CREATING AND DESIGNING A LOCAL IPC PANEL FOR A UNICOS APPLICATION

Abstract

This document explains how to create an application for a local IPC panel in an UNICOS project.
## HISTORY OF CHANGES

<table>
<thead>
<tr>
<th>REV. NO.</th>
<th>DATE</th>
<th>PAGES</th>
<th>DESCRIPTIONS OF THE CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.0</td>
<td>2013-10-08</td>
<td>All</td>
<td>First Version (Silvia Izquierdo Rosas)</td>
</tr>
<tr>
<td>v1.1</td>
<td>2014-04-02</td>
<td>22, 41, 42, 44, 46</td>
<td>Extra information for UNICOS library, Siemens workaround for screen speed added, RT License message information added, how to change RT version added, having more than one PLC connection added.</td>
</tr>
<tr>
<td>v1.2</td>
<td>2014-04-29</td>
<td>All</td>
<td>Extra information about PCO implementation, templates, general advice, Configure device: hardware added</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.2</td>
<td>Analog Digital</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.3</td>
<td>Analog Input</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.4</td>
<td>Analog Input Real</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.5</td>
<td>Analog</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.6</td>
<td>AnaDO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.7</td>
<td>Analog Output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.8</td>
<td>Analog Output Real</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.9</td>
<td>Analog Parameter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.10</td>
<td>Analog Status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.11</td>
<td>Digital Alarm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.12</td>
<td>Digital Input</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.13</td>
<td>Digital Output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.14</td>
<td>Digital Parameter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.15</td>
<td>Local</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.16</td>
<td>On Off</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.17</td>
<td>Process Control Object</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.18</td>
<td>Controller</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.19</td>
<td>Word Parameter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.20</td>
<td>Word Status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td>EXAMPLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Power supply</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Creating new project</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Choosing IPC model</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ethernet port configuration</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Adding HMI application</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Renaming device</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RT loader</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>MPI connection configuration</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Template configuration</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Set template as default one</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Screen configuration</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>New folder link</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Opening UAB application project</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>UNICOS Generators window</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>WinCC Flexible Instance Generator</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Importation of tags</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Tag tables</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Importing text lists</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Importing alarms</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Configuring script</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Script load</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Opening UNICOS-CPC6 library</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Upgrade library</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Drag and drop AI widget and choose the corresponding tags</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Drag and drop AI faceplate and choose the corresponding tags</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>AI widget modification</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Disable layer</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Properties / Release faceplate type</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>PCO widget/faceplate</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Text lists imported</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Configuring text list for PCO option modes</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>PCO widget</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>PCO faceplate</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>New text list on WS widget</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Configuring text list for WS widget</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Runtime system with simulator</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Panel simulation</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Layers</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Transfer from TIA portal</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Transfer of the application via Ethernet</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Distribution of navigation buttons</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Configuration of navigation buttons</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>Users</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>Script SecuritySettings</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>Placing login/logoff buttons</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>Writing text for buttons</td>
<td>37</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>Selecting ShowLogonDialog function</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>Configuring IO Field for userName</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>View of the login/logoff buttons and user display</td>
<td>38</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>Scheduled tasks / Properties</td>
<td>39</td>
<td></td>
</tr>
</tbody>
</table>
1. HARDWARE NEEDED

To create an application in a touch panel you need:

- 1 Ethernet cable
- 1 Power supply for the touch panel

The Ethernet cable is used to connect the touch IPC panel with the PLC.

Figure 1 - Power supply
2. SOFTWARE NEEDED

It is necessary to install WinCC Advanced. The version used in this tutorial was WinCC Advanced V11 SP2 (TIA portal).

Before getting started, make sure you have the following software installed:

1) Check you have access to: \\
cern.ch\dfs\Applications\Siemens\LicensedSoftware
   a. If you don't have access, send email to Icecontrols.support@cern.ch

2) Simatic Step7Professional (for PLC project)
   a. Install latest version of Step7Professional software
   b. Go to: \\
cern.ch\dfs\Applications\Siemens\LicensedSoftware\Step7Professional
   c. For the license, go to: \\
cern.ch\dfs\Applications\Siemens\LicensedSoftware\Step7Professional\CERN license
      i. pick correct version which you installed, and run setup.exe

3) WinCC Advanced
   a. Install latest version of WinCC Advanced
   b. Go: \\
cern.ch\dfs\Applications\Siemens\LicensedSoftware\TIAPortal\WinCC
   c. For the license go to: \\
G:\Applications\Siemens\LicensedSoftware\TIAPortal\WinCC\CERN License

4) UAB Bootstrap
   a. Install latest version of UAB Bootstrap
   b. Go: \\
https://j2eeps.cern.ch/wikis/display/EN/UAB+Bootstrap
3. CREATION OF IPC PROJECT IN TIA PORTAL

Start TIA PORTAL (Totally Integrated Automation), V12 has been used for this procedure.

