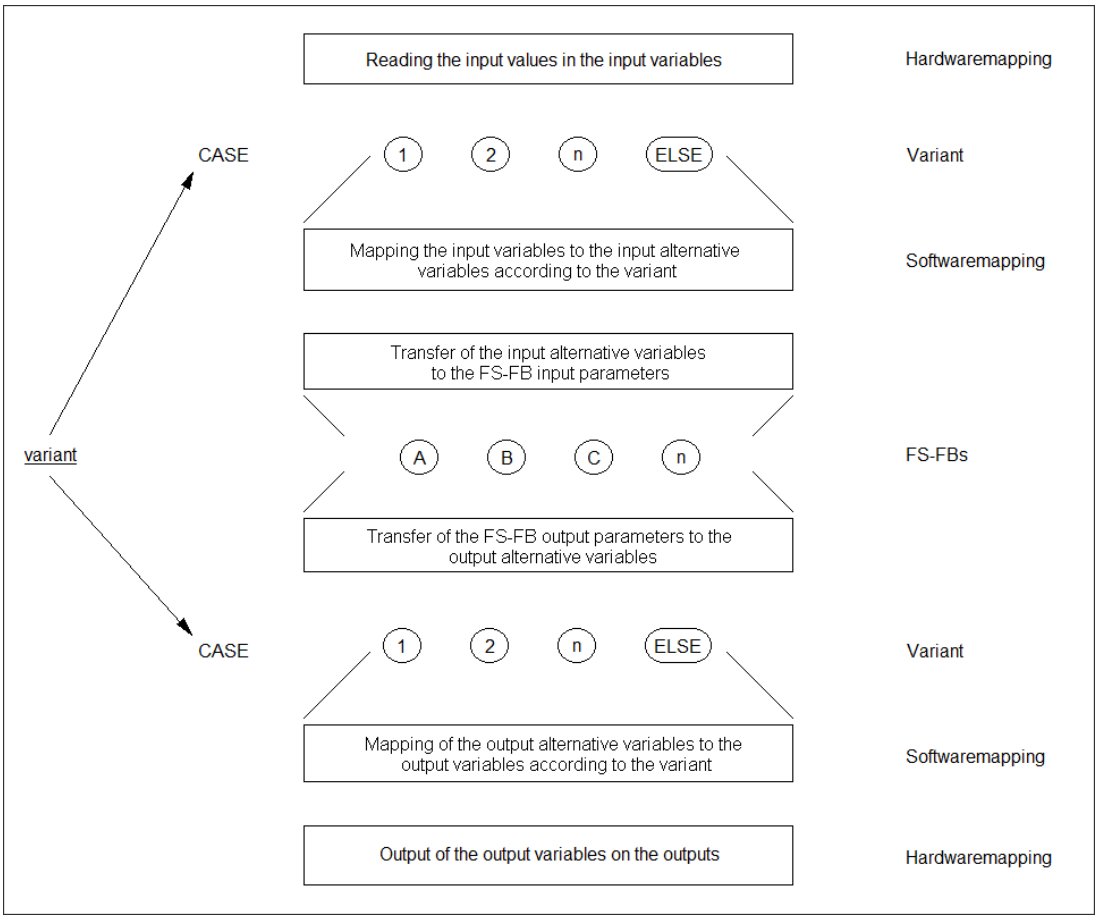


Fig. 8: Procedure of expanded hardware use by variant mapping

3.3.4. Including the check sum(s) in the FS project

In PAS4000 in the project tree, under *Properties/General*, you can view the check sums FS project, *Application parameters of the FS project* and FS *project without application parameters*.

