New Energy Gateway GUI Design by AppWizard

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code utilizes the SEGGER emWin AppWizard tool to create a graphical user interface (GUI) for a new energy gateway. It is designed to be compatible with the NUC980 series development platform to achieve a user-friendly interface for controlling the new energy gateway.
BSP Version	emWin Software Package for NUC980
Hardware	NuMaker-IIoT-NUC980G2D V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. Overview

This example code demonstrates how to use the SEGGER emWin AppWizard tool and the NuMaker-IIoT-NUC980G2D evaluation board to complete the GUI (Graphical User Interface) interface design and screen interaction of the new energy gateway step by step.

1.1 Principle

1.1.1 Block Diagram

This example code uses the Nuvoton GUI development platform NuMaker-IIoT-NUC980G2D evaluation board, which uses NUC980G2D microprocessor, a 2.4" TFT-LCD resistive touch panel (with a resolution of 240x320), and 1 Gbit SPI NAND Flash. The system block diagram is shown in Figure 1-2.

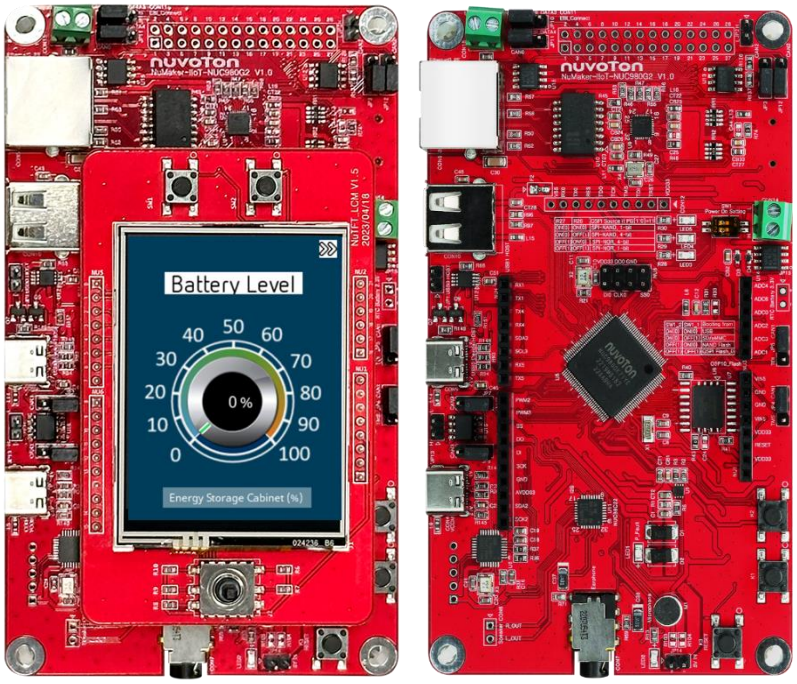


Figure 1-1 NuMaker-IIoT-NUC980G2D

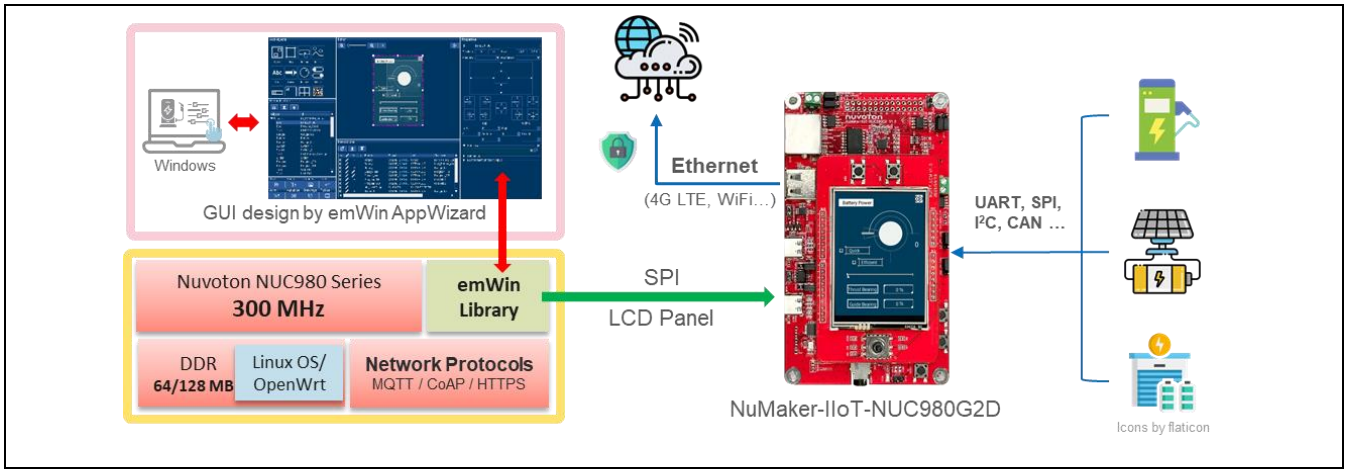


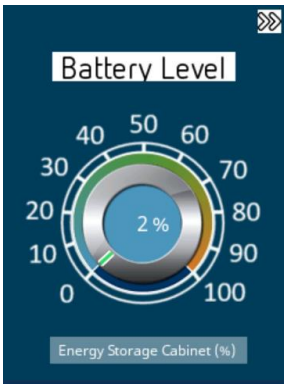
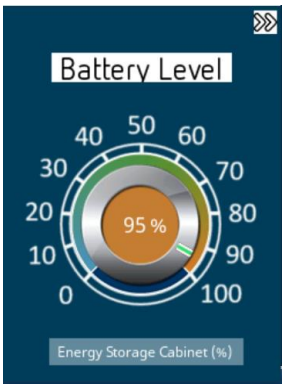
Figure 1-2 System Block Diagram

1.1.2 Supporting Resources

- Nuvoton Forum: Nuvoton emWin Forum
- SEGGER emWin Forum: Segger emWin Forum
- User manual of SEGGER emWin AppWizard: AppWizard User Manual

1.2 GUI Design of New Energy Gateway

The GUI design of this example is shown in Table 1-1.

	Initial Screen	Interactive Effect
Home Screen		

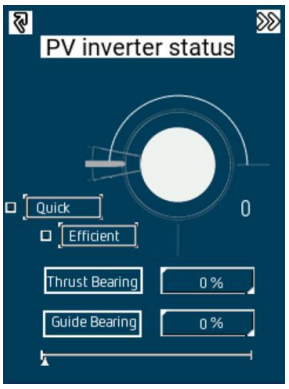
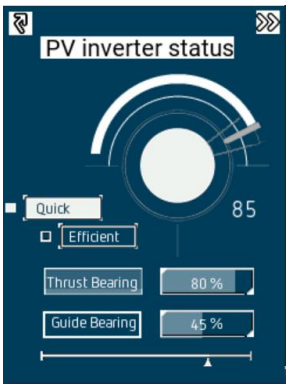
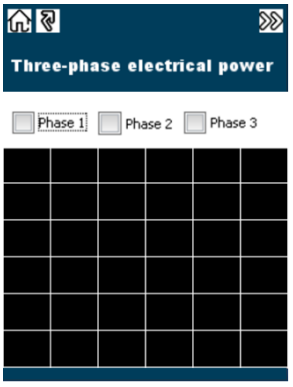
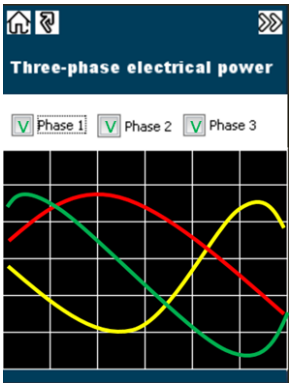
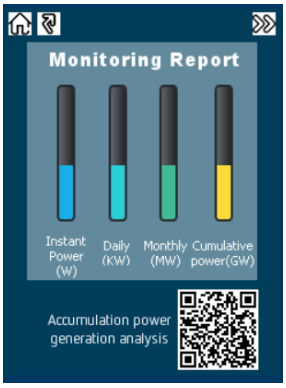
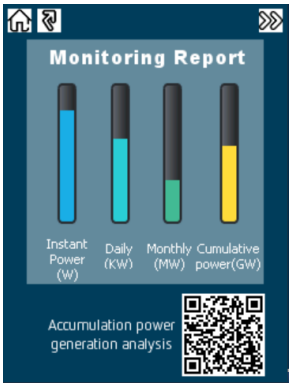
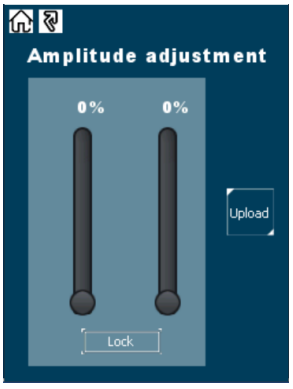
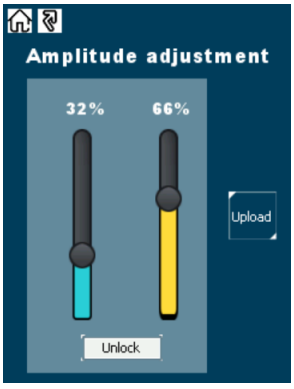
Screen 2		
Screen 3		
Screen 4		
Screen 5		

Table 1-1 GUI Design of New Energy Gateway

2. Execution Steps

Design a GUI project by AppWizard can be divided into four stages. In the first stage, prepare necessary hardware and software. The second stage, design widgets, such as object properties, interactions, and animations in AppWizard.

In the third stage, the generated code from AppWizard is exported, and custom code can be added to enhance additional functionalities. The fourth stage is programming the board. Each of the stages are described in Section 2.1 to Section 2.4.

2.1 Hardware and Software Preparation

1. Refer to EVB Introduction, Schematic,and User Manual for more related information about the NuMaker-IIoT-NUC980G2D evaluation board.
2. Download GUI_Builder and AppWizard software tools.
   - emWin
     - GUIBuilder.exe
   - emWin AppWizard IDE and emWin Package
     - Please download the emWin Package in Nuvoton Website. (Need to Sign-In)
     - Unzip the emWin Package and click the file *AppWizard_Install.exe* to install emWin AppWizard IDE.

The *GUIBuilder.exe* has been included in the example package, and the location of files are shown in Figure 2-1.

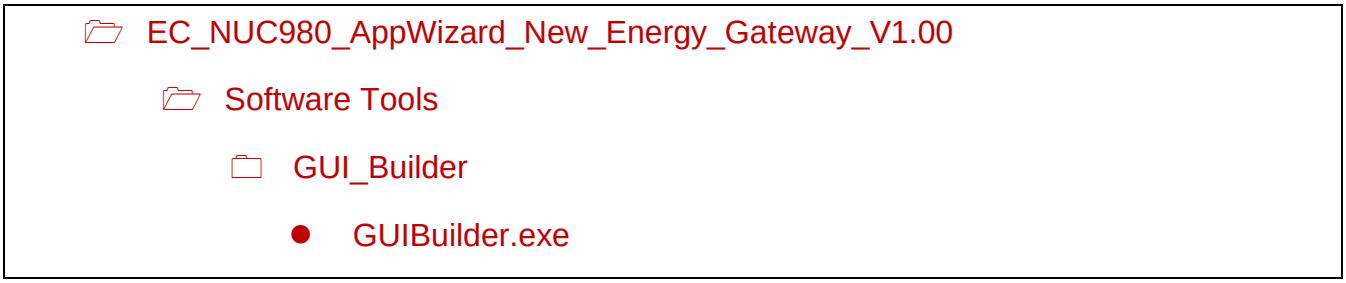


Figure 2-1 Necessary Tools and File Path

3. NuMaker-IIoT-NUC980G2D is developed by Linux environment. Please confirm that the VMWare virtual machine is installed on the computer, or download the NUC980 series dedicated virtual machine through the Nuvoton's GitHub.

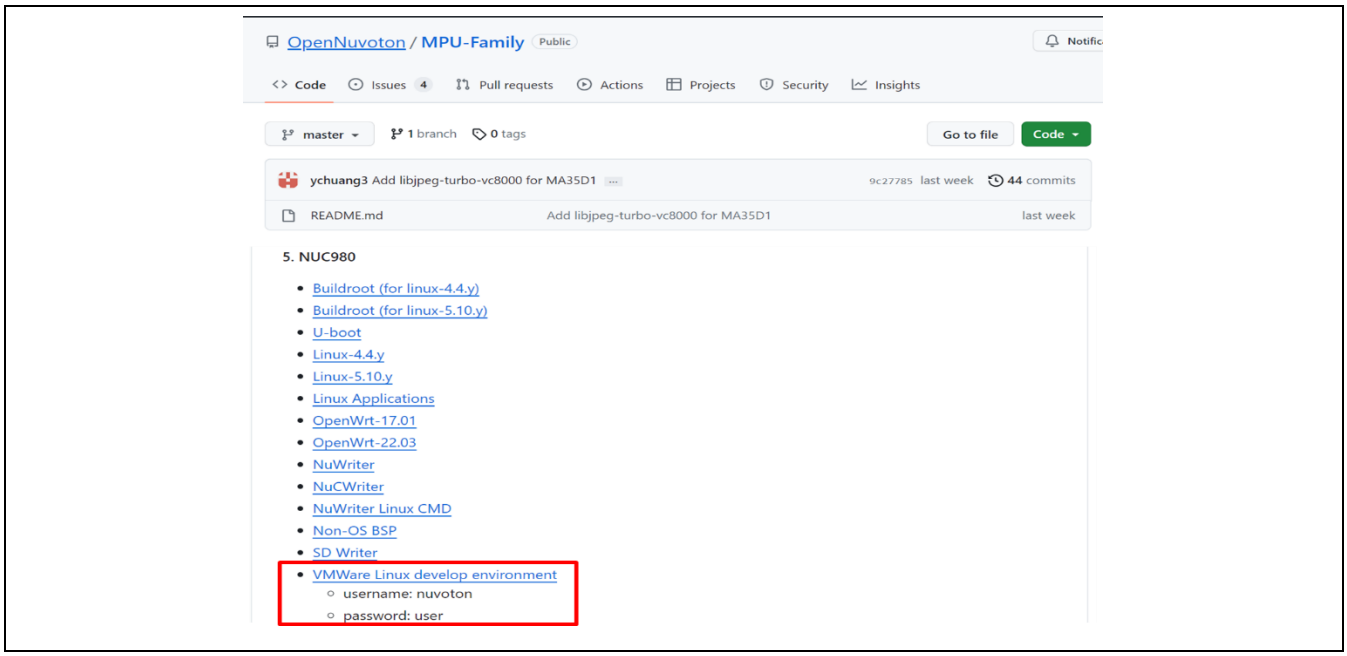


Figure 2-2 VMWare Environment Provided by Nuvoton

4. Before executing this example code, users need to build the NUC980 Linux Kernel (including Buildroot, u-boot, tool chain... and others). Currently NuMaker-IIoT-NUC980G2D provides Linux kernel version 5.10 resources; the detailed compilation process of files are as follows.

   - Open the VMWare provided by Nuvoton, enter the MA35D1_Buildroot folder through terminal, and compile the NuMaker-IIoT-NUC980G2D configuration file "*nuvoton_nuc980_iot_g2d_defconfig*" through the command **make**.

```
users@ubuntu: ~/MA35D1_Buildroot$ make nuvoton_nuc980_iot_g2d_defconfig
users@ubuntu: ~/MA35D1_Buildroot$ make
```

   - After compilation, there will be five images used for downloading:
     - **Image**: Linux kernel image used for running on DDR.
     - **uImage**: Linux kernel image used for booting with uboot.
     - **dtb**: Device tree binary, describe the setting of peripherals in detail.
     - **u-boot.bin**: Uboot image.
     - **u-boot-spl.bin**: Uboot Secondary Program Loader image, used for booting from SPI NAND.
   - The images are generated at the following locations:

- **Image**, **uImage**, and **dtb** are generated at the path "*MA35D1_Buildroot/output/images*".
- **u-boot.bin** is generated at the path "*MA35D1_Buildroot/output/build/uboot-customer*".
- **u-boot-spl.bin** is generated at the path "*MA35D1_Buildroot/output/build/uboot-master/spl*".

5. Since there are several files that need to be programmed, so users need to create an *env.txt* to configure the programmed location and settings of image files. The content of *env.txt is* as follows.

```
baudrate=115200
bootdelay=1
stderr=serial
stdin=serial
stdout=serial
loadkernel=nand read 0x7fc0 0x200000 0x800000
loaddtb=nand read 0x1400000 0x180000 0x20000
bootcmd=run loadkernel;run loaddtb;bootm 0x07fc0 - 0x1400000
```

6. After downloading the programming tool of NUC980, NuWriter, program the image files and *env.txt* to the NuMaker-IoT-NUC980G2D evaluation board through NuWriter. The programming process is as follows:

- Switch **SW4** of the evaluation board to USB boot mode.
- Choose the programming type as **SPI NAND**.
- To erase all blocks.
- Program ***u-boot-spl.bin*** image
  - Image Name: Specify the file path
  - Image Type: Loader
  - Image execute address: 0x200
  - Image start offset: N/A
- Program ***uBoot*** image
  - Image Name: Specify the file path
  - Image Type: Data
  - Image execute address: N/A
  - Image start offset: 0x100000
- Program ***nuc980-iot-g2d-v1.0.dtb*** image

  - Image Name: Specify the file path
  - Image Type: Data
  - Image execute address: N/A
  - Image start offset: 0x180000
- Program ***uImage***
  - Image Name: Specify the file path
  - Image Type: Data
  - Image execute address: N/A
  - Image start offset: 0x200000
- Program **environment** image (***env.txt***)
  - Image Name: Specify the file path
  - Image Type: Environment
  - Image execute address: N/A
  - Image start offset: 0x80000
- After programming is completed, switch SW4 back to Nand Boot mode and reset the evaluation board to use it.