To create a new project click on “Create new project”, choose a Project name and the path where you want to save it and click on “Create”.

![Figure 2 - Creating new project](image.png)
4. **CONFIGURE DEVICE**

4.1 **Software**

To configure the IPC model, we will use the option “Configure a device” in next window. Then click on “Add new device” in the left part.

From next window choose the IPC model that will be used in the application. In this case we will use an IPC277D 15” (PC systems/SIMATIC Panel PC/IPC277D). Then click on “Add”. See Figure 3.

![Figure 3 - Choosing IPC model](image)

To configure the IP address of the device click on the ETHERNET port that will be used for the connection, and go to the configuration in “PROFINET Interface”, add a new subnet and link it in the option “Subnet”. Then configure the IP address, subnet mask and router address if needed. See Figure 4.
To add the HMI application to the device go to the catalog on the right side, and drag and drop “WinCC RT Advanced” under “SIMATIC HMI Application”, in the first slot of the device. See Figure 5.
It is possible to rename the device name under properties, it is recommended to name after the name used in netops. See Figure 6.
4.2 Hardware

- Create a local user account with the following login: cern/cern
- Set the computer name to **TP-LLL-XYZZZ**, same as the name declared in netops.
- Switch the Ethernet adapter on DHCP and check that cable is plugged on the matching active socket.
- Go to Control Panel\Network and Internet\Network and Sharing Center. Click "Change Adapter Settings", right-click on the local connection which is active, and select Properties. Select "Internet Protocol Version 4 (TCP/IPv4)" and click "Properties". Select "Obtain an IP address automatically" and "Obtain DNS server address automatically".
- Then try to ping your device using IP address or netops name TP-XXX-YYYY from VM on TN. You should get a response.
- In the control panel, Set the correct PG/PC interface according to the selected canal of communication that you choose for the project (TCP-IP).
- Go to My Computer > properties, select 'remote settings' on the left, and ensure that "allow remote connection from any ..." is selected. This means you can remote desktop to the touch PC from the technical network, using the **TP-LLL-XYZZZ** name, or the IP address.
- Configure the firewall for remote connection in the section 'Advanced settings' and also by creating a new inbound rule on TCP port 2308 which permit you to load HMI project from TIA portal when WinCC RT Loader is waiting for connection or while a project is running (if 'remote control' is enabled in RT Loader).
- The WinCC RT loader runner, should be left in 'Transfer' mode at least the first time, so that you can load the panel remotely, i.e. you should click on 'Transfer', see screenshot below:

![Figure 7 - RT loader](image-url)
5. **CONNECTIONS**

To configure the connection between the PLC and the IPC double click on 'Connections' in the project tree.

Add a new connection (double click on first line) and configure it as follows: choose the name “Connection_1”\(^1\), select the corresponding communication driver (usually *SIMATIC S7 300/400* for Siemens and *Modicon MODBUS TCP/IP* for Schneider). Then configure the communication as type Ethernet and configure the IPC IP address and PLC IP address, slot and rack. See Figure 8.

![Figure 8 – MPI connection configuration](image)

To know more about the communication possibilities check the official Siemens panel documentation on the web:


It has been decided to connect the IPC with an Ethernet cable directly to the PLC to avoid problems in case the switch fails and like this ensure the local supervision.

\(^1\) It is important you call it "Connection_1", if not it won’t fit with the generated file for tags.
6. CREATING A TEMPLATE FOR THE SCREENS

A template is used to display common things to all screens such as the name application, such as navigation buttons or the application name.

To add a new template double click on “Add new template” under “Screen management / Templates” in the Project tree.

Select as background colour dark grey [51,51,51] and use a text field from basic objects to write the application name. See Figure 9.

To avoid choosing the template in each screen, you can set it as default in “Runtime settings” under the project tree.

It is strongly recommended to place the navigation buttons on the template and have a separate template for CCC screens with CCC navigation buttons.
7. **ADDING SCREENS**

To add new screens double click on “Add new screen” under Screens in the Project tree. The screens will be the same ones as in the WinCC OA application (Overview, CCC alarms, Commands...).

Choose a name for your screen, same background colour as before (dark grey [51,51,51]) and the template created before. See Figure 11.

![Figure 11 - Screen configuration](image)

Do the same for the rest of the screens.
8. CREATE BACKGROUND

You can take a screenshot from the panel in PVSS or ask the project responsible to provide it.

The background colour must be changed from “unSynopticBackground” to “[51,51,51]”. Widgets must be removed to leave space to place Touch Panel widgets.