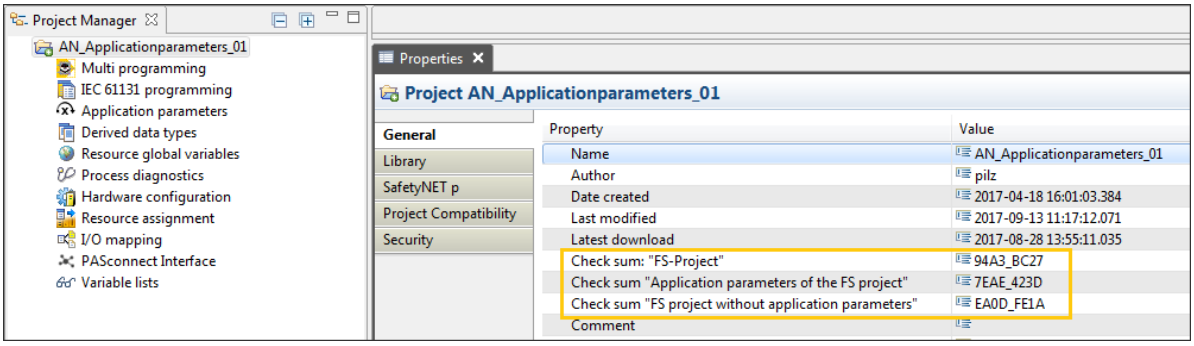


Fig. 9: Check sums

Check sum "FS Project"
Includes all the safety-related parts of a project from the user program, the hardware configuration, the resource assignment and the I/O assignment.

Check sum "Application parameters of the FS project"
Includes the values of all the application parameters available on the FS resources of the project.

Check sum "FS project without application parameters"
Corresponds to the check sum "FS project", but without the values of the application parameters present on the FS resources of the project. The application parameters itself are included in the check sum, i.e. the check sum changes when application parameters are added or deleted.

Evidence that the FS project (except the values for the application parameters) have not been changes can be provided using the check sum "FS project without application parameters". This check sum remains constant also when changing the data values in the application parameters.

3.4. Variant selection during commissioning

3.4.1. Registration as an operator with restricted rights

To avoid an accidental change of a project during commissioning, a user with limited access should be created in the user administration.
To do this, a new user group is configured, and the relevant user is assigned the required authorisations via this user group.

For this example, a user is required who only has the *Read project* and *Application parameters* authorisations.
This permits to only change the application parameters during commissioning, and therefore the check sum "*FS project without application parameters*" cannot be changed.

Under *Tools -> User Management Editor*, under *User groups* the button [*New group*] is used to create a new user group named *ApplicationParameterUser*.

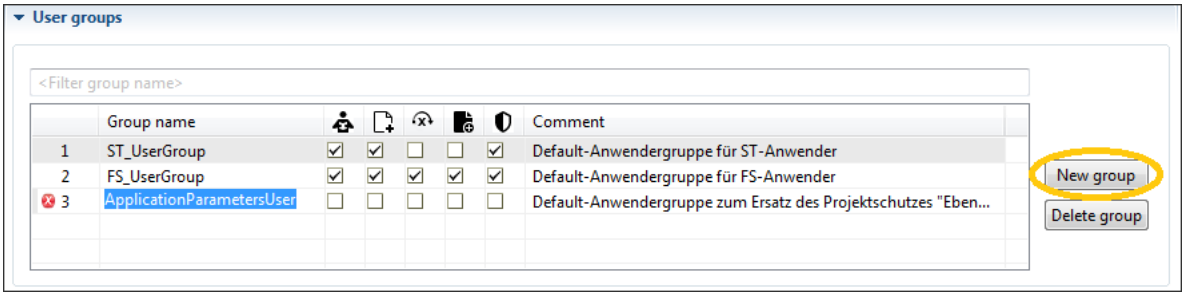


Fig. 10: User group

For this new group the permissions *Read project* and *Application parameters* are assigned now.