Note: For more details of the Kernel compilation and programming process, please refer to the NUC980 Linux Environment on VMware User Manual and NuMaker-IIoT-NUC980G2D User Manual (Section 5.5 NuWriter Tool).

2.2 Starting the GUI Design by AppWizard

1. Open the AppWizard tool using the desktop shortcut.

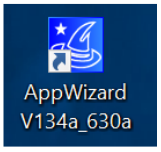


Figure 2-3 AppWizard Desktop Shortcut

2. Click “Create new project” to create a new project.

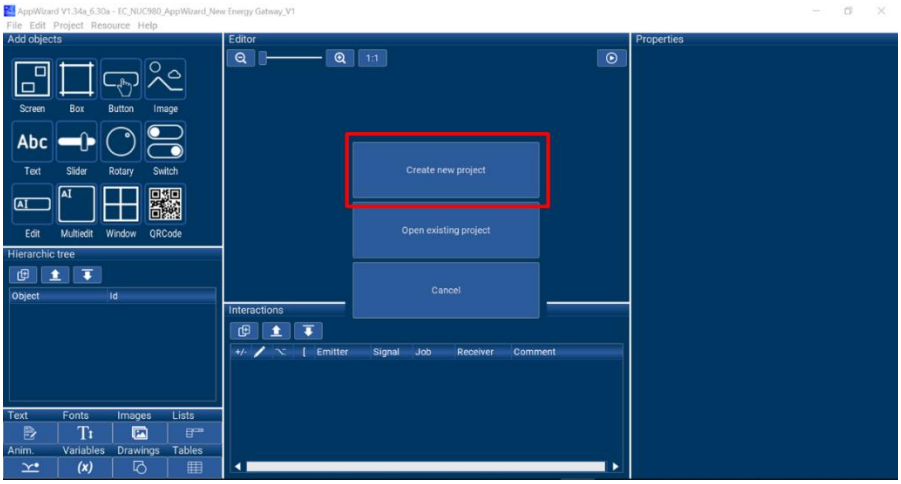


Figure 2-4 Create New Project

3. Edit the project path and project name. Ensure that the color format should match the screen’s specification. According to Figure 2-5, choose “16 Bit, GUICC_M565”, and check “Enable Multibuffering”.

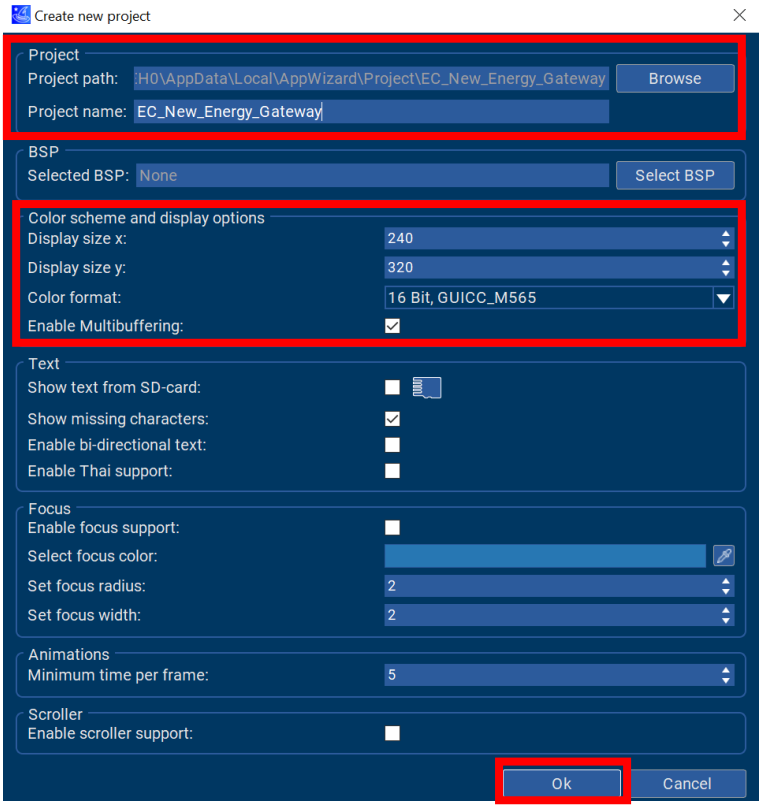


Figure 2-5 New Project Setting

4. Click “Screen” five times under “Add objects” to create five screens. In the Hierarchic tree, name the screens as “ID_SCREEN_Home”, “ID_SCREEN_02” to “ID_SCREEN_05” respectively for easier identification.

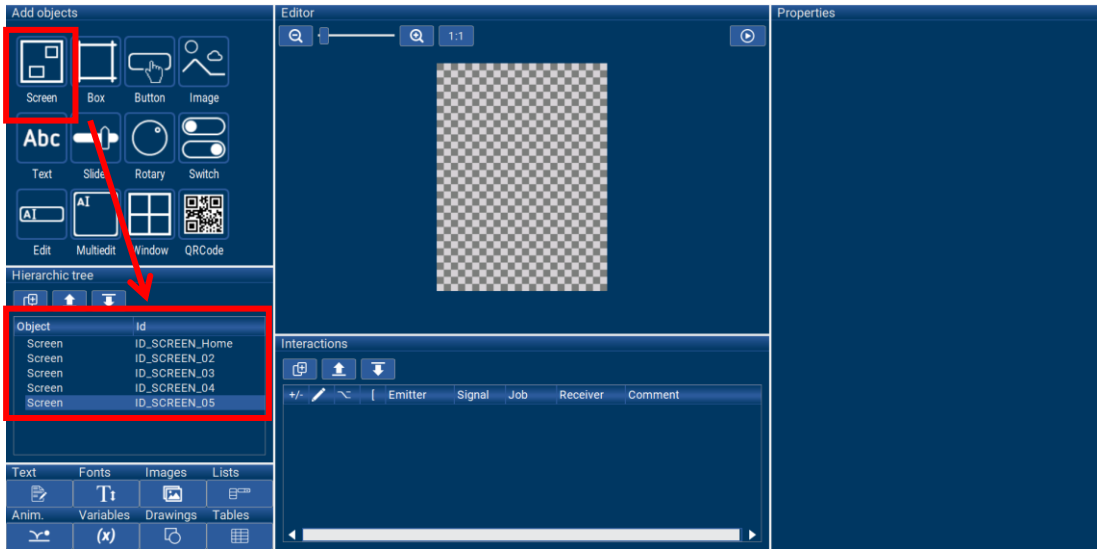


Figure 2-6 Add Screens

5. Click “Images” icon to import all the PNG images used in this project into AppWizard. (File located at: *EC_NUC980_AppWizard_New_Energy_Gateway_V1.00\EC_New Energy Gateway\Resource\Image*)

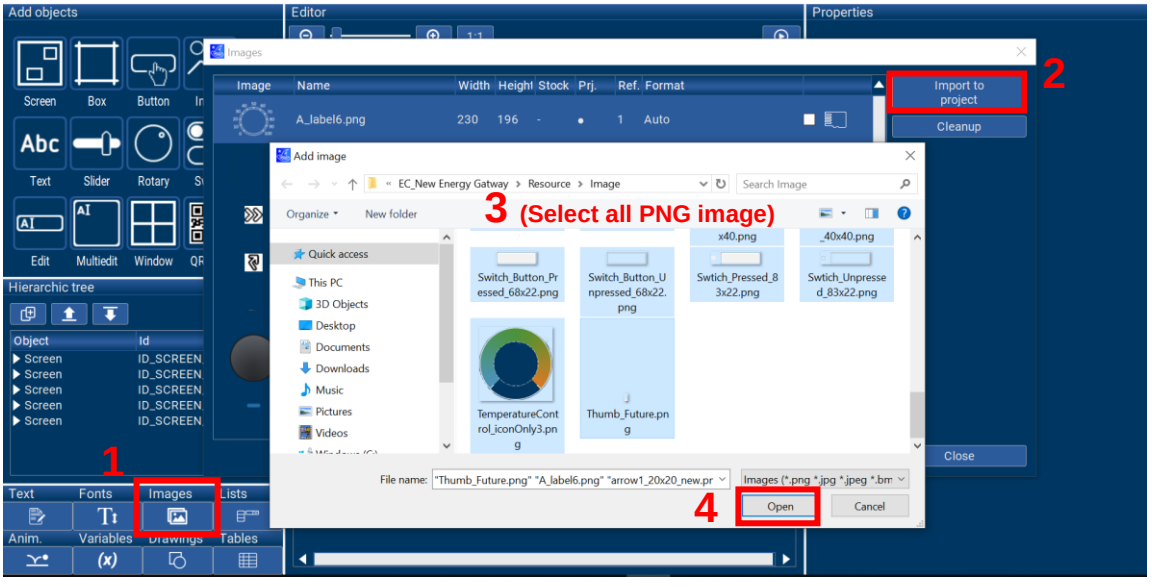


Figure 2-7 Import PNG Images into AppWizard

6. Click ID_SCREEN_Home and ID_SCREEN_02 to ID_SCREEN_05 screens respectively. Under “Add objects”, select “Box” to create a solid color background for each screen.

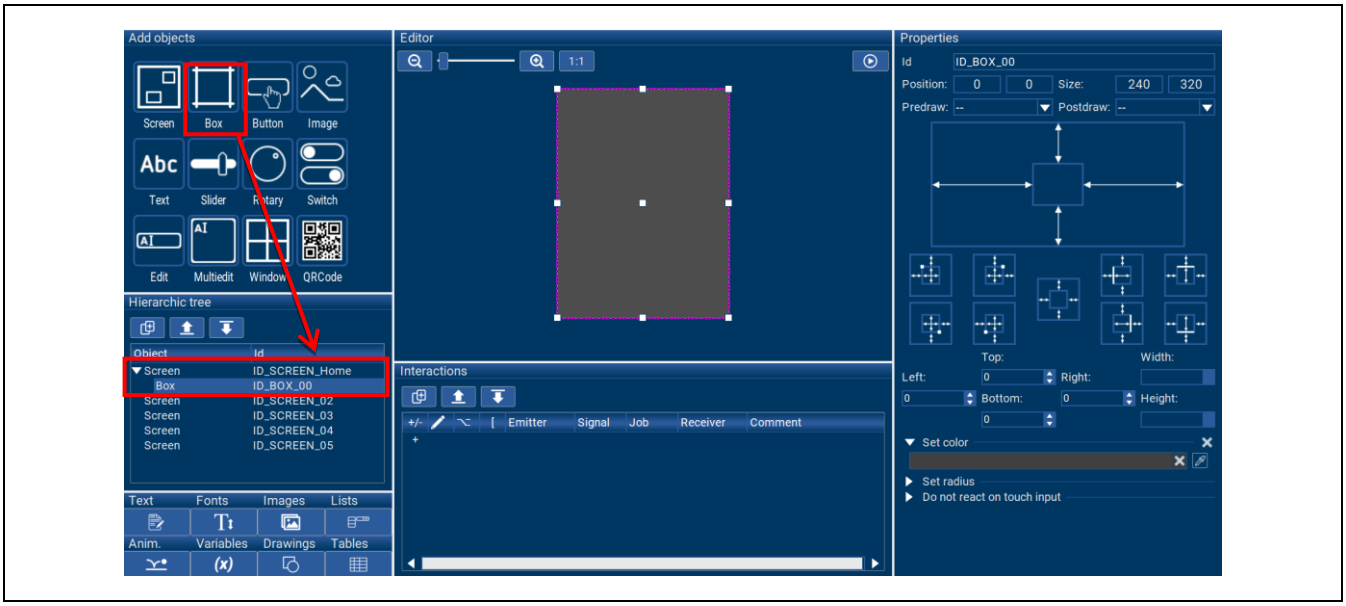


Figure 2-8 Add a Box Widget as Background

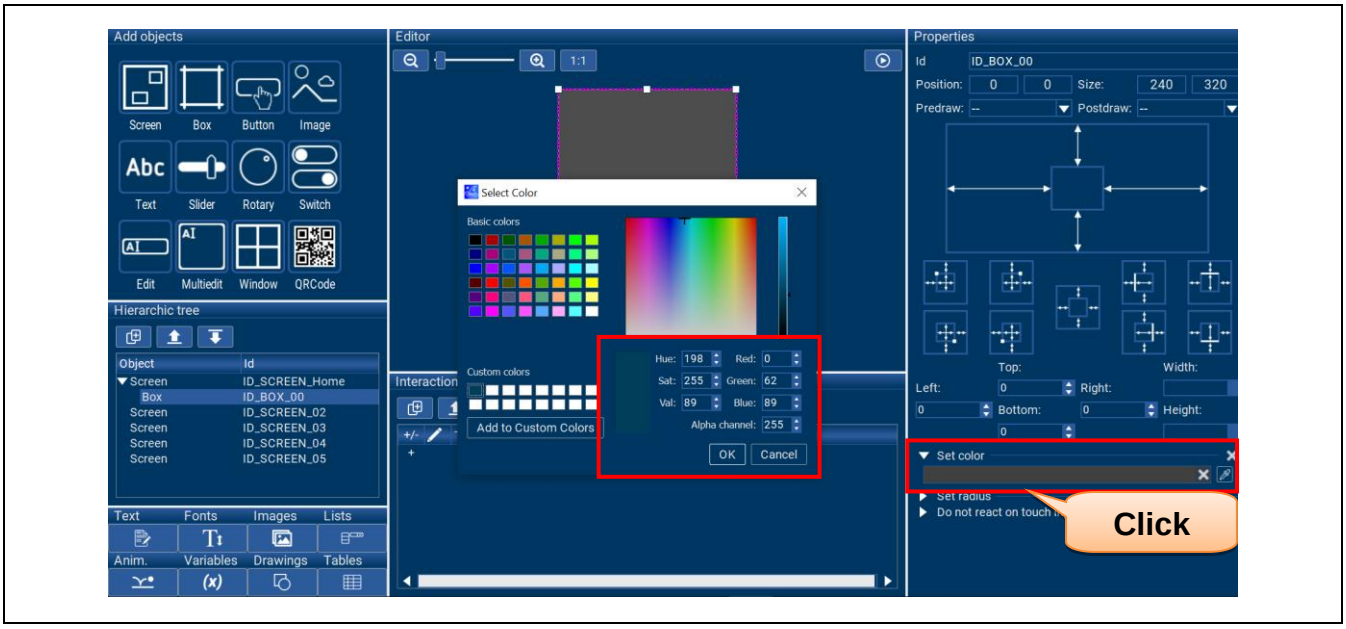


Figure 2-9 Set the Color of Box

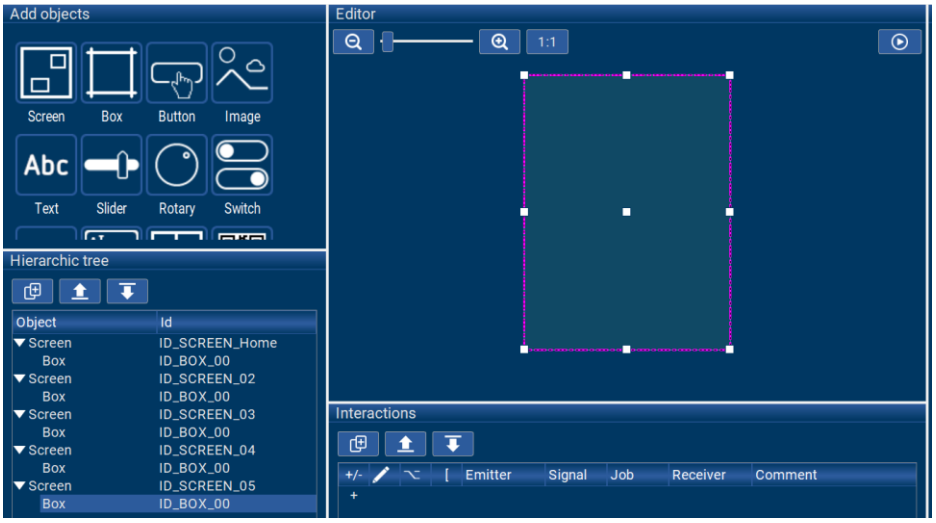


Figure 2-10 Added Solid Background for All Screens

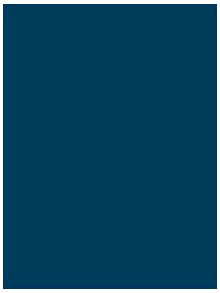
2.3 Implementing Page Design of AppWizard

New Energy Gateway is widely used. Common applications include energy storage, charging piles, photovoltaic inverters, current meters, and other industrial applications.

This example will design a GUI for common applications and describe the design steps, interaction settings, and animations in detail for each screen. Users can adjust this example according to their requirement.

2.3.1 Design of Battery Meter (Home Screen)

The design of the battery meter can be applied to products that need to display power consumption, such as charging piles. The widgets used on this screen are shown in Table 2-1.

New Energy Gateway Description	Application GUI	Widget Type
Page background (240x320)		Box
Background of battery meter (Place below rotaries)		Image

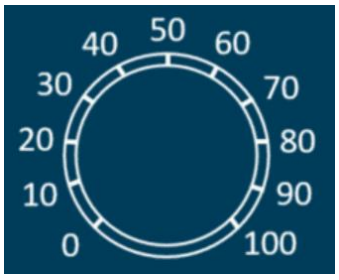
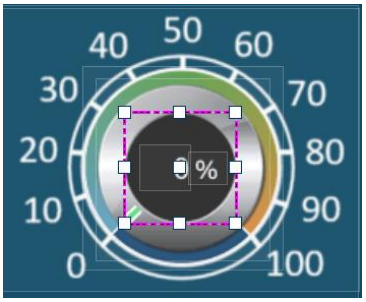
Background of Rotary 1		Rotary
Meter of Rotary 1		
Scale of the battery meter (Bitmap of Rotary 2)		Rotary
Color changing area		Box
Page title	Battery Level	Text
Battery value	0 %	
Page notes	Energy Storage Cabinet (%)	
Button for next page		Button

Table 2-1 Widgets of Battery Meter Page

2.3.1.1 Widgets Configuration of GUI

1. Click on *ID_SCREEN_Home* to start editing the home screen, add an Image object (ID_IMAGE_00) with a size of 123x123 through “Add objects”, and select the image *ColorVariety.png* as the bitmap.