To add a background taken from WinCC OA, you need to make accessible the folder where the screenshots are saved as follows: in the Toolbox menu, under Graphics, right click on “My graphics folder” and click on ‘Link...’, pick up the desired folder where the images are stored. Now you can drag and drop the images inside the folder on your screens to have backgrounds. See Figure 12.

![Figure 12 - New folder link](image-url)
9. GENERATION AND IMPORTATION OF TAGS/TEXT LISTS/ALARMS/SCRIPTS

As mentioned in section 2, **UAB Bootstrap** is needed. Run UAB Bootstrap, choose **cpc-wizard** and open UAB project of the panel application if it is not already opened.

![Opening UAB application project](image)

Figure 13 - Opening UAB application project

In the UNICOS Generators window choose “Touch Panel Instance Generator” and click on “Next”. See Figure 14.

![UNICOS Generators window](image)

Figure 14 - UNICOS Generators window

Select the appropriate platform for your touch panel, in this case, since we are using a Siemens IPC, the platform will be TIA Portal. To generate the importation files (alarms, tags, scripts and text lists) of all the objects click on “Select All” and then click on “Generate”.

Select option “Open Existing Application”
Select the path where the project is located
Figure 15 - WinCC Flexible Instance Generator

You can find the generated files in “...\UAB_Project\Output\TouchPanelGenerator\TIAPortal”. The files are:

- TIAPortal_alarm.xlsx: this file is used to create alarm variables, useful if you are using an alarm list in the panel.
- TIAPortal_script.txt: this text file contains the script used to assign names/units to the tags.
- TIAPortal_securitySettingsScript.txt: this text file contains the script used for the security settings.
- TIAPortal_tags.xlsx: this file contains all the tags definitions.
- TIAPortal_textLists.xlsx: this file is used to import the text lists used for the PCO option modes and WS with pattern.

9.1 TAGS

To import the tags double click on ‘HMI tags > Show all tags’ to open the tags window. Click on the button “Import” and choose the tags file from the UAB project.
Tags will be created into different tag tables as you can see in Figure 17.

![Figure 17 - Tag tables](image)

It is necessary to configure the connection before importing as described in chapter 5, or there will be warnings on the importation and non-internal variables won’t be imported.

You can take a look at the required tags at the end of this document in chapter 15.

### 9.2 TEXT LISTS

Open the text list menu under the Project tree in ‘Text and graphic lists’, and click on the Import button. Choose the directory where the text lists file has been generated under the UAB project.

![Figure 18 - Importing text lists](image)
9.3 **ALARMS**

To open the alarms double click on “HMI alarms” in the *Project tree*. Select *Discrete alarms* and click on *Import*, select the output file for the alarms from the UAB project. This is just useful if you are using an ‘Alarms list’ in the panel (Toolbox > Controls > Alarm view).

![Figure 19 - Importing alarms](image)

9.4 **SCRIPTS**

Two scripts are generated, one of them is used for security issues, and the other one is used to assign names and units as text to the tags of type *string*.

To add a new script in the project go to the Project tree and under ‘Scripts > VB scripts’ double click on “Add new VB function”. The new script will be opened automatically, open the text generated (*TIAPortal_script.txt*) and copy paste the text in the script between the header and the end. It is recommendable to use meaningful names for the scripts, such as “Naming_Units” and “Security_Settings”.

To check the spelling right click anywhere on the script and click on ‘Check syntax’ or in the dedicated button, if it is correct you will read in the Output “Syntax check successfully completed”. Do the same with the script called *TIAPortal_securitySettingsScript.txt*. For more information about the security settings check chapter 14.2.
It is possible to modify names, expert names, units and description in the script, following the limitations: 40 digits for names, 8 characters for expert names, 10 digits for units and 100 digits for descriptions.

It is necessary to make a call to the script from a panel. Go to the main panel and go to ‘Events > Loaded’. Select the scripts from User Scripts and add it. Like this, the script will be executed when the main panel is loaded.
10. UNICOS LIBRARY

To open UNICOS-CPC6 library open the tab Libraries and click on Open on Global libraries.

You can find the library in your UAB project in the following generic path “\UAB_Project\Baseline”, select it and click on Open (the library is called “ucpc-wincc-tia-vx.x”). Take into account that the library is zipped, so it is necessary to extract it before opening it from TIAPortal.

![Figure 22 - Opening UNICOS-CPC6 library](image)

Now you can start designing the panels.

Please note that the library has been saved in TIAPortal V11, so if you are using TIAPortal V12, it is necessary to upgrade the library. For this, after opening the library as it is described above, right click on it and click on “Upgrade library” (see next figure). It will be saved on a different folder for version V12.