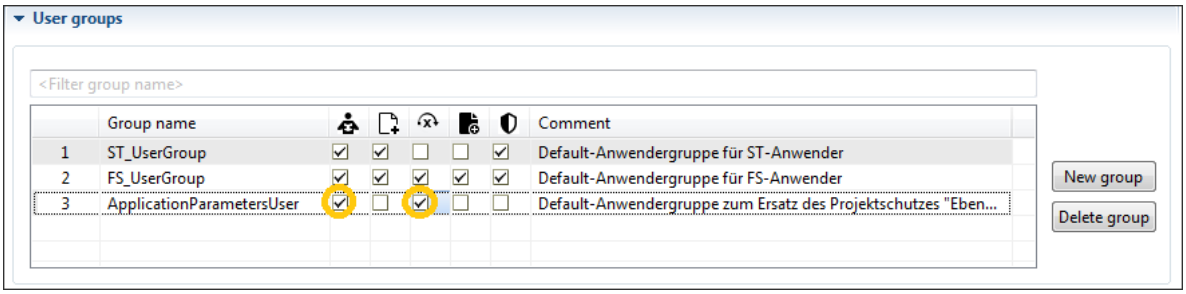


Fig. 11: Permissions

Under users, a new user with the name *Commissioning_User* is created using the button [*New user*] created.

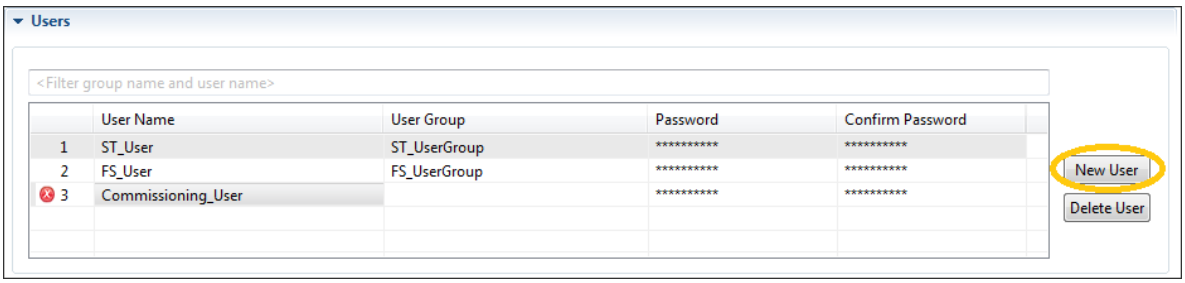


Fig. 12: New user

For this user, the previously created user group *ApplicationParameterUser* is selected, and a password is assigned.

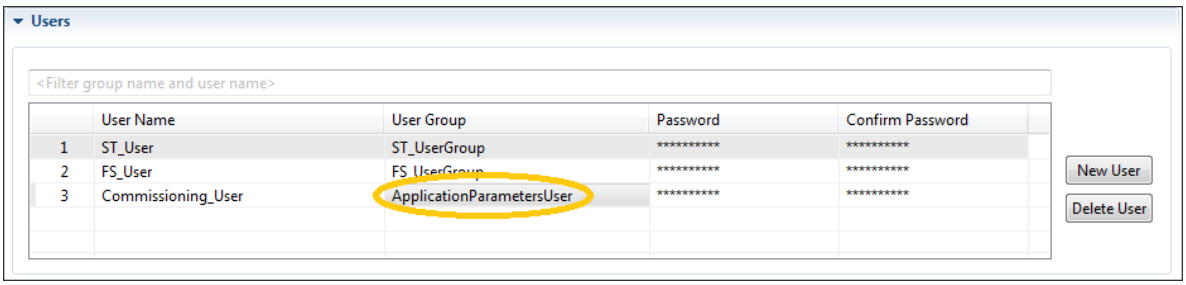


Fig. 13: Mapping user group

3.4.2. Selecting the plant variant

The selection of the project variant is run by setting the variant variable to a valid value in the application parameters Editor.
The selected project variant has to match the plant variant on the hardware side.

Information
The project variants are static expansion phases that are set once during commissioning in accordance with the hardware design of the plant.