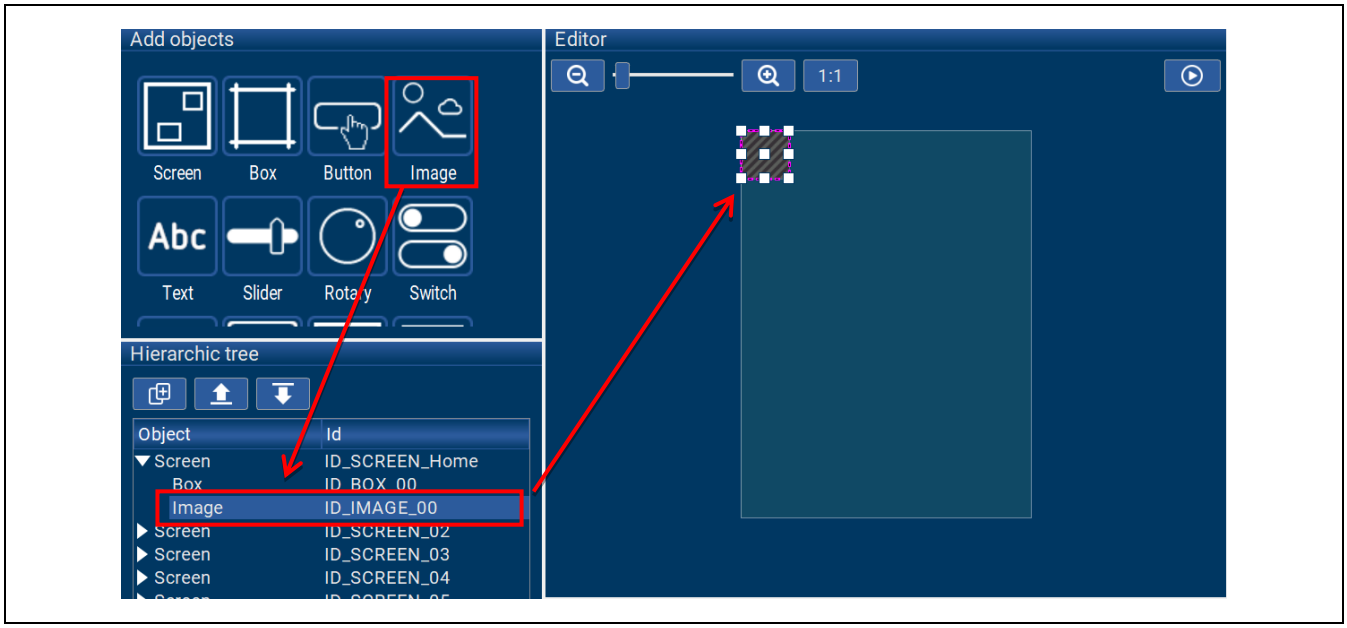


Figure 2-11 Add Image Object

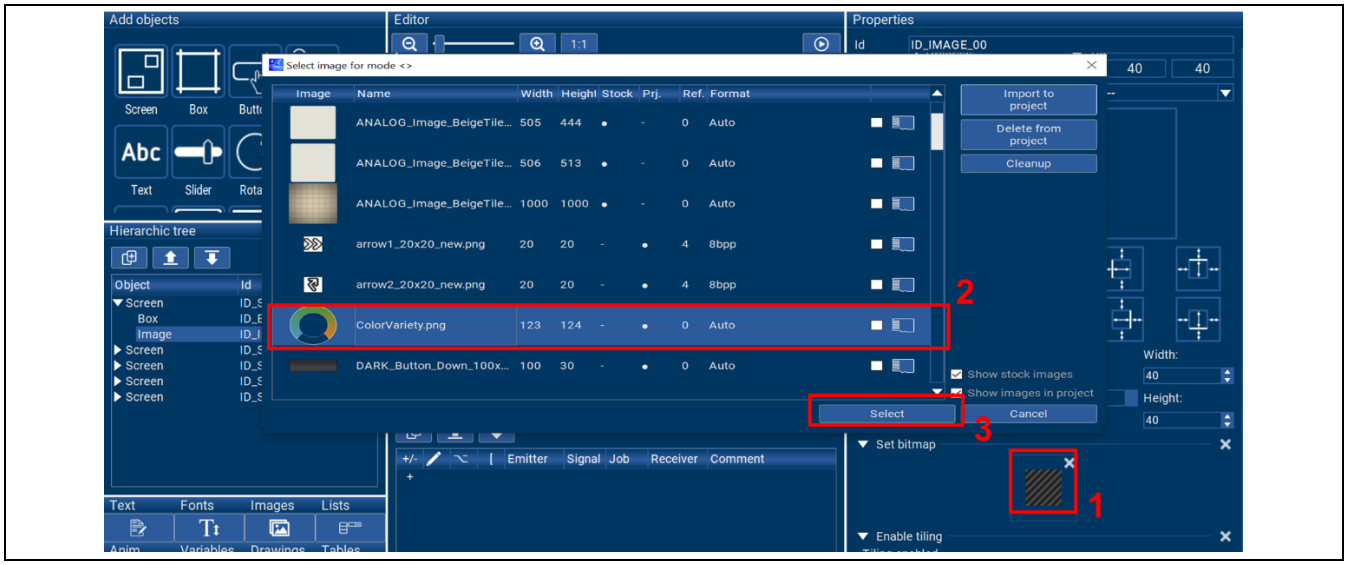


Figure 2-12 Set the Object Bitmap

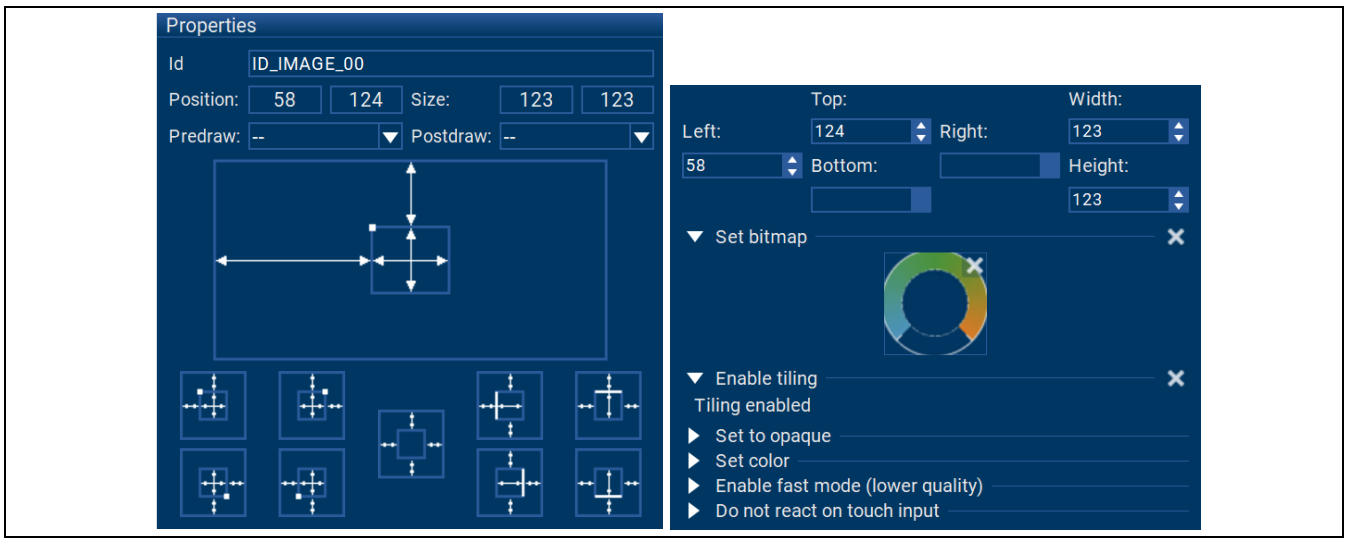


Figure 2-13 Set Properties of ID_IMAGE_00

2. Add a Rotary object (ID_ROTARY_idx) with a size of 219x181 and set it above ID_IMAGE_00. Select the image *scaleValue.png* as the background bitmap and cancel the marker bitmap.

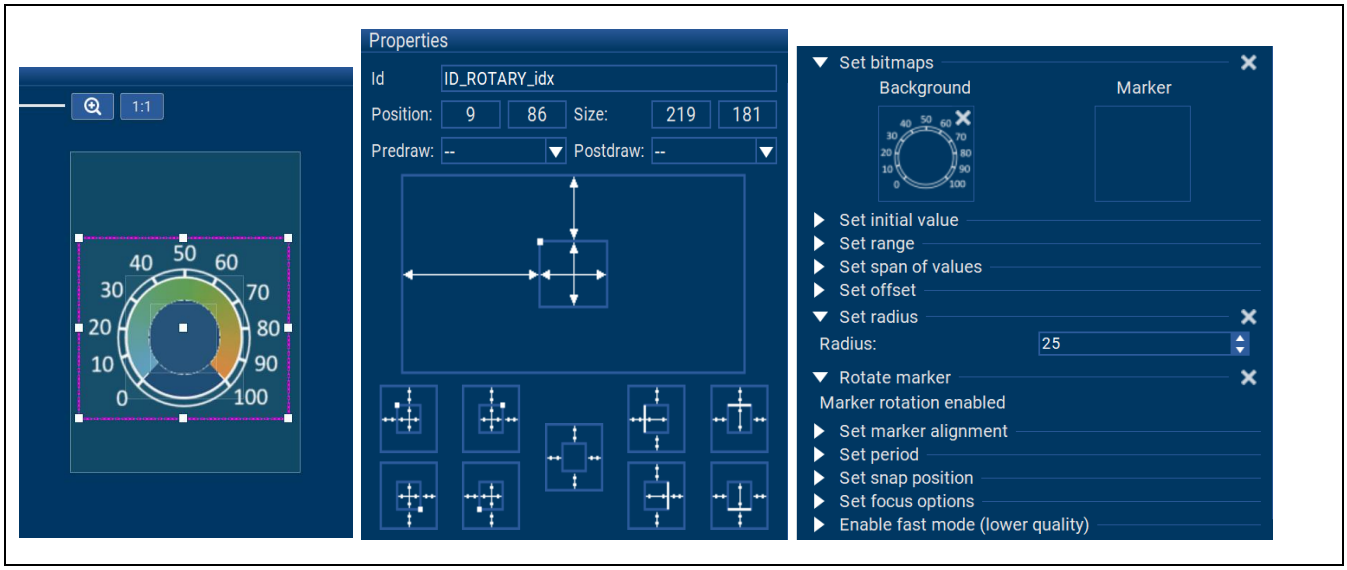


Figure 2-14 Set Properties of ID_ROTARY_idx

3. Add a Box object (ID_BOX_color) with a size of 69x69 as a color-changing area (cancel the default color) and set it above ID_ROTARY_idx.

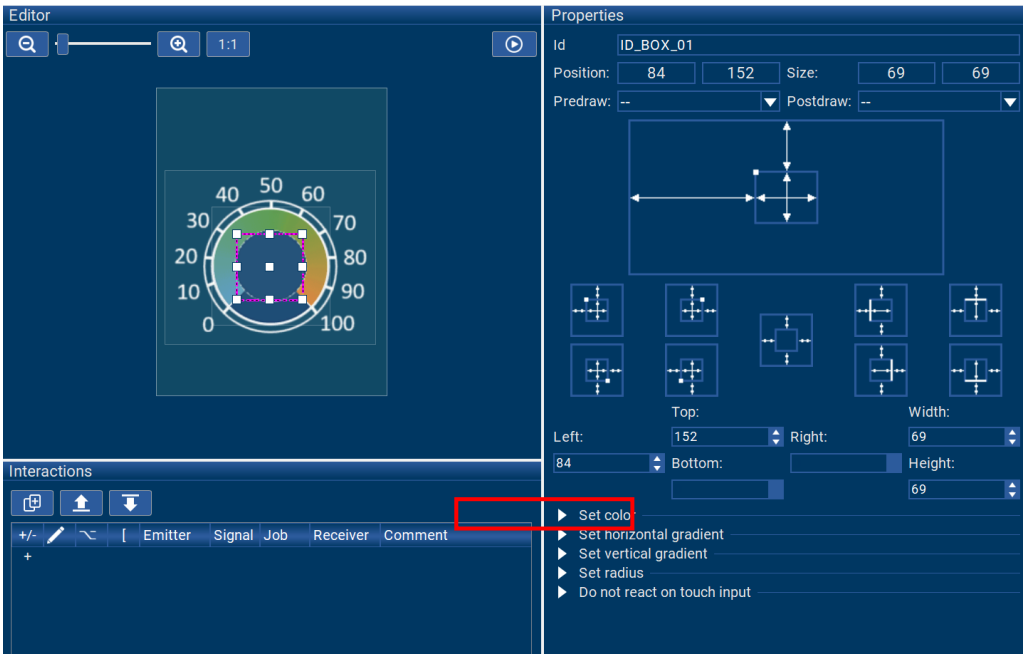


Figure 2-15 Set Properties of ID_BOX_color

4. Add a rotary object (ID_ROTARY_val) with a size of 107x112 and set it above ID_BOX_color. Set the properties according to Table 2-2.

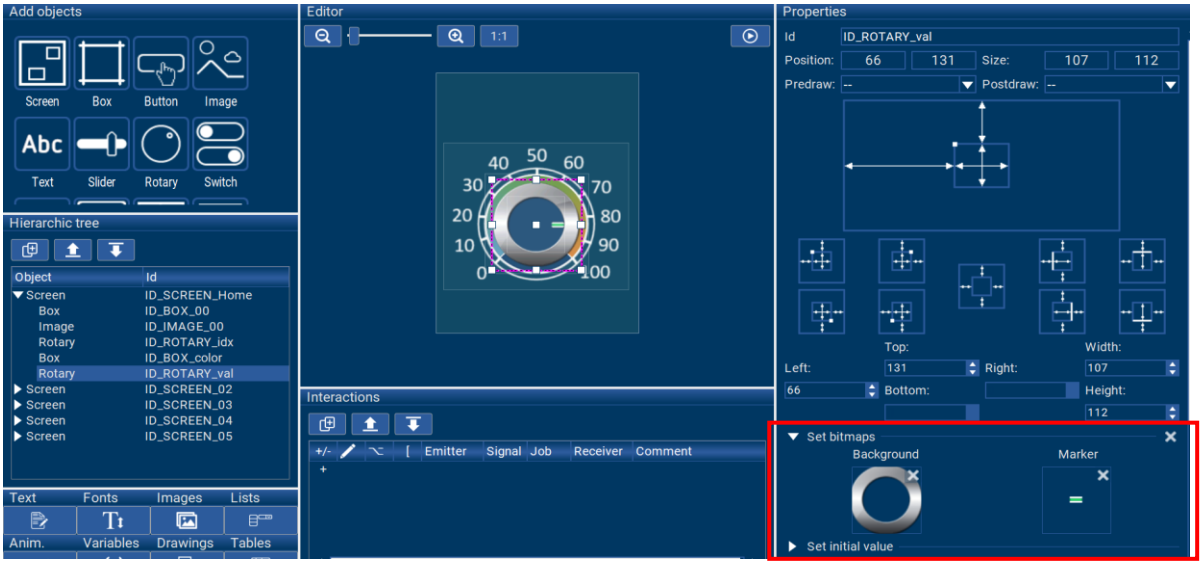


Figure 2-16 Bitmaps of Rotary(ID_ROTARY_val)

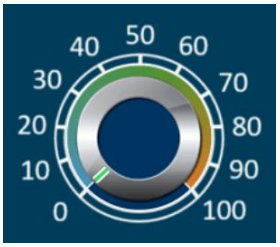


Figure 2-17 Properties Result of Rotary(ID_ROTARY_val)

Properties	Description	Setting values
Initial Value	Starting speed value	0
Span of values	Range of speed value	Min: 0 Max: 999
Range	Range of rotation angle and direction	Positive: 1325 Negative: 1350 (measured in 10th of degrees)
Offset	Starting angle	900 (measured in 10th of degrees)
Radius	Radius of the Rotary.	43
period	Period how long the marker moves when released.	300

Table 2-2 Properties Description of Rotary(ID_ROTARY_val)

5. Double-click on “Text” in “Add objects” to create title and description.

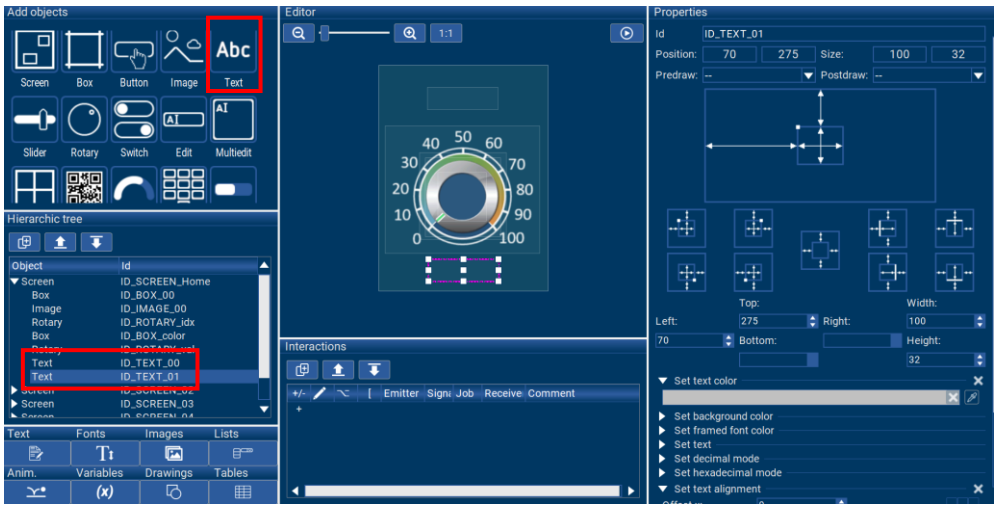


Figure 2-18 Create Title and Description by Text Object

1) According to the steps in Figure 2-19 to Figure 2-20, create a language label for text and set the display contents for ID_TEXT_00.