![Figure 23 - Upgrade library](image)
11. PANELS DESIGN

It is possible to drag and drop widgets and/or faceplates, let’s add an Analog Input widget and faceplate to the panel “Overview_” (name “Overview” is reserved for Overview general panel in TIA portal).

Drag and drop “AI widget – cpc6” from the library in the panel and choose the corresponding tags of the desired object under the tag “Interface”. See Figure 24.

Figure 24 - Drag and drop AI widget and choose the corresponding tags

Now drag and drop “AI faceplate – cpc6” and choose the corresponding tags.

You can see that every time a widget or faceplate is used from the global library, it appears in the project library. If it is necessary to modify them, they have to be modify from the project library.

Suggestions:

2. Under Properties > Properties > Miscellaneous, change name to “instanceName_widget” for widget and “instanceName_faceplate” for faceplate. Then you can easily find the widget and faceplate using ‘Find’ (Ctrl+F)

3. To make it quicker you can copy/paste the name of the object (“Name.”) in each field of the faceplate/widget, and just choose the desired tag

Note: every widget must have its corresponding faceplate, except for the ones with no faceplate (Analog Status, Word Status)
Figure 25 - Drag and drop AI faceplate and choose the corresponding tags
11.1 Analog Input/Output widget

To change the format of the Analog input or an Analog output you can modify the widget by right click and clicking on ‘Edit type’, either from the screen (‘Edit faceplate’), either from the project library. Now click on the IO Field used to display and change Format pattern under ‘Properties > Properties > General’ as you want (the ‘s’ at the beginning of the format means signed, instead of using this signed option, you can add one more digit and the sign will be just present when the value is negative)\(^2\). For this example we will configure the widget for format “s##.#”.

![Figure 26 - AI widget modification](image)

Figure 26 - AI widget modification

Note that to access a field of the widget, such as the IO Field, it is necessary to disable the invisible button first, to do this, you can hide its layer from “Layout”, where it is possible to see all the objects from the different layers (see Figure 27).

![Figure 27 - Disable layer](image)

Figure 27 - Disable layer

To keep a track of the different versions, add a comment on the version properties to show the format (i.e. “AI s##.#”). To do this, right click on the modified object and click on “Properties…” (see Figure 28).

---

\(^2\) Advice: zoom in to 400%
Once you’ve finished it is necessary to go to the Project library and right click on the modified object to click on “Release faceplate type”. See Figure 28.

![Figure 28 – Properties / Release faceplate type](image)

Do the same with the faceplate to fit the format.

For a more detailed explanation check chapter 14.5.

11.2 PCO widget/faceplate

PCO objects are unique if they have option modes, because it is necessary to configure the text lists for them.

![Figure 29 - PCO widget/faceplate](image)

Text lists will be generated with **WinCC Flexible Instance Generator**. To import text lists check section 9.2.

Text lists will be accessible in the Project tree in ‘Text and graphics Lists’.
Right click on the PCO widget in the project library and click on *Edit faceplate*. Copy the text list from the ones imported (to select it click on the square at left, see Figure 30) and paste it in the widget *Text list editor* (Figure 31).

![Figure 30 - Text lists imported](image)

As you can see in Figure 31, there are two different windows for the text list, one with the general definition, and another one with the different texts according to the value. These windows can be hidden on the signalised arrows.

Now click on the *IO Field* in the widget (see Figure 32) and select the text list just defined in the field *Text list*.

Note that “displayName” will be used on the top of the widget (see Figure 32) to show the name of the object. By default, this text is limited to 8 characters following the recommendation on the specification file, if the PCO name is longer you can either modify it manually on the script or modify the field where it is displayed for having more characters.

---

3 **NOTE:** If there is not a text list for the PCO, there will be a “0” instead of the PCO mode name in the screen, so the best option is to remove the IO Field from the widget. The solution for the faceplate is to place a square over the IO Field.
Do the same with PCO faceplate for the IO Fields marked in Figure 33.

Do not forget to release the widget/faceplate by right click on them and then clicking on ‘Release faceplate type’.

It is necessary to create as many widgets/faceplates as types of option modes. PCOs using same text lists for option modes can use same widget/faceplate.

To keep a track of the different versions check the explanation in chapter 11.1.

This can easily be tested from the “Commands” screen where all PCOs are placed. **Trick:** to avoid configuring the tags on each screen, once they are configured, copy/paste widget and faceplate where they are necessary (they will keep tags configured).
11.3 Word Status widget

If a Word Status with a pattern is used in the application, the widget used for this object should be “WS_text widget – cpc6”. The pattern must be created as a text list in the widget, so as happens with PCOs, one widget per WS is needed in this case.