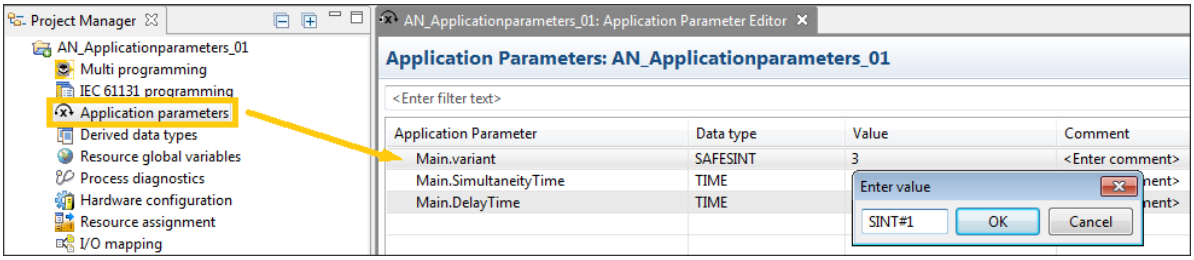


Fig. 14: Set application parameters

When further application parameters have been created, it is now possible to set them too.

In the example, application parameter variables have also been created for the time variables (*SimultaneityTime* and *DelayTime*) of the safety gate blocks.

```
SimultaneityTime;
DelayTime;
```

3.4.3. Building project and checking the check sums

The project has to be built again, and then the check sums must be compared to the check sums approved during creation.

Attention
The correctness of the check sums relating to the existing plant variant has an effect on safety.
An incorrect project variant can cause the failure of individual safety functions.
The correct function of the entire plant must be checked when commissioning (*see 3.4.5*).

3.4.4. Download project

If the check sums of the project match the specifications, the project can be transferred to the control system.

3.4.5. Commissioning of the plant

Perform the further general and safety-related commissioning in accordance with the directives and standards valid for the plant.

4. Table of figures

Support

Technical support is available from Pilz round the clock.

Americas
Brazil
+55 11 97569-2804
Canada
+1 888-315-PILZ (315-7459)
Mexico
+52 55 5572 1300
USA (toll-free)
+1 877-PILZUSA (745-9872)

Asia
China
+86 21 60880878-216
Japan
+81 45 471-2281
South Korea
+82 31 450 0680

Australia
+61 3 95446300

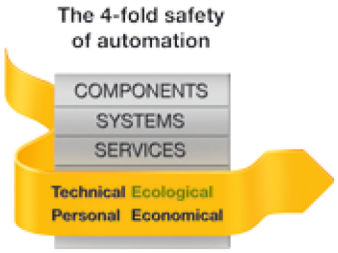
Europe
Austria
+43 1 7986263-0
Belgium, Luxembourg
+32 9 3217575
France
+33 3 88104000
Germany
+49 711 3409-444
Ireland
+353 21 4804983
Italy
+39 0362 1826711
Scandinavia
+45 74436332
Spain
+34 938497433
Switzerland
+41 62 88979-30
The Netherlands
+31 347 320477
Turkey
+90 216 5775552
United Kingdom
+44 1536 462203

You can reach our international hotline on:
+49 711 3409-444
support@pilz.com

Pilz develops environmentally-friendly products using ecological materials and energy-saving technologies. Offices and production facilities are ecologically designed, environmentally-aware and energy-saving. So Pilz offers sustainability, plus the security of using energy-efficient products and environmentally-friendly solutions.

Energy saving by Pilz

BLUECOMPETENCE Alliance Member
Partner of the Engineering Industry Sustainability Initiative



pilz
Partner of:
The Best of German Engineering

Pilz GmbH & Co. KG
Felix-Wankel-Straße 2
73760 Ostfildern, Germany
Tel.: +49 711 3409-0
Fax: +49 711 3409-133
info@pilz.com
www.pilz.com

PILZ
THE SPIRIT OF SAFETY

100XXX-DE-0X
0-0-1-3-000, 2015-00 Printed in Germany