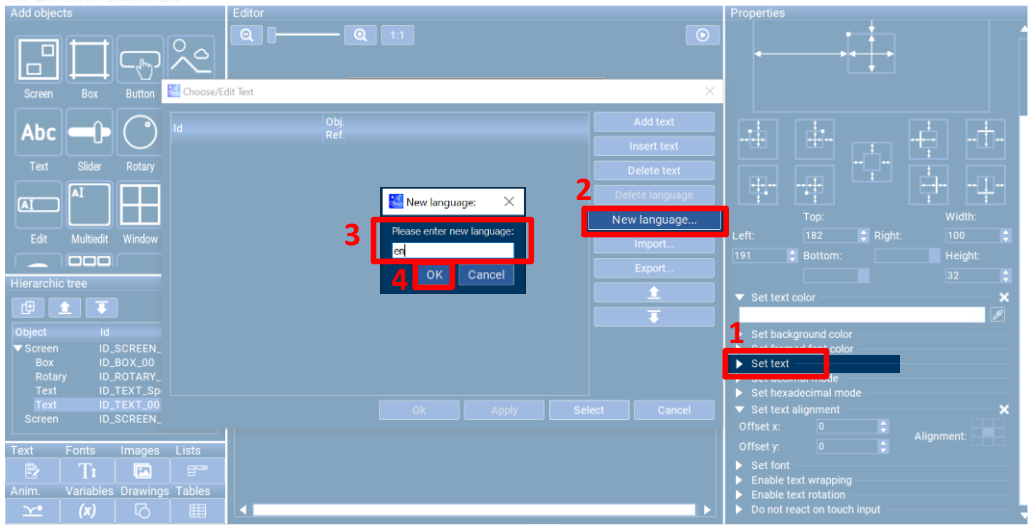


Figure 2-19 Text Object – Properties – Create 1st Text Language (English)

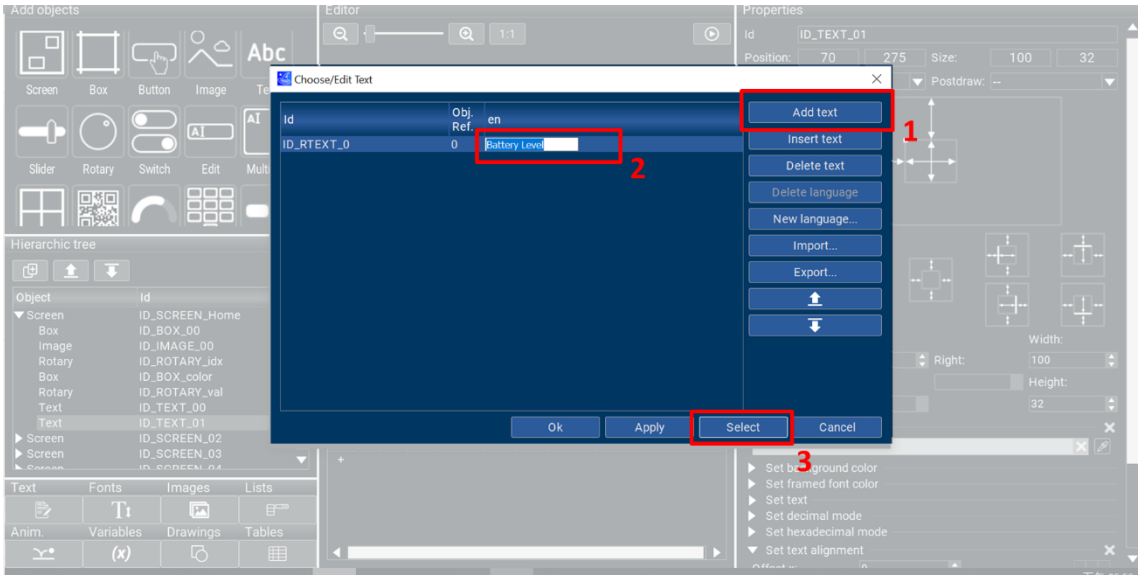


Figure 2-20 Text Object – Properties – Edit Text Contents for Title

2) According to the steps in Figure 2-21 to Figure 2-23, set the title content color, background color, and text alignment.

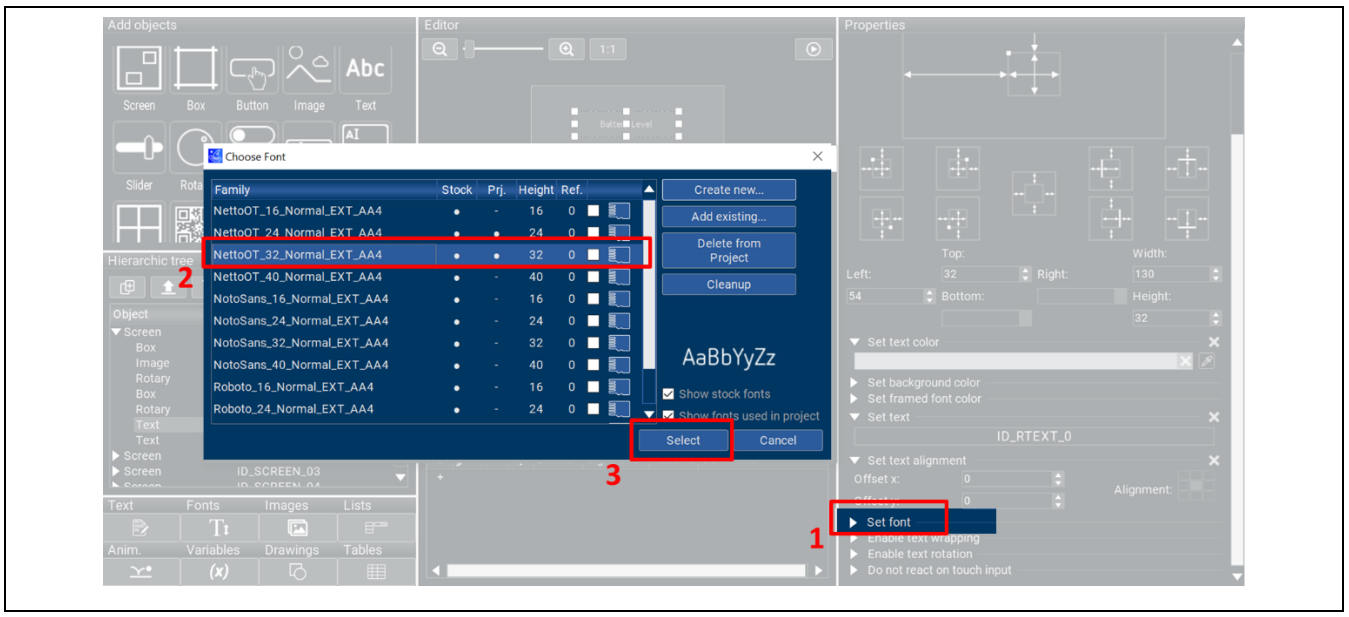


Figure 2-21 Text Object – Properties - Font

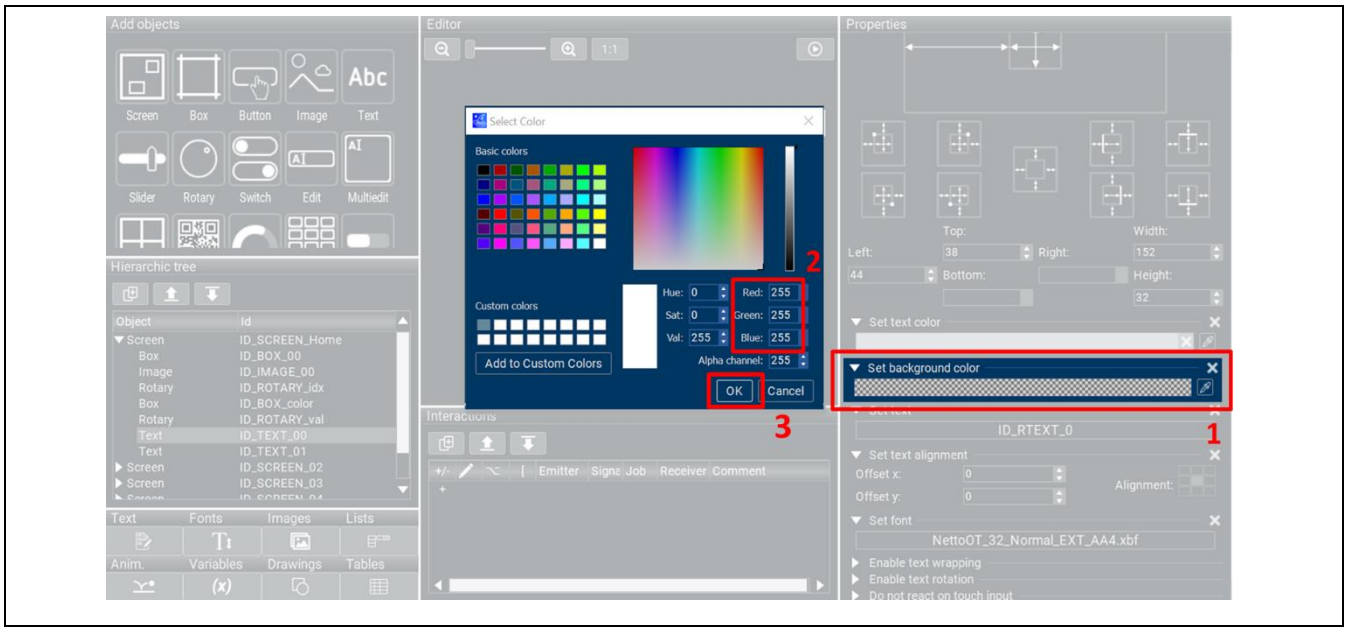


Figure 2-22 Text Object – Properties – Color of Object Background (White)

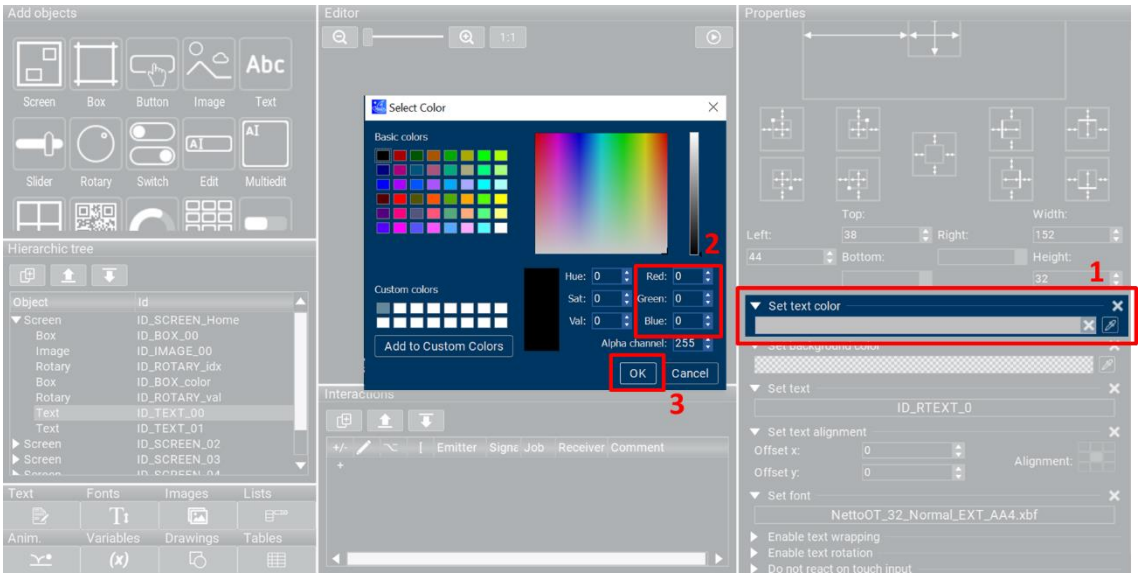


Figure 2-23 Text Object – Properties – Color of Contents (Black)

3) Use the same method to set the content of the description. The settings are shown in Figure 2-24 to Figure 2-25.

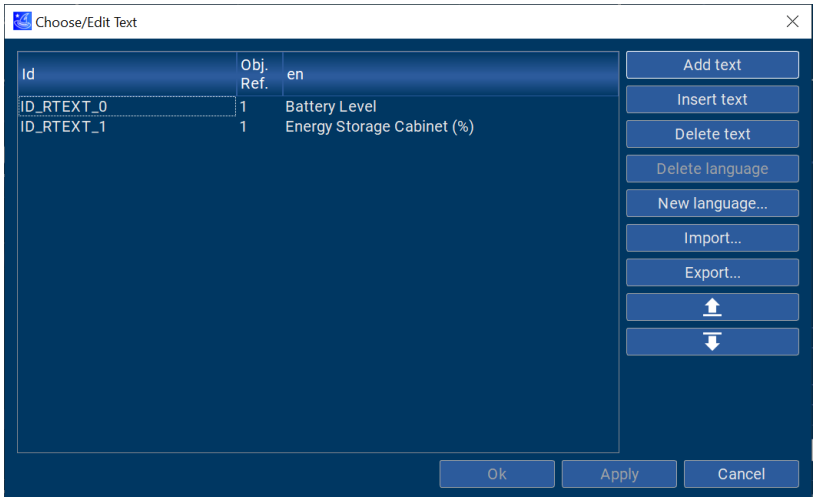


Figure 2-24 Text Object – Properties – Edit Text Contents for Descriptions

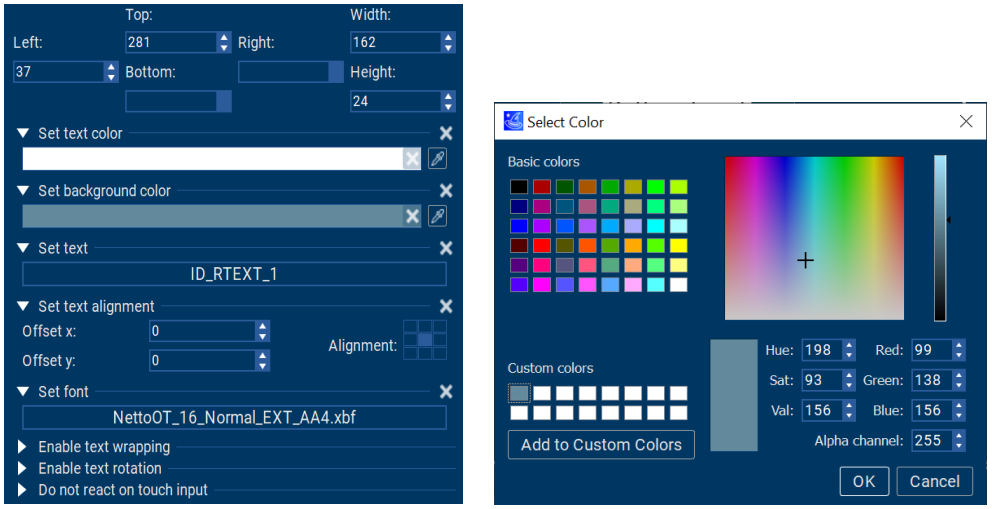


Figure 2-25 Text Object – Properties – Descriptions

6. Click on "Button" in "Add objects" to create a button (ID_BUTTON_next) for the switch screen, and set the properties according to Figure 2-26 and Figure 2-27.