Copy/paste the widget in the library project with a different name, and then edit the widget with right click over the widget and clicking on “Edit faceplate”. To configure the text list click on the tab “Text lists”, and then double click to add a new one. Give a name to the text list (using the WS name from the specification file is recommended) and configure it with the defined pattern, see Figure 34.

![Figure 34 - New text list on WS widget](image)

Now assign the test list to the IO field of the widget. You can see an example in Figure 35.

![Figure 35 - Configuring text list for WS widget](image)
11.4 Simulation

We can simulate the panel by clicking in the button ‘Start simulation’ that can be seen in the next figure.

![Runtime system with simulator](image)

Figure 36 - Runtime system with simulator

A simulation of the touch panel screen is made, see Figure 37.

![Panel simulation](image)

Figure 37 - Panel simulation
11.5 Layers

It is very useful to use layers for example to hide the faceplates, it is possible to put the faceplates in different layers and then activate/deactivate them. To make visible/invisible the different layers and see the distribution of the objects go to 'Layout' on the right part of the screen. See Figure 38.

Figure 38 - Layers
12. DOWNLOADING THE APPLICATION

There are several options to download the application to the touch panel, in this procedure it is shown how to transfer the project via Ethernet.

12.1 Transfer via Ethernet

If the IPC is connected to the network you can download the application via Ethernet.

In the IPC, choose the option “Transfer...” on the WinCC flexible loader.

On TIA portal choose the mode “Ethernet”, by default the IP address defined in the Device configuration will be chosen, it is possible to choose a different one by choosing the option “User other IP”. Once the connection has been established, it is possible to click on the button “Load”.

Figure 39 - Transfer from TIA portal

Figure 40 - Transfer of the application via Ethernet
13. LICENSE

In order to avoid warning messages about the number of Power Tags, you need to transfer the RT license that was purchased with the touch panel to the touch panel itself. The license comes on a USB key in the box with the touch panel. Connect the USB key to the touch panel, and open the WinCC Automation License Manager (there should be a shortcut on the desktop) and then follow the instructions to transfer the license key from the USB key to the touch panel.
14. MISCELLANEOUS

14.1 Screen navigation

To navigate through the different screens, it is recommended to use buttons, either in the application template, either in each screen.

Use the basic button, available in the toolbox under “Elements”. Use as many buttons as screens and try to expand them to fit the screen, see Figure 41.

![Figure 41 - Distribution of navigation buttons](image1)

To configure the navigation of each button, click on the desired button and go to Properties, in the tab Events, on “Click” choose the function “ActivateScreen” and choose the desired screen.

![Figure 42 - Configuration of navigation buttons](image2)

To indicate the active screen, it is possible to change the text colour on the button that points to the active screen. See Figure 41, where the active screen is ”Overview”.
14.2 Security settings

To configure security settings first you have to set users. In the Project tree open ‘User Administration’ with double click. Add the desired user name and its password with double click on <Add new>. See next figure.

Two tags are necessary for the login/logoff, they are created on the importation file in UAB. These tags are: `userName` and `userLogged`.

We need the script mentioned in chapter 9.4 to check if the user has logged in. Give a meaningful name such as “SecuritySettings”; follow the steps in chapter 9.4.
Now, to have the Login/logoff buttons accessible from all the screens we are going to place them in the template. Open the template and drag and drop two buttons from ‘Toolbox > Elements’ and place them in the desired place, we will place them in the right top corner, leaving some space to display the user name, as you can see in next figure.

![Figure 45 - Placing login/logoff buttons](image)

**Configuration of login button**: in ‘Properties > general’, write ‘Login’ in the field ‘Text OFF’.

![Figure 46 - Writing text for buttons](image)

On ‘Properties > Events > Click’, choose the function ‘ShowLogonDialog’, which will display the log on dialog in the panel pressing the button.
Configuration of logoff button: in ‘properties > general’, write ‘Log off in the field ‘Text OFF’. On ‘Properties > Events > Click’, choose the function ‘Logoff’, which will log off the user pressing the button. See previous figures.

We will display the name of the user logged in all screens by adding a TextField and a IO Field to the template. In the TextField (Tools > Basic objects) we write ‘User:’ in ‘Properties > General’. It is also possible to change the font of the text in ‘Properties > Text’ and the colour in ‘Properties > Appearance’.

You can find the IO Field in ‘Tools > Elements. Drag and drop it and choose ‘Output’ as ‘Mode’, ‘String’ as ‘Format type’ and the created tag “userName” as ‘Process tag’.

To run the script and get the user name we will use the Scheduler. Open it in the Project tree ‘Scheduled tasks’. Double click on the first line to add a job. Choose a name for it (“Login” for example). In the Trigger field choose ‘User change’. Under Events add the function GetUserName and choose tag “userName” as Tag(Out). Now add function SecuritySettings. See next figures.
To disable objects in the screens you can place an invisible button over them. Take the button from ‘Toolbox > Elements’. In the ‘Button mode’ choose ‘Invisible’, place it in layer 31 so it is over everything else, and make as big as you need to cover all the objects except for the login/logoff buttons.

Place the invisible button covering all widgets on each screen as in Figure 55.

Figure 53 - Enable invisible button

Figure 54 - Visibility of invisible button

Figure 55 - Invisible button
14.3 Alarm list

It is possible to add an alarm list to display all the active alarms as the one showed in next figure.

Figure 56 - Alarm list

You can find it in the Toolbox under Controls, its name is ‘Alarm view’.

Alarms will be generated on the UAB project, see chapter 9.3.

14.4 Updating the library

If a new version of the library is released while you are creating your application it is possible to update the objects. You can copy the desired widgets/faceplates from the library and paste them in the project library. You will get a window like the one in next figure, choose the option ‘Update existing type’ and click Ok.

Figure 57 - Updating objects
14.5 Modifying widgets/faceplates locally

This can be very useful for objects with Analog input/outputs, see chapter 0.

As an example we will explain how to change the number of digits in an Analog input widget. Go to the project library, copy/paste AI widget and rename it.

For the example we will rename the widget as “AI 2.3”. After changing the name open the faceplate (right click > edit faceplate).

Now click on the IO field used to show the PosSt information and change the Format pattern to the desired one, in this case “s99.999”, so it is signed with two digits for the integer part and three digits for the decimal part.

![Figure 58 - Changing the format type](image)

This changes the size of the IO field, so it is necessary to rearrange the fields to make it look like in next figure. For this just move the IO field used for units so it is not behind the PosSt field, and also move the letters F and h so they get placed in the bottom right corner of the PosSt field, see Figure 59.

![Figure 59 - Widget after rearranging fields](image)

Do not forget to make the same with the AI faceplate so they are consistent.

![Figure 60 - AI 2.3 decimals faceplate](image)
14.6 Find/replace object names and tags

It is recommended to assign the name of the object on the widget or faceplate, to make easier to recognise the object on the layout or to make possible to find it. To assign the name, you can do it under Properties, on Miscellaneous, as you can see on Figure 61.

![Figure 61 - Assign an object name](image1)

It is possible to look for an object name on a screen with “Control+F” or on “Edit, Find and replace”, you can see the option window on Figure 62. This can be useful for similar projects where the touch panel application can be reused, names and tags could be replaced following the particular pattern.

![Figure 62 – “Find and replace” tool](image2)
To find/replace object tags on a screen you can use the tool “Change object reference”. To do it on a screen, click somewhere on the screen background (like this all the objects on the screen will be taken into account), and then click on “Tools, Change object reference”.

A window will be opened, where you will be able to specify the reference you want to change and with which name you want to replace it. See Figure 63.

Figure 63 - "Change object reference" tool
14.7 Problems on screen change speed

It has been found that there are problems on changing from one screen to another on the touch panel run time. This is the solution proposed by Siemens to avoid the delay provoked by this problem (problem will be solved in future versions of the software).

- Create a dummy button in the screen having the delay, configured as “Invisible”, see Figure 64. Call it “dummy_button”

![Figure 64 - dummy_button](image)

- Create a script called “FastScreenChange” to set the TabStopID on the dummy button. Write the following text on the script (on the example it is just used for the screen “General”):

```plaintext
' Workaround in TIA Portal V12 SP1 (and lower)
' for slowly screen changing
On Error Resume Next
If HmiRuntime.ActiveScreen.ObjectName = "General" Then
    HmiRuntime.Screens("General").ScreenItems("dummy_button").Activate
End If
```

![Figure 65 - FastScreenChange script](image)
If the solution was implemented for several screens, it would look like this:

' Workaround in TIA Portal V12 SP1 (and lower)
' for slowly screen changing
On Error Resume Next
If HmiRuntime.ActiveScreen.ObjectName = "Screen_1" Then
    HmiRuntime.Screens("Screen_1").ScreenItems("dummy_button").Activate
End If
If HmiRuntime.ActiveScreen.ObjectName = "Screen_2" Then
    HmiRuntime.Screens("Screen_2").ScreenItems("dummy_button").Activate
End If

- Call the script on each navigation button on the screen, before changing the screen. See next figure for the configuration of the navigation button.

![Figure 66 - Navigation button (screen changing speed problem)](image)

14.8 Message “Too many tags configured”

When running the RT on your computer, you might get the message showed on Figure 67, pointing that the license doesn’t cover the number of tags used in the project.