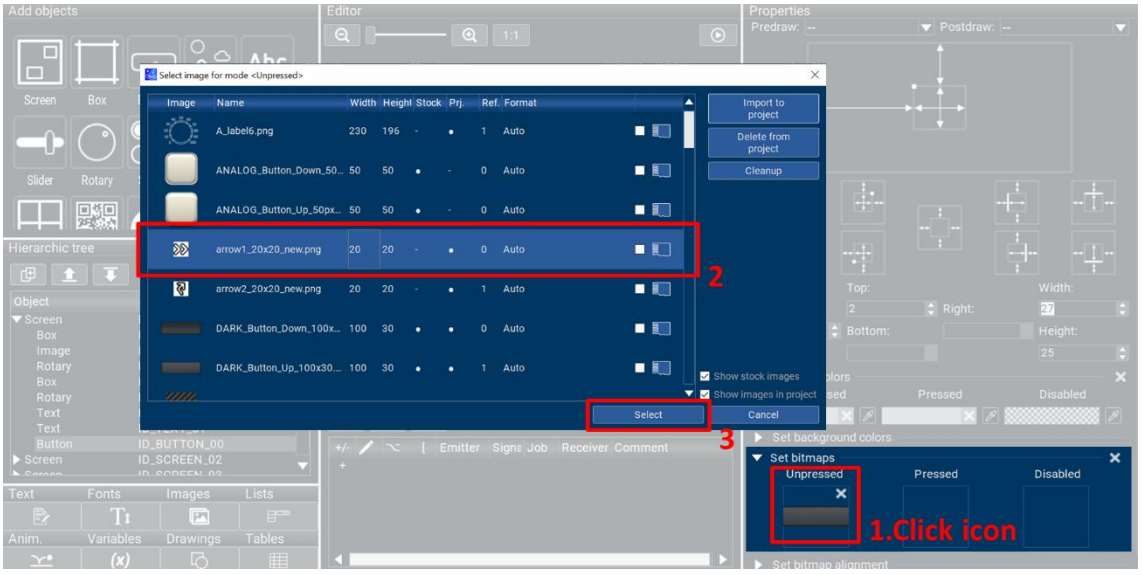


Figure 2-26 Set the Bitmap of Button

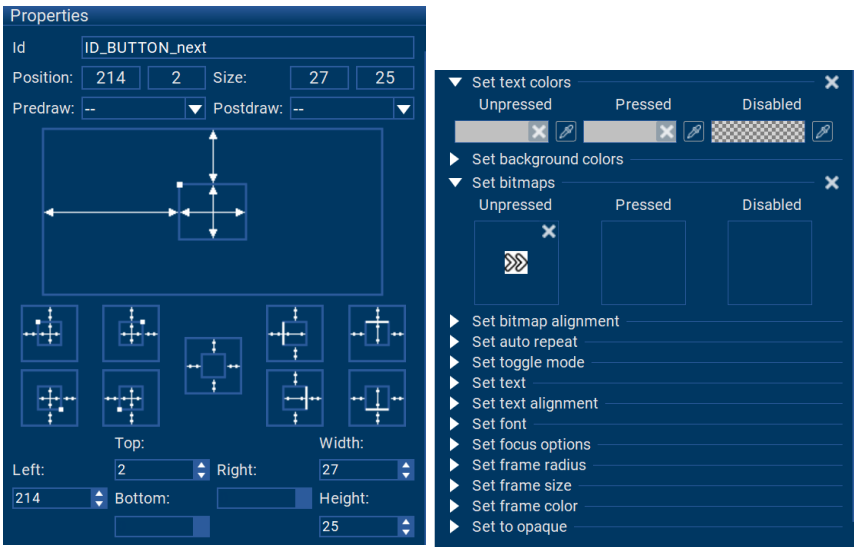


Figure 2-27 Set Properties of ID_BUTTON_next

7. Double-click on "TEXT" in "Add objects" to display the value and unit of Rotary respectively, and set the properties according to Figure 2-28 to Figure 2-30.

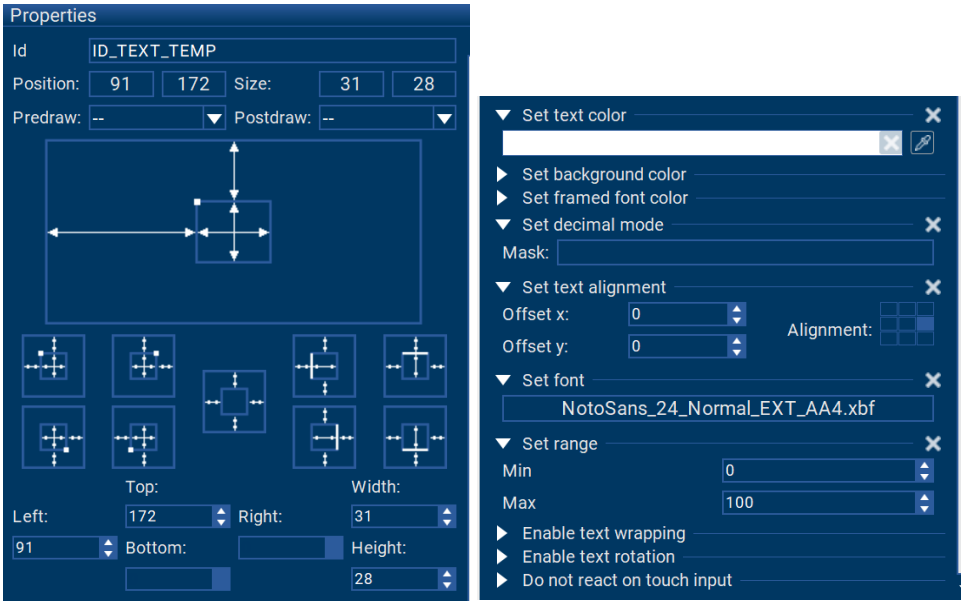


Figure 2-28 Set Properties of ID_TEXT_TEMP

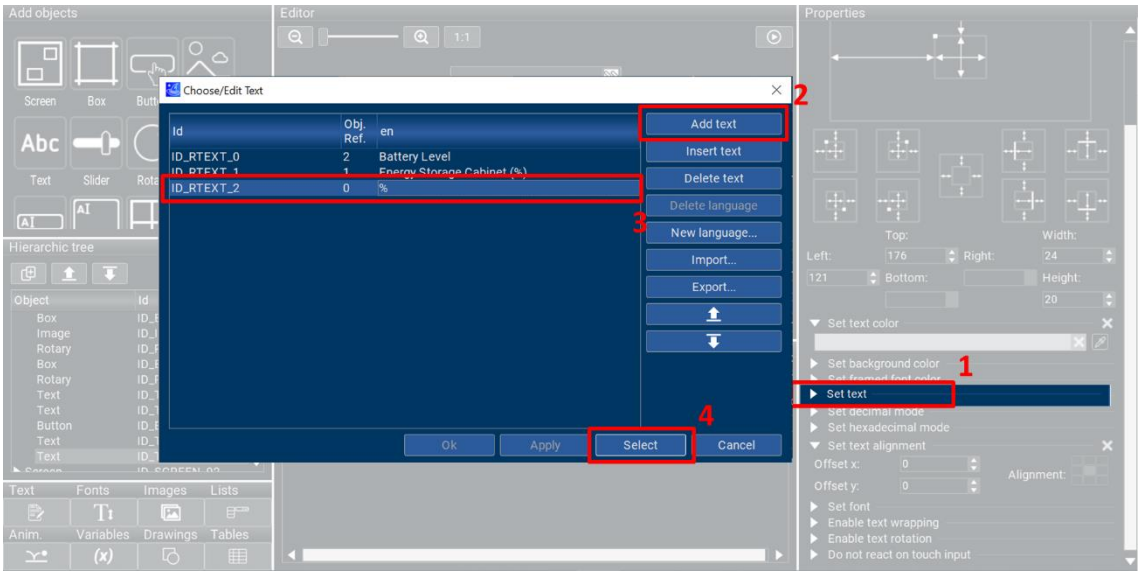


Figure 2-29 Set the Contents of ID_TEXT_SIGN

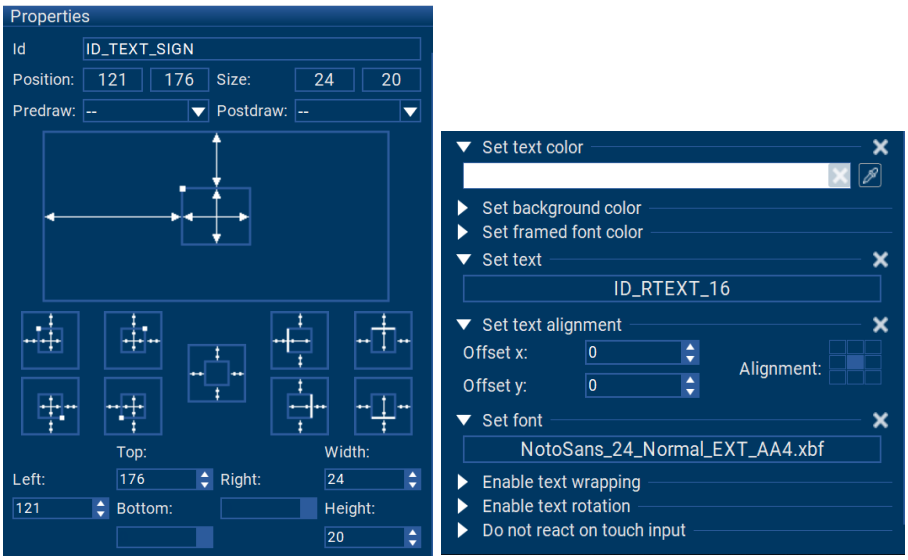


Figure 2-30 Set Properties of ID_TEXT_SIGN

2.3.1.2 Interaction Settings

The completed design is shown in Figure 2-31. Next, users need to establish interactions between objects so that corresponding behaviors will be displayed when operating the objects.

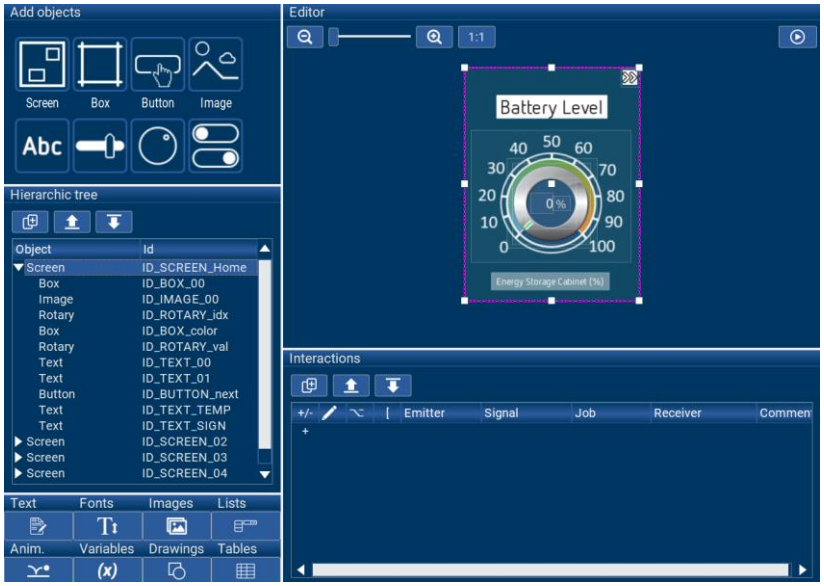


Figure 2-31 Completed Appearance of Home Screen

1. Set the object interactions between Rotary(ID_ROTARY_val) and TEXT(ID_TEXT_TEMP).

   1) According to the steps in Figure 2-32 to Figure 2-35, create a variable by the icon "Variables".

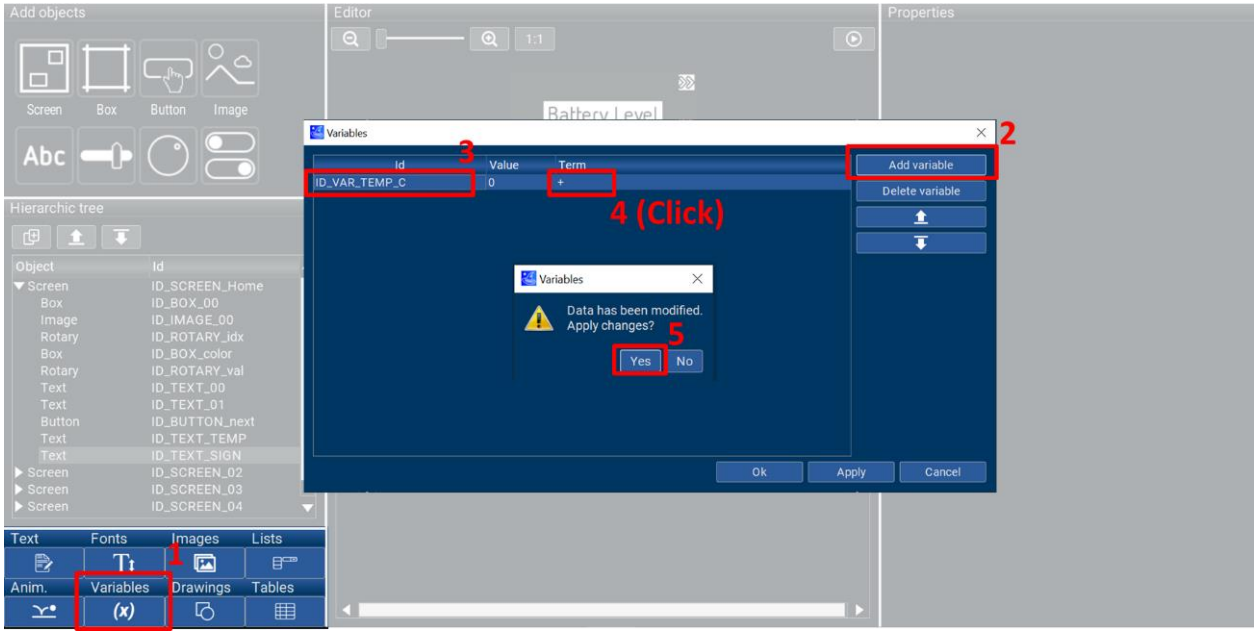


Figure 2-32 Add a Variable (ID_VAR_TEMP_C)

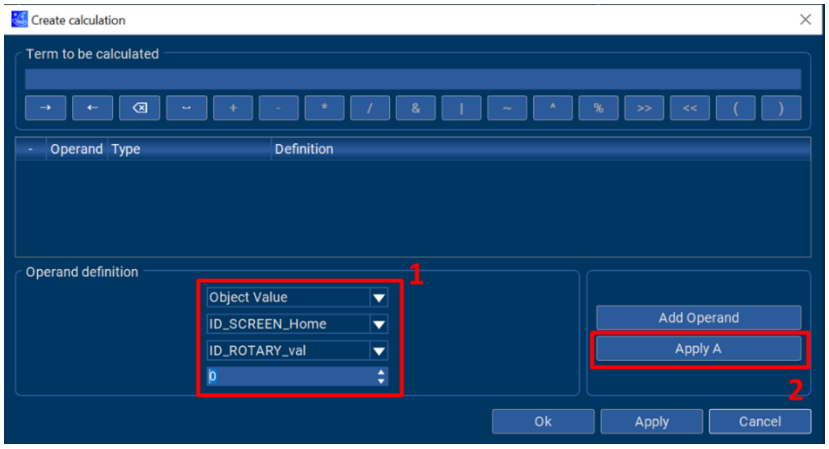


Figure 2-33 Set the Calculated Value of ID_VAR_TEMP_C (1)

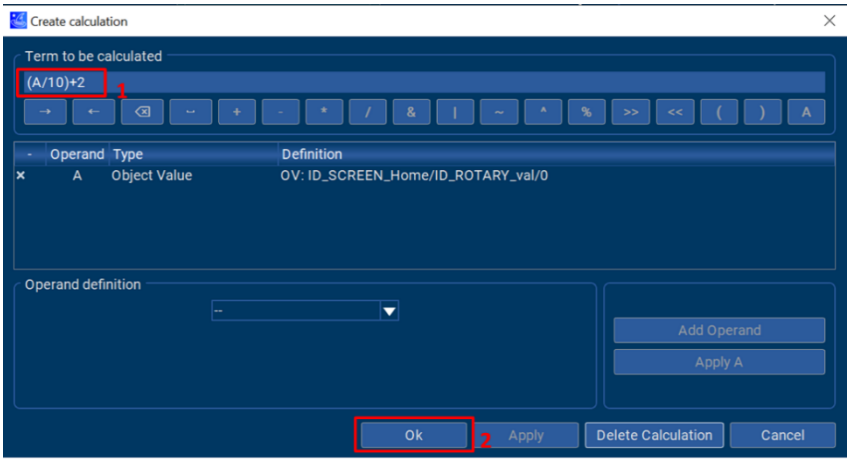


Figure 2-34 Set the Calculated Value of ID_VAR_TEMP_C (2)

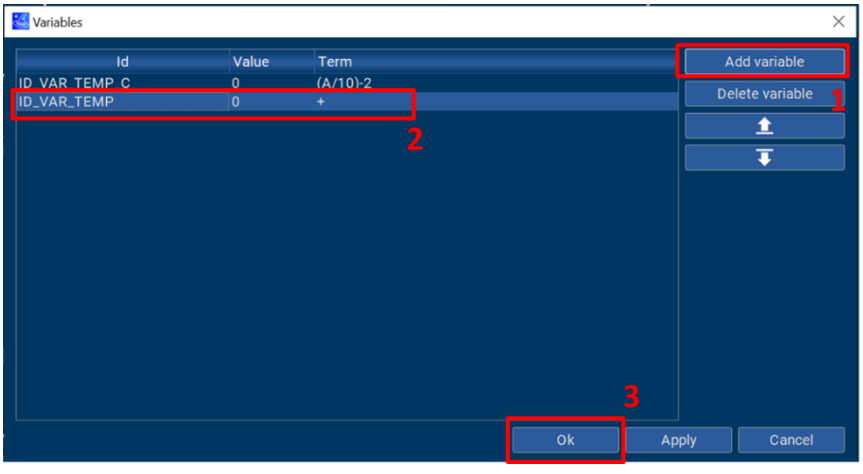


Figure 2-35 Add a Variable (ID_VAR_TEMP)

2) Set up the interaction between Rotary and TEXT in the interactions field, so that turning Rotary simultaneously changes the displayed value of TEXT.

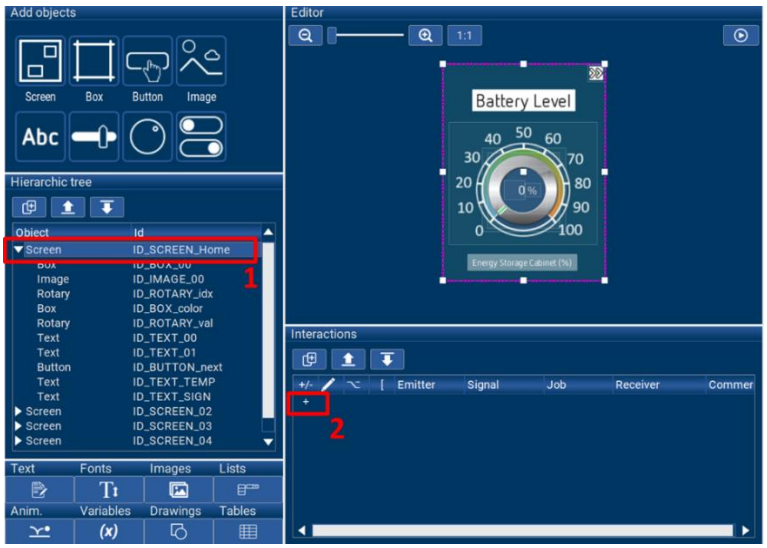


Figure 2-36 Add an Interaction Item

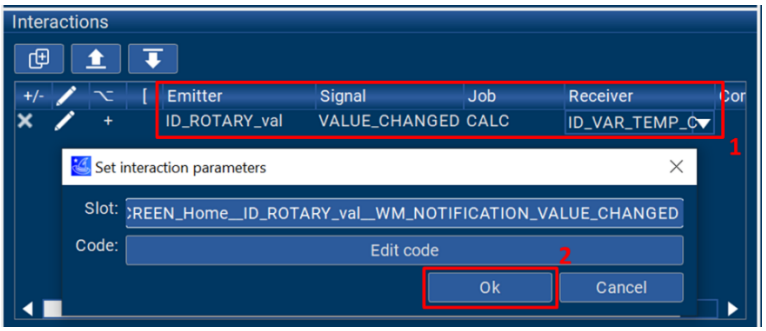


Figure 2-37 Map the Rotary Value to Variable (ID_VAR_TEMP_C)

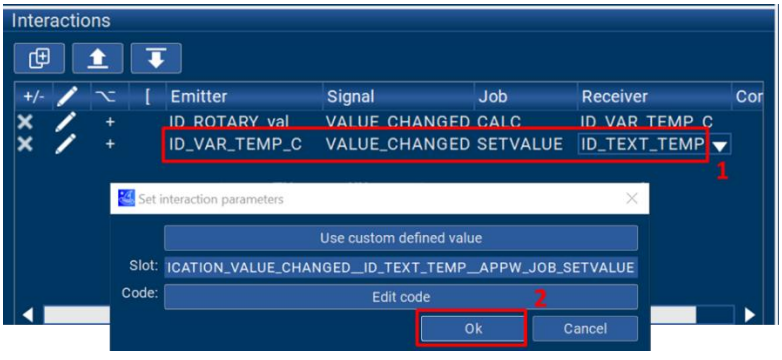


Figure 2-38 Assign the Variable (ID_VAR_TEMP_C) Value to Text Object (ID_TEXT_TEMP)

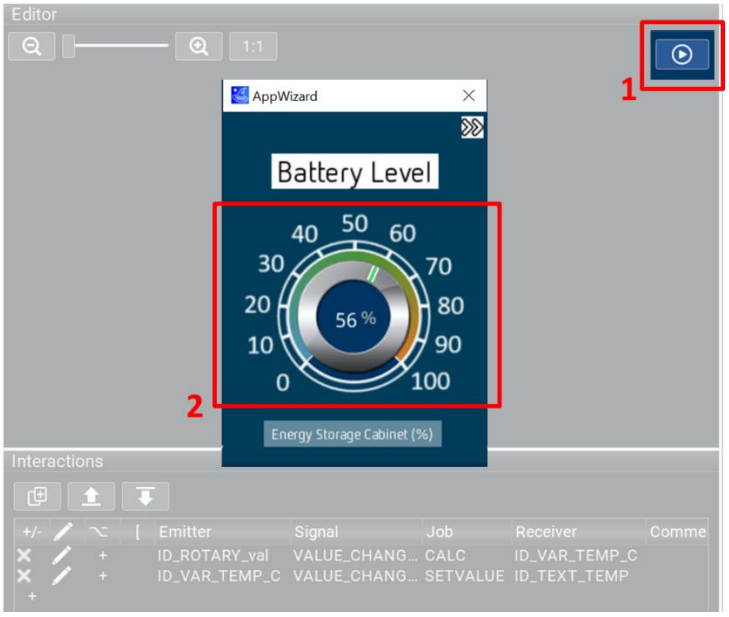


Figure 2-39 Confirm the Interaction Behavior through “Start Play Mode”

2. Assign the color for the values of the Rotary (ID_ROTARY_val) and let the value change affect the display color of the Box (ID_BOX_color).

   1) According to the steps in Figure 2-40 to Figure 2-43, click on "Variables" and add the variables of colors (ID_VAR_COLOR_BLUE, ID_VAR_COLOR_GREEN, ID_VAR_COLOR_ORANGE), the switches of change color (ID_VAR_COLOR_0, ID_VAR_COLOR_1), and the pointer of color (ID_VAR_COLOR_POS).

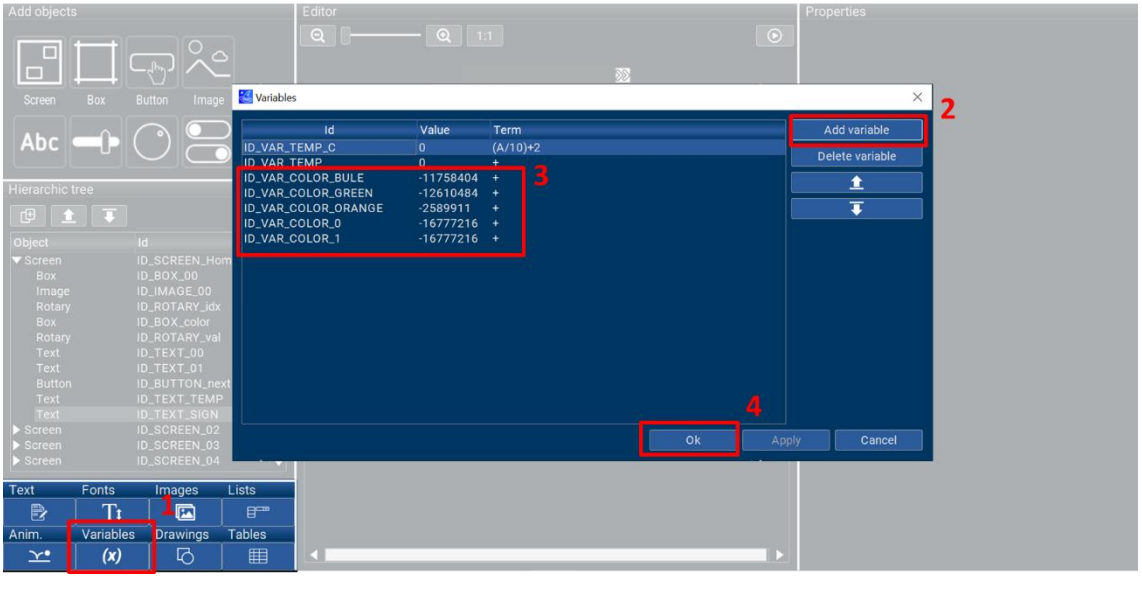


Figure 2-40 Set the Initial Values of Colors Variables

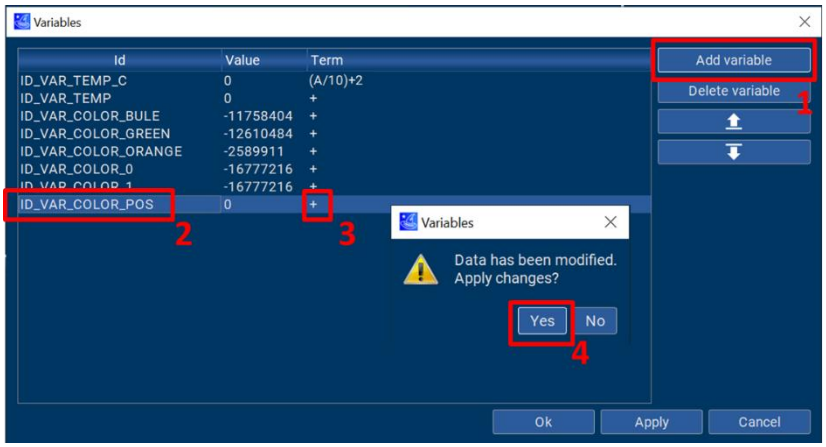


Figure 2-41 Add Variable for the Color Pointer

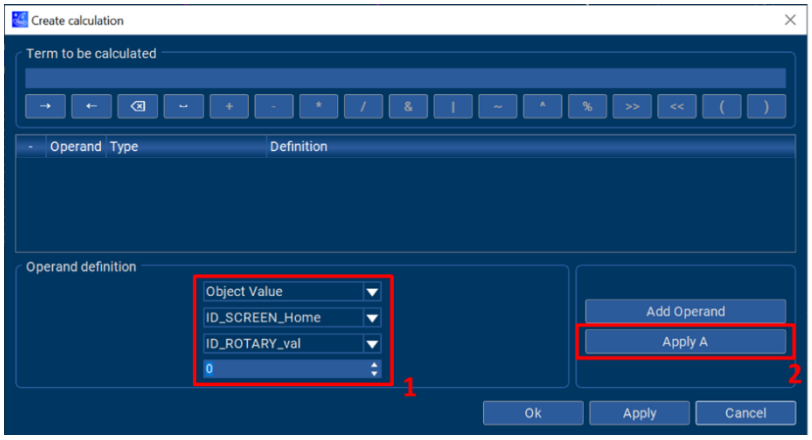


Figure 2-42 Set Variable of the Color Pointers (1)

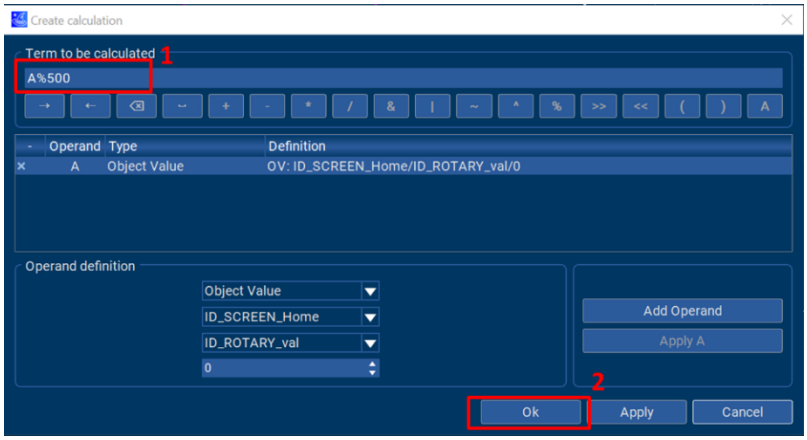


Figure 2-43 Set Variable of the Color Pointers (2)

2) Set the interaction between the Rotary (ID_Rotary_val) and color variables, and let the display color change with the value of the Rotary.

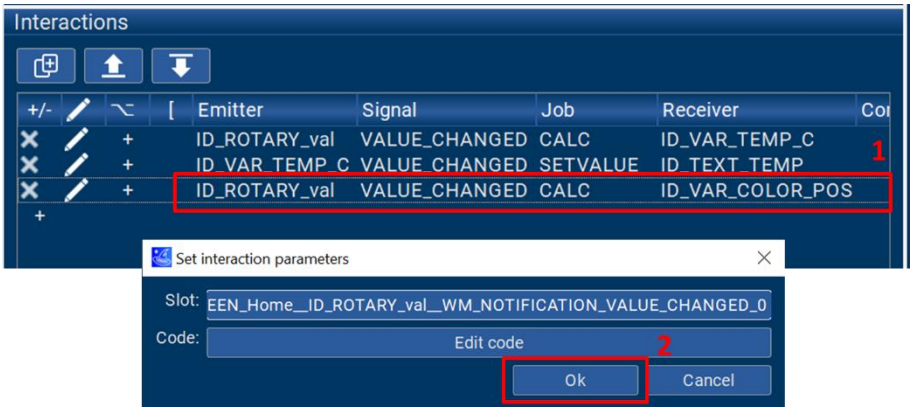


Figure 2-44 Assign the Value of Rotary (ID_Rotary_val) to Variable (ID_VAR_COLOR_POS)

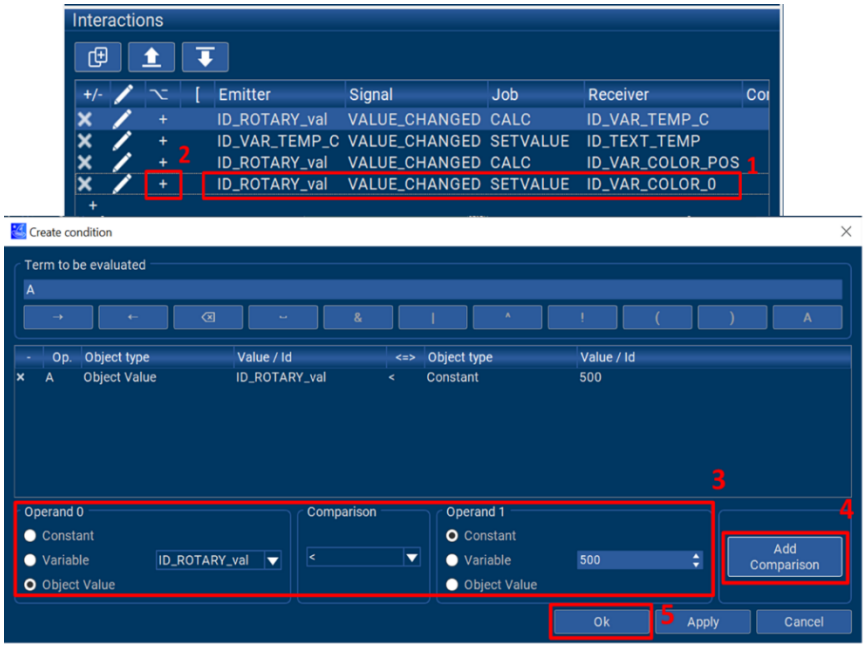


Figure 2-45 Settings of Color Switching (1)

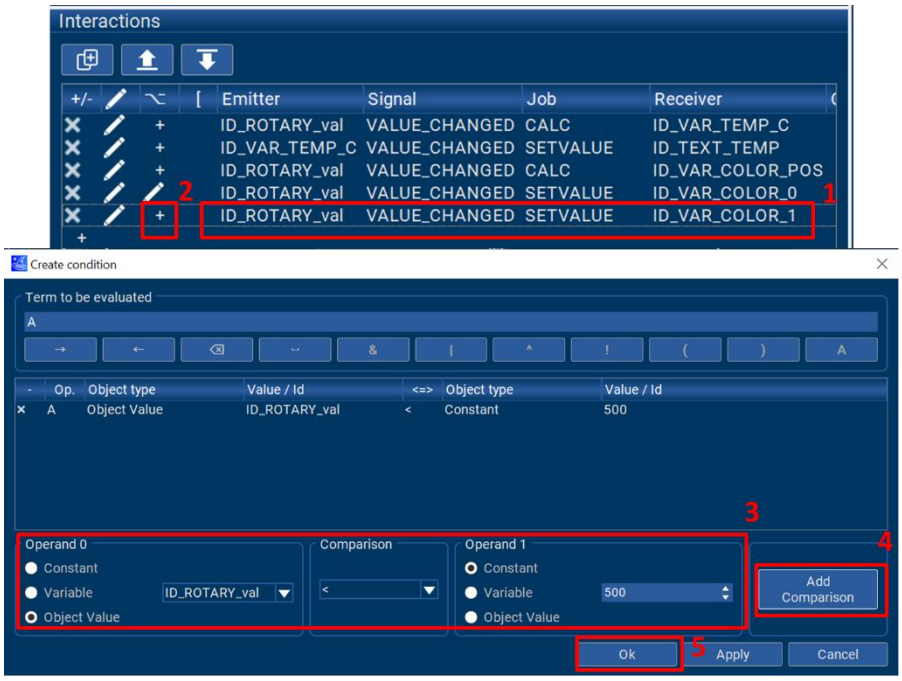


Figure 2-46 Settings of Color Switching (2)

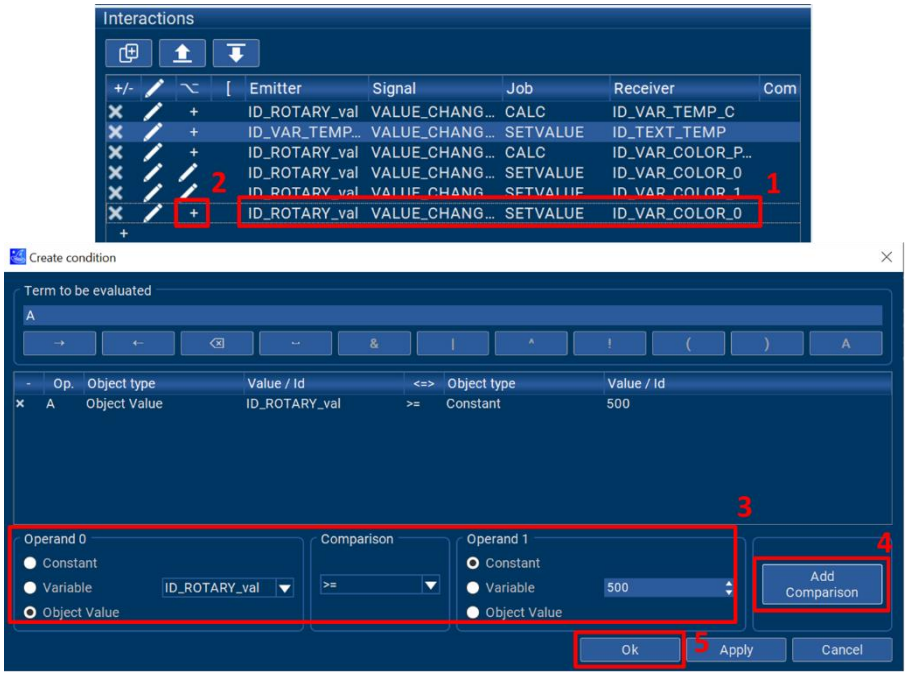


Figure 2-47 Settings of Color Switching (3)