This is because no license has been activated for running the RT, just the license for TIAPortal. This message can be ignored on your PC, by clicking on “Noted”.

A RT license must be bought and activated on touch panels, so this message should not appear on production applications.

![Figure 67 - Too many tags have been configured](image)
14.9 Remote desktop connection

To connect remotely to the touch panel with Windows tool “Remote Desktop Connection”, you can write the name used in Netops (network.cern.ch) and connect under user name “cern”, as shown in Figure 68. Note that the touch panel execution will be interrupted for the user, so check first that it is not being used.

![Remote Desktop Connection](image.png)

Figure 68 - Remote connection to touch panel

It is recommended to do it with the good resolution according to the panel (this can be checked in the TIA portal project under Runtime settings).

User name and password are the ones defined on chapter 4.2.
14.10 General advice

- Not placing things out of the screen, they won’t be displayed, a warning will be produced in compilation, see Figure 69.

![Figure 69 - Invalid position](image)

- Save often!!! The software is prone to crashing and you can lose all your work

- Use the buttons showed in next figure, they are very useful for arranging and placing objects

![Figure 70 - Placing/arranging objects](image)

- **Trick**: to avoid configuring the tags of an object on each screen it is used, once they are configured, copy/paste widget and faceplate where they are necessary (they will keep tags configured).

- To select several objects on a screen you can do it by using “Shift”.
15. **DIFFERENT RUN TIME VERSION**

The run time (RT) version on your touch panel device doesn't necessarily have to match with the TIAPortal version used to develop the application. It is strongly recommended to use the latest TIAPortal version.

If RT version in the touch panel device is inferior to TIAPortal, it is possible to change RT version from the *Device configuration* in the application: open the *Device configuration* from the *Project tree*, select the RT module and click on *Change device/version*.

![Image of device configuration](image)

**Figure 71 - Change RT version**

In the window for changing the device, select on the right side WinCC RT Advanced, and then you will be able to select the desired RT version. See next figure.
Figure 72 - Change RT version options
16. **USE CASE: SEVERAL PLC CONNECTIONS**

In the case where several PLCs have to be connected to the touch panel application, it is necessary to create several connections in TIAPortal and modify manually the importation file for tags.

For creating several **connections**, follow the normal procedure: double click on Connection line and assign a significant name (e.g. Connection_A), do the same with the rest of the connections (e.g. Connection_B, Connection_C, etc).

For the manual modification of the **tags** importation file, open the file `TIAPortal_tags.xlsx` that you can find on your UAB project under “...\UAB_project\Output\TouchPanelGenerator\TIAPortal”, and “Find/replace” the default connection name “Connection_1” by the desired name, matching with the name chosen in the touch panel application (Connection_A, Connection_B, Connection_C, etc). For more information on tag importation check 9.1

![Figure 73 - Example of several connections](image)

For **alarms** file, it is a bit more complicated. Alarms have an ID number assigned, if two alarm files are imported with same ID numbers, alarms will be overwritten on importation on TIA portal. To fix this, check the highest number on one of the files (e.g. 20 alarms), and open the other one to modify it making alarms number start by next number to the highest on first file (e.g. 21) and going up. Check Figure 74 to get a better idea. For more information on importing alarms check chapter 9.3.
Take into account that both scripts have to be run on TIA portal, so both texts can be pasted to the same script and be called once, or two independent scripts can be created and the call has to be made for both of them. For more information on scripts check chapter 9.4.

Note that it is also necessary to import both text lists. For more information about this check chapter 9.2.

17. REQUIRED TAGS

17.1 Analog alarm

- AlarmName.visible (internal tag, bool)
- AlarmName.objName (internal tag, string)
- AlarmName.stsReg01
- AlarmName.stsReg02
- AlarmName.manReg01

17.2 Analog Digital

- AnaDigName.visible (internal tag, bool)
- AnaDigName.setValue (internal tag, bool)
- AnaDigName.objName (internal tag, string)
- AnaDigName.unit (internal tag, string)
- AnaDigName.stsReg01
• AnaDigName.stsReg02
• AnaDigName.manReg01
• AnaDigName.PosSt
• AnaDigName.MPosR

17.3 Analog Input
• AIname.visible (internal tag, bool)
• AIname.objName (internal tag, string)
• AIname.unit (internal tag, string)
• AIname.setValue (internal tag, bool)
• AIname.stsReg01
• AIname.manReg01
• AIname.PosSt
• AIname.MPosR

17.4 Analog Input Real
• AIRname.visible (internal tag, bool)
• AIRname.objName (internal tag, string)
• AIRname.unit (internal tag, string)
• AIRname.setValue (internal tag, bool)
• AIRname.stsReg01
• AIRname.manReg01
• AIRname.PosSt
• AIRname.MPosR