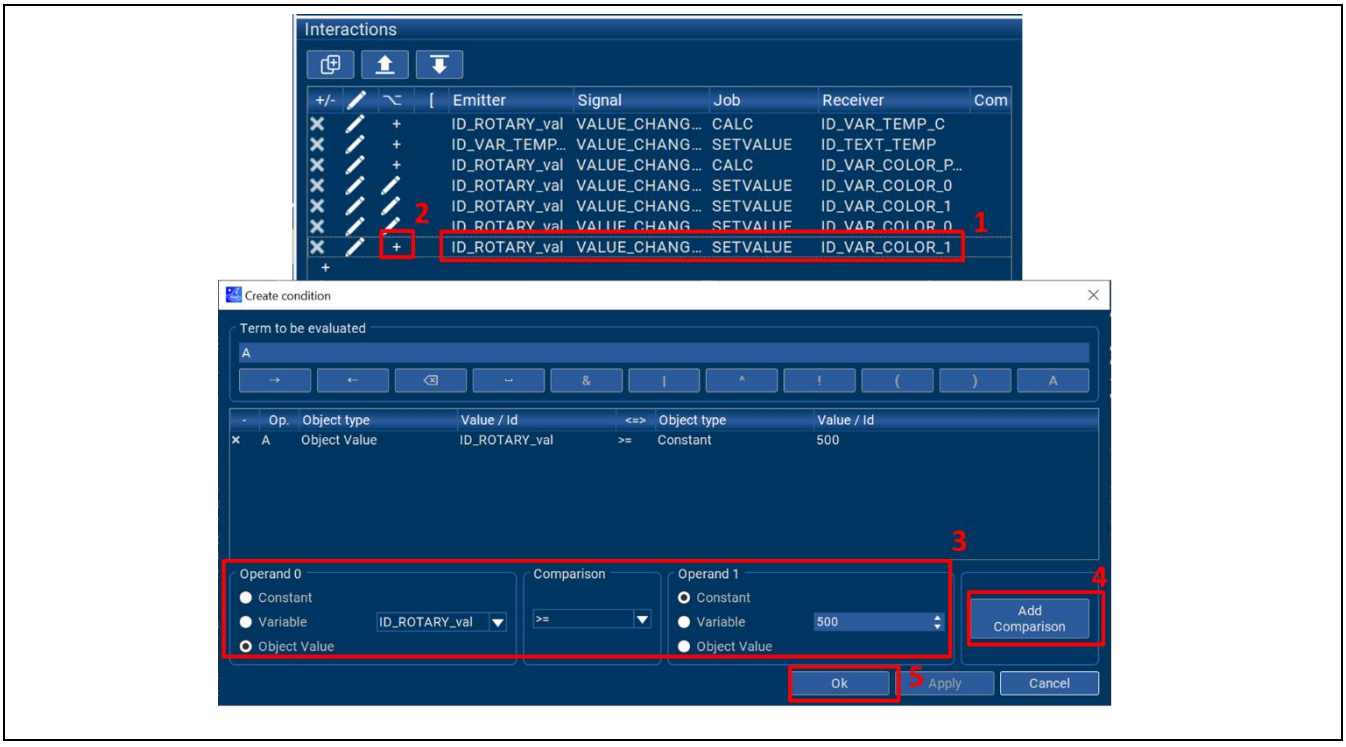


Figure 2-48 Settings of Color Switching (4)

3) According to the steps in Figure 2-49 to Figure 2-52, changing the initial values of color variables.

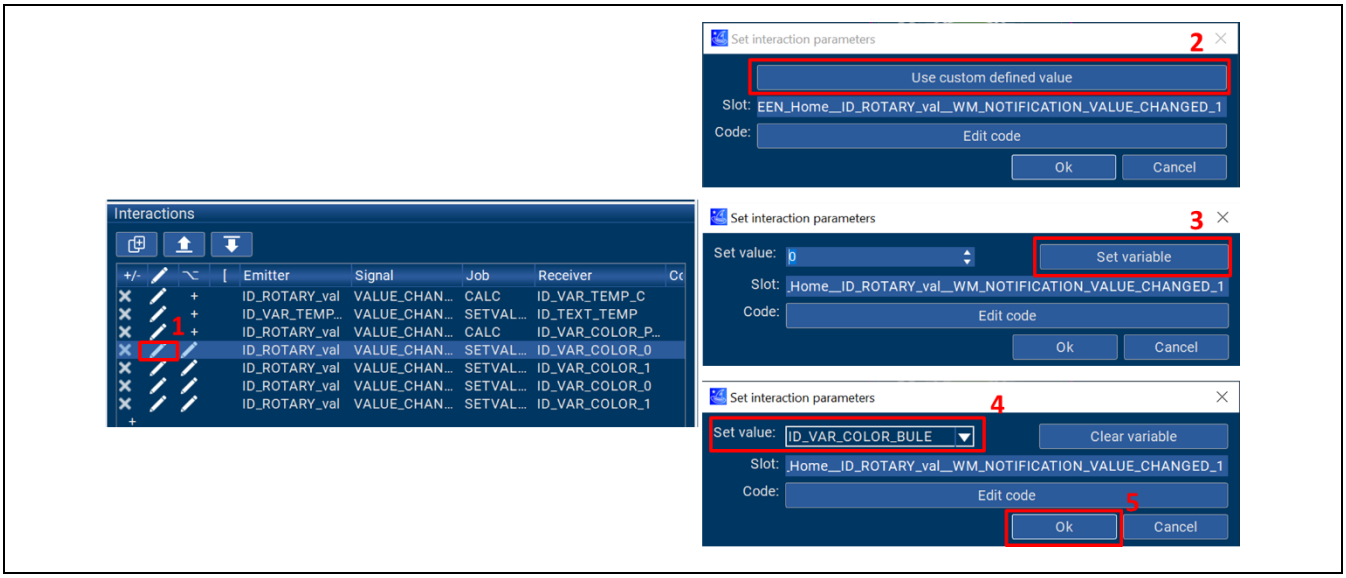


Figure 2-49 Modify the Initial Value of the Color Variables (1)

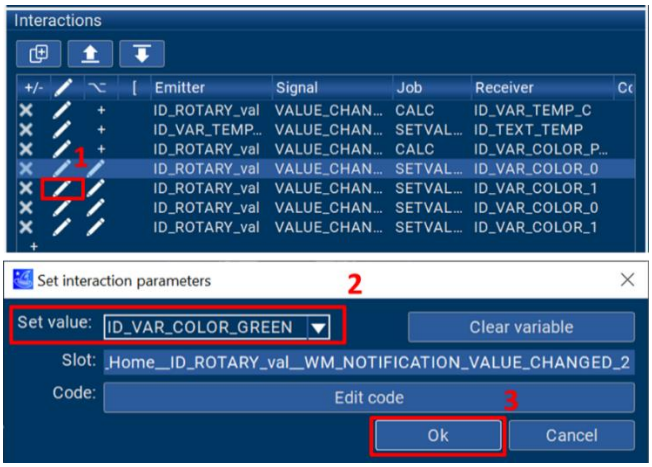


Figure 2-50 Modify the Initial Value of the Color Variables (2)

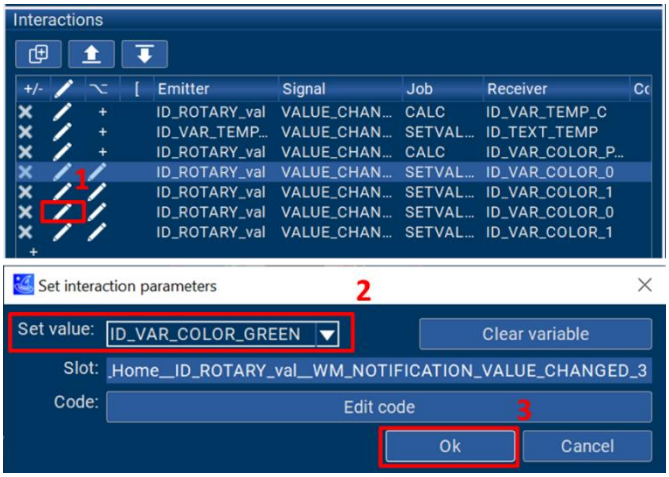


Figure 2-51 Modify the Initial Value of the Color Variables (3)

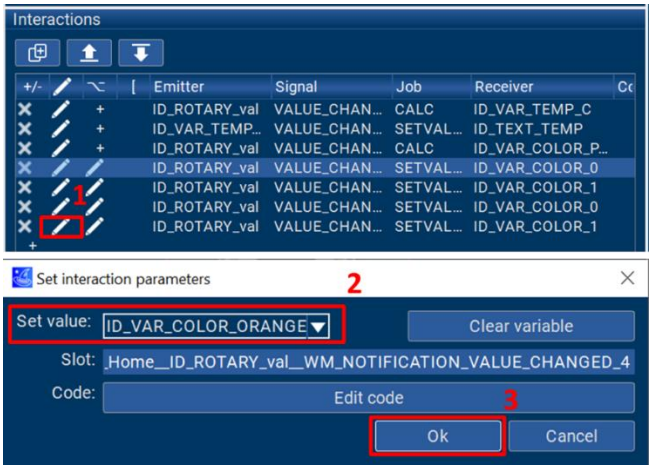


Figure 2-52 Modify the Initial Value of the Color Variables (4)

4) According to the steps in Figure 2-53 to Figure 2-58, click on "Drawings" to add a function to display color according to the numerical changed.

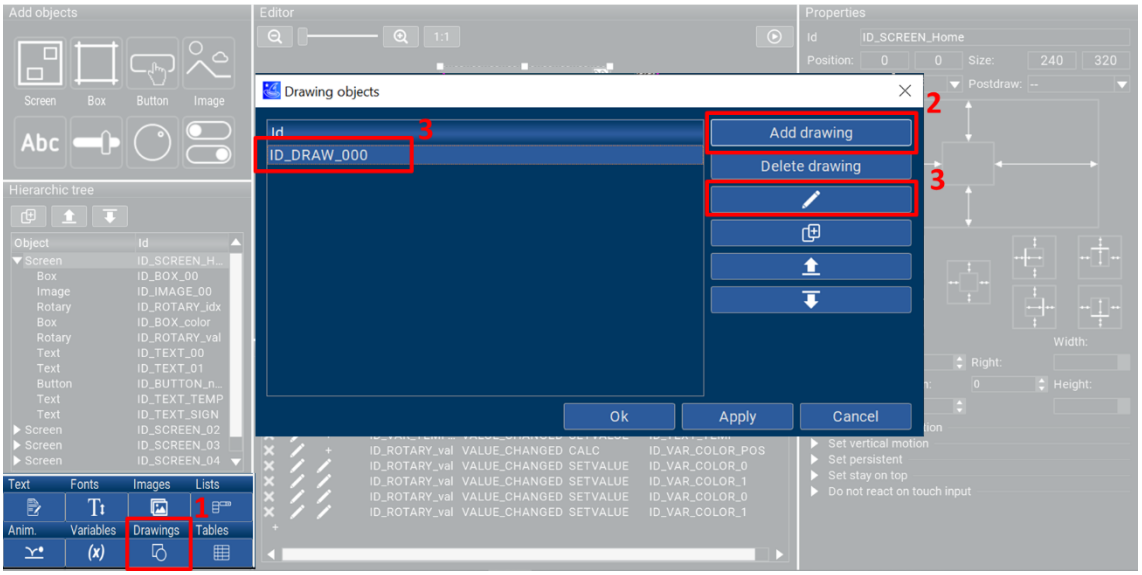


Figure 2-53 Add A Drawings Function (ID_DRAW_000)