17.5 Analog
• AnaName.visible (internal tag, bool)
• AnaName.objName (internal tag, string)
• AnaName.unit (internal tag, string)
• AnaName.setValue (internal tag, bool)
• AnaName.stsReg01
• AnaName.stsReg02
• AnaName.manReg01
• AnaName.PosSt
• AnaName.MPosR

17.6 AnaDO
• AnaDOname.visible (internal tag, bool)
• AnaDOname.objName (internal tag, string)
• AnaDOname.setValue (internal tag, bool)
• AnaDOname.stsReg01
• AnaDOname.stsReg02
• AnaDOname.manReg01
• AnaDOname.PosSt
• AnaDOname.MPosR

17.7 Analog Output
• AOname.visible (internal tag, bool)
• AOname.objName (internal tag, string)
• AOname.unit (internal tag, string)
• AOname.setValue (internal tag, bool)
• AOname.stsReg01
• AOname.manReg01
• AOname.PosSt
• AOname.MPosR

17.8 Analog Output Real
• AORname.visible (internal tag, bool)
• AORname.objName (internal tag, string)
• AORname.unit (internal tag, string)
• AORname.setValue (internal tag, bool)
• AORname.stsReg01
• AORname.manReg01
• AORname.PosSt
• AORname.MPosR
17.9 Analog Parameter
- APARname.visible (internal tag, bool)
- APARname.objName (internal tag, string)
- APARname.unit (internal tag, string)
- APARname.setValue (internal tag, bool)
- APARname.manReg01
- APARname.PosSt
- APARname.MPosR
- APARname.MPosRSSt

17.10 Analog Status
- ASname.unit (internal tag, string)
- ASname.PosSt

17.11 Digital Alarm
- DAname.visible (internal tag, bool)
- DAname.objName (internal tag, string)
- DAname.stsReg01
- DAname.manReg01

17.12 Digital Input
- DIName.visible (internal tag, bool)
- DIName.objName (internal tag, string)
- DIName.stsReg01
- DIName.manReg01

17.13 Digital Output
- DOname.visible (internal tag, bool)
- DOname.objName (internal tag, string)
- DOname.stsReg01
- DOname.manReg01

17.14 Digital Parameter
- DPARname.visible (internal tag, bool)
- DPARname.setValue (internal tag, bool)
• DPARname.objName (internal tag, string)
• DPARname.stsReg01
• DPARname.manReg01

17.15 Local
• LocalName.visible (internal tag, bool)
• LocalName.objName (internal tag, string)
• LocalName.stsReg01

17.16 On Off
• OnOffName.visible (internal tag, bool)
• OnOffName.objName (internal tag, string)
• OnOffName.stsReg01
• OnOffName.stsReg02
• OnOffName.manReg01

17.17 Process Control Object
• PCOname.visible (internal tag, bool)
• PCOname.objName (internal tag, string)
• PCOname.setOpMo (internal tag, bool)
• PCOname.stsReg01
• PCOname.stsReg02
• PCOname.manReg01
• PCOname.OpMoSt
• PCOname.MOpMoR
• PCOname.AuOpMoSt

17.18 Controller
• PIDname.visible (internal tag, bool)
• PIDname.objName (internal tag, string)
• PIDname.setSetPoint (internal tag, bool)
• PIDname.stsReg01
• PIDname.stsReg02
• PIDname.manReg01
- PIDname.manReg02
- PIDname.MV
- PIDname.ActSP
- PIDname.ActTi
- PIDname.ActKc
- PIDname.ActTd
- PIDname.ActTds
- PIDname.MSP

17.19 Word Parameter
- WPARname.visible (internal tag, bool)
- WPARname.objName (internal tag, string)
- WPARname.unit (internal tag, string)
- WPARname.setValue (internal tag, bool)
- WPARname.PosSt
- WPARname.MPosRSt
- WPARname.manReg01
- WPARname.MPosR

17.20 Word Status
- WSname.unit (internal tag, string)
- WSname.PosSt
18. **EXAMPLE**

This is an example of a touch panel application. As mentioned before in the tutorial, the structure is based on the one for PVSS.

As in PVSS the default screen is “Overview”.

- **Figure 75 – Overview**
  - Navigation buttons: See section 14.1
  - Security settings: See section 14.2

- **Figure 76 – Commands**
  - Widget and faceplate are placed next to each other
Background can be taken from WinCC OA panels see chapter 8

Figure 77 – Pumps

Figure 78 – Regulation
List of active alarms, see chapter 13.3

Figure 79 - CCC alarms

Figure 80 - Active alarms