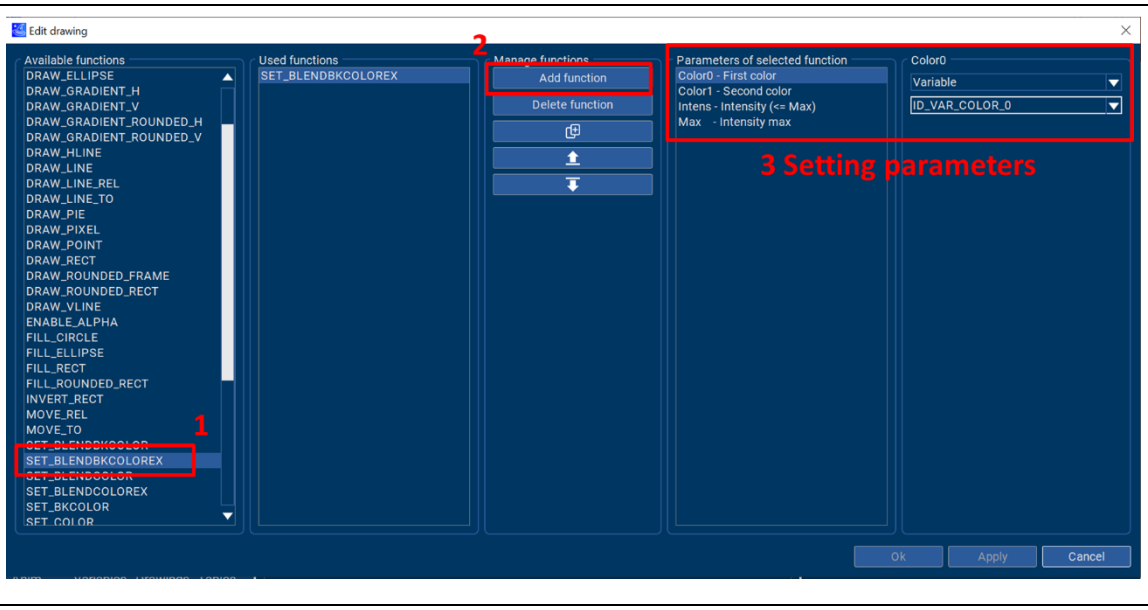


Figure 2-54 Added Coloring Function in ID_DRAW_000

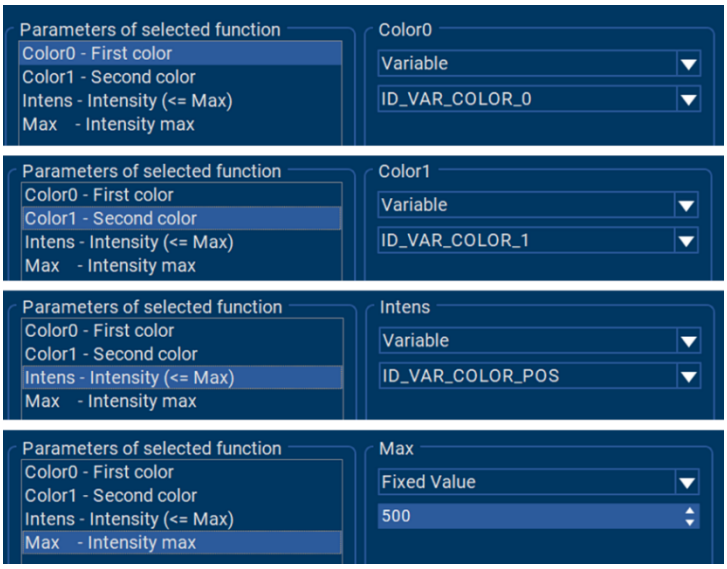


Figure 2-55 Properties Settings of the Coloring Function

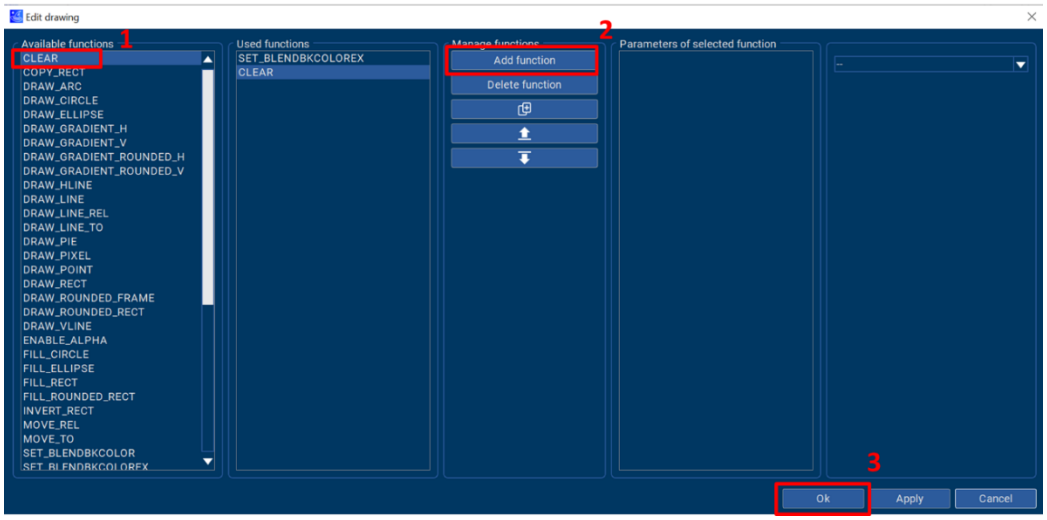


Figure 2-56 Added Clear Function in ID_DRAW_000

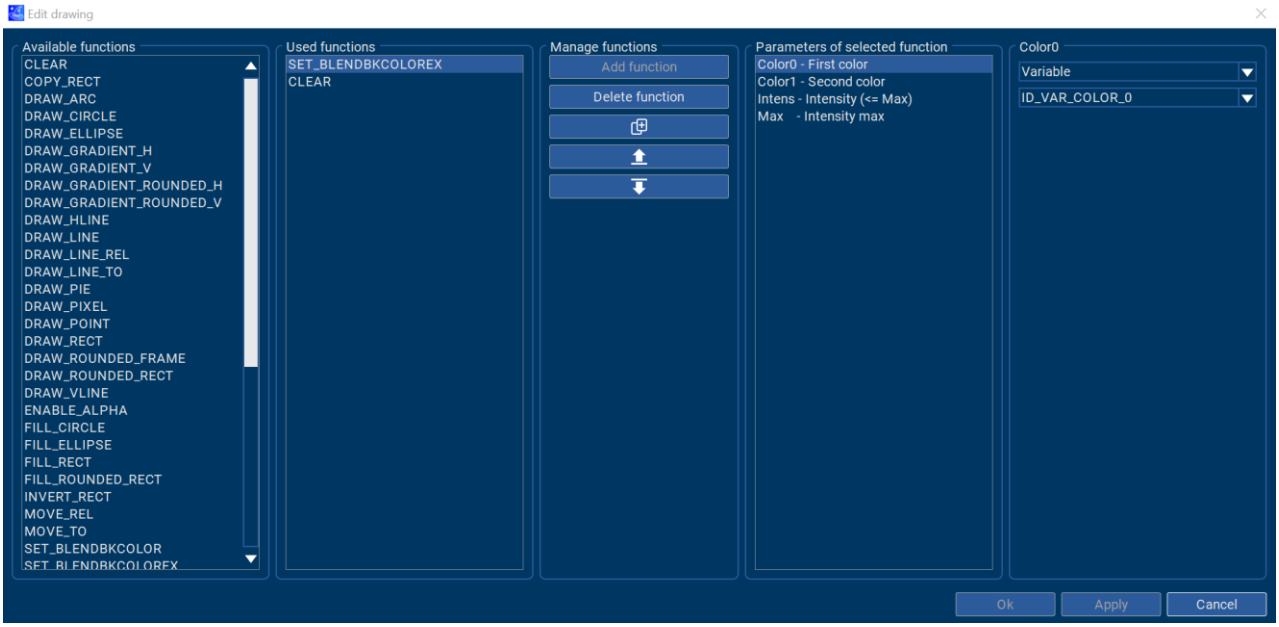


Figure 2-57 Function Overview of ID_DRAW_000

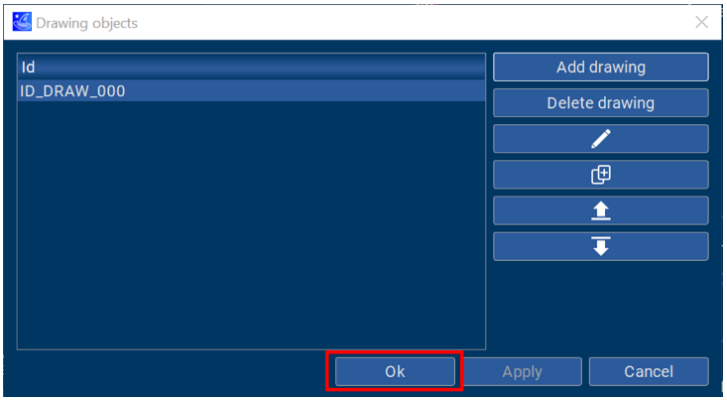


Figure 2-58 Save the Settings of ID_DRAW_000

5) Click on the object "ID_BOX_color" and assign "ID_DRAW_000" to the Postdraw field.

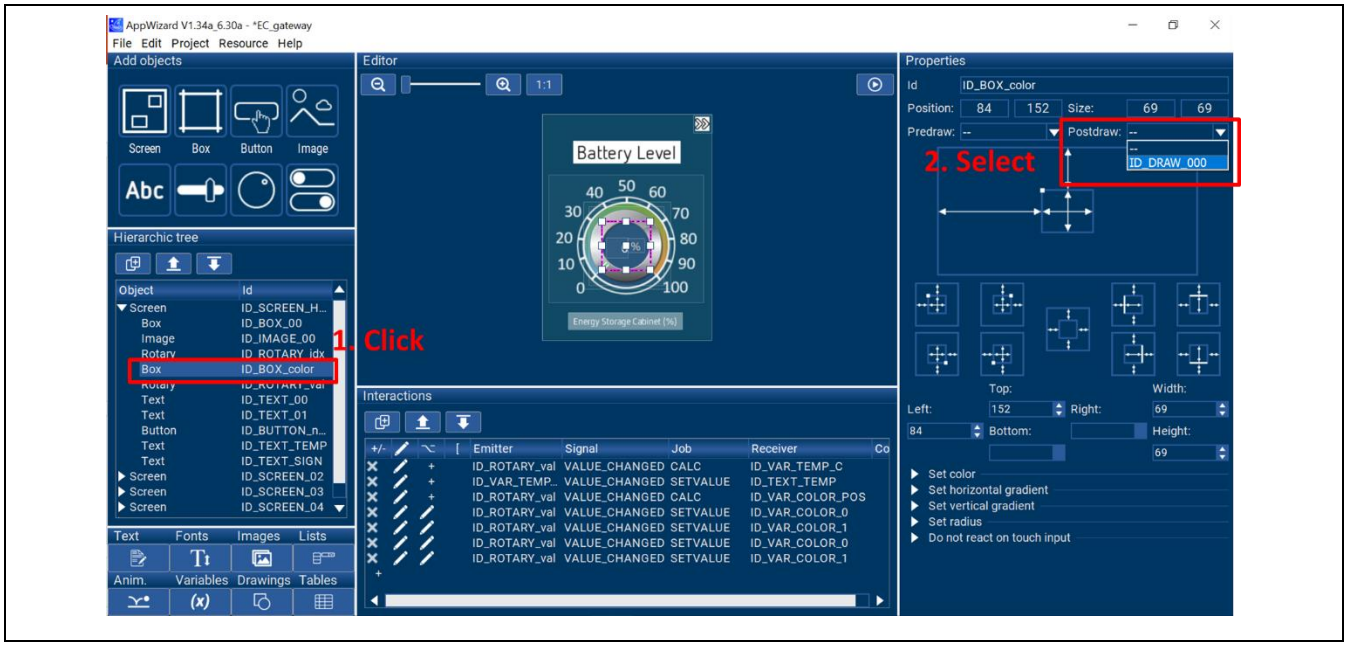


Figure 2-59 Set the Postdraw of ID_DRAW_000

6) After the above actions, the ID_BOX_color color will be changed with the value of ID_Rotary_val.

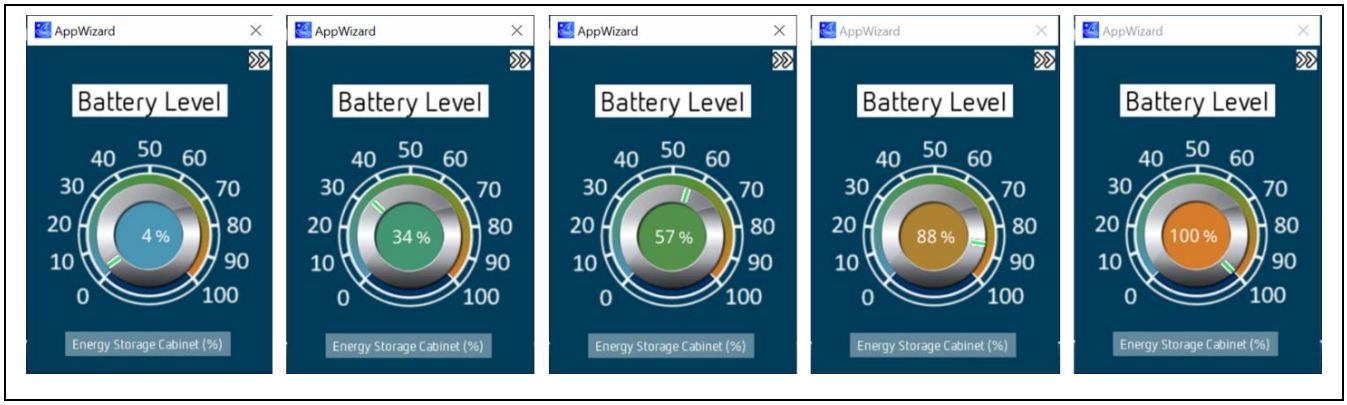


Figure 2-60 Preview ID_BOX_color changes through Play Mode

3. Add the screen switching interaction, make users can switch the screen to ID_SCREEN_02 by the Button (ID_Button_next).

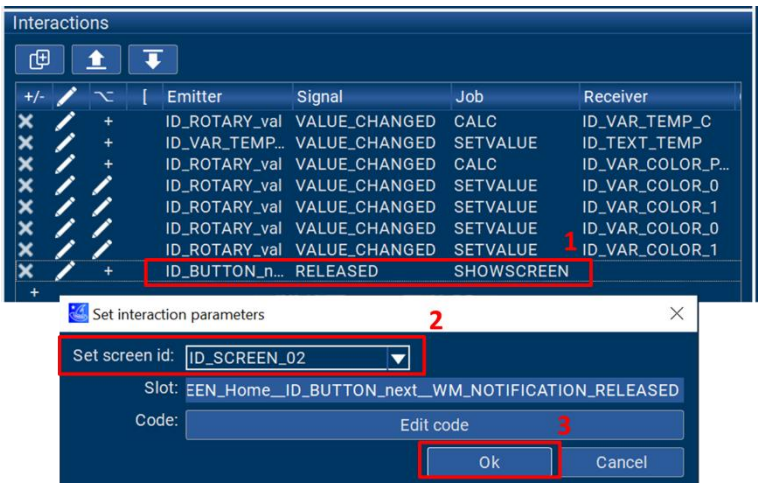
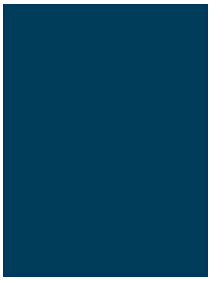
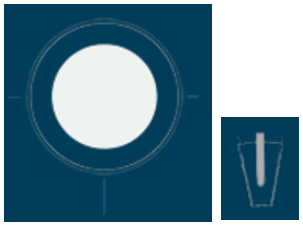


Figure 2-61 Button Settings of Screen Switching

2.3.2 Design of Photovoltaic Inverter (Screen 2)

A photovoltaic inverter (PV Inverter) is an inverter that can convert the DC voltage generated by solar PV panels into mains frequency alternating current (AC) and can be used in commercial power transmission systems or off-grid power grids. This design can also be applied as a user interface for other types of inverters. The widgets used in this screen are shown in Table 2-3.

New Energy Gateway Description	Application GUI	Widget Type
Background (240x320)		Box
Value rotary knob		Rotary
Value background of gauge		Gauge

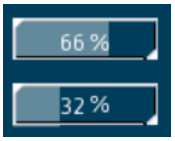
Value display		Gauge
Switch of Progbar		Switch
Display of bearing conversion rate		Progbar
Display value of Progbar		Slider
Text of title		Text
Display value of Rotary		
Button of previous page		Button
Button of next page		

Table 2-3 Widgets of PV Inverter

2.3.2.1 Widgets Configuration of GUI

1. Choose *ID_SCREEN_02* to start editing screen 2, and click on "Text" in "Add objects" to display the title (ID_TEXT_title).

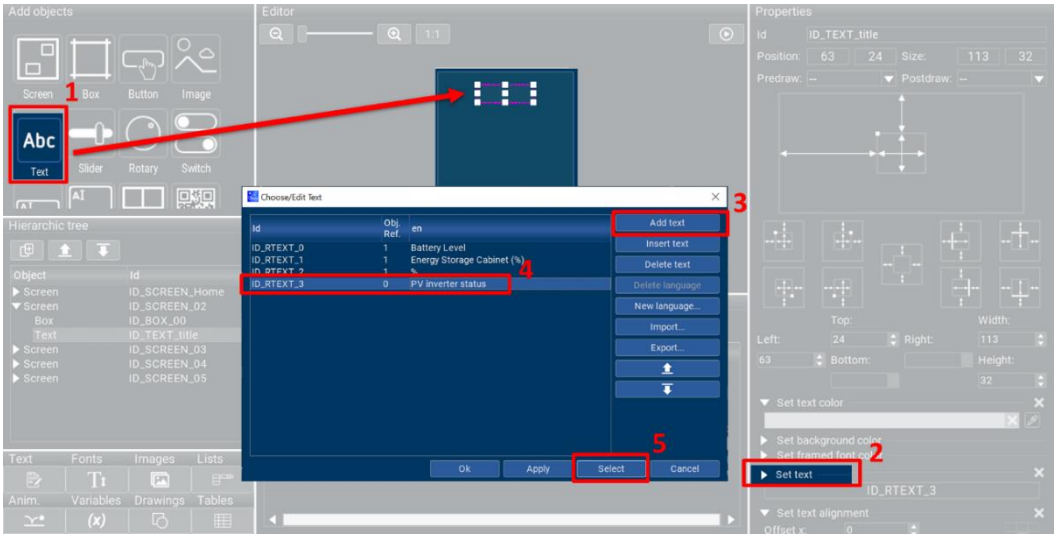


Figure 2-62 Set the Display Contents of Title (ID_TEXT_title)

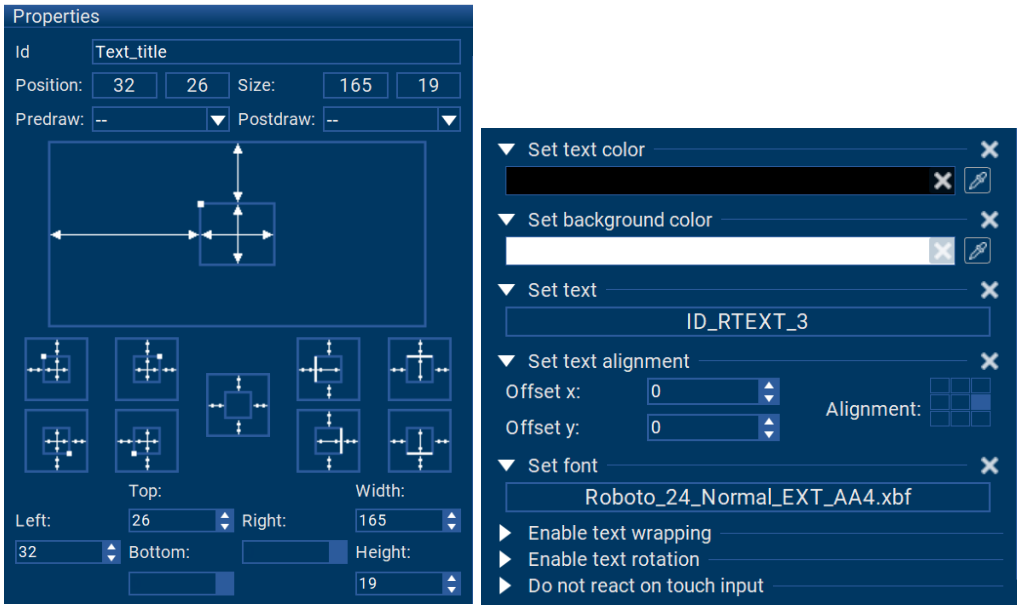


Figure 2-63 Set Properties of ID_TEXT_title

2. Click on "Gauge" in "Add objects" to create a display background of variables changing.

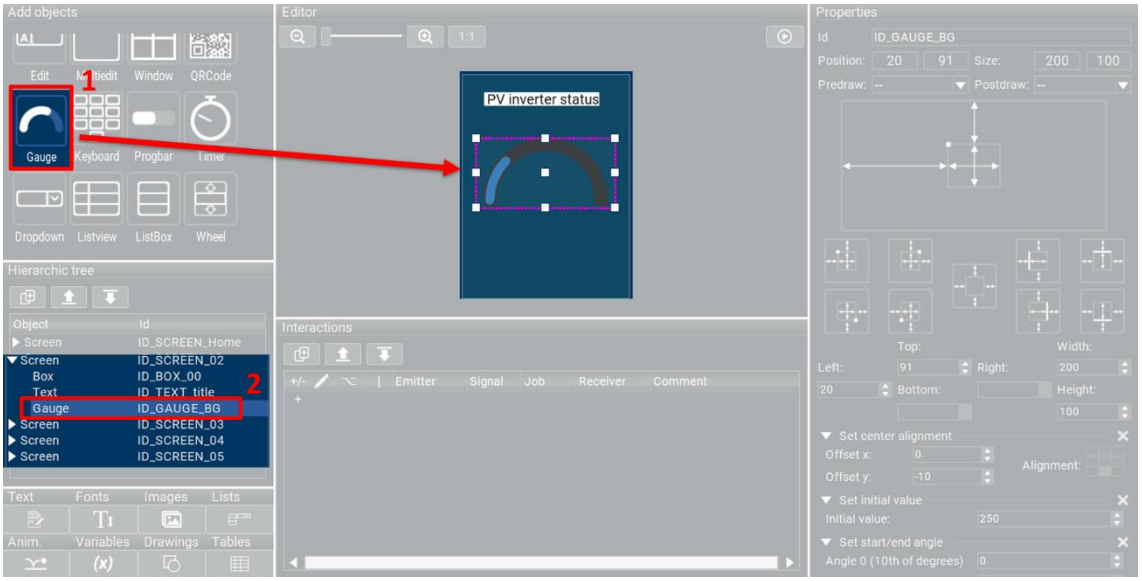


Figure 2-64 Add Gauge Object (ID_Gauge_BG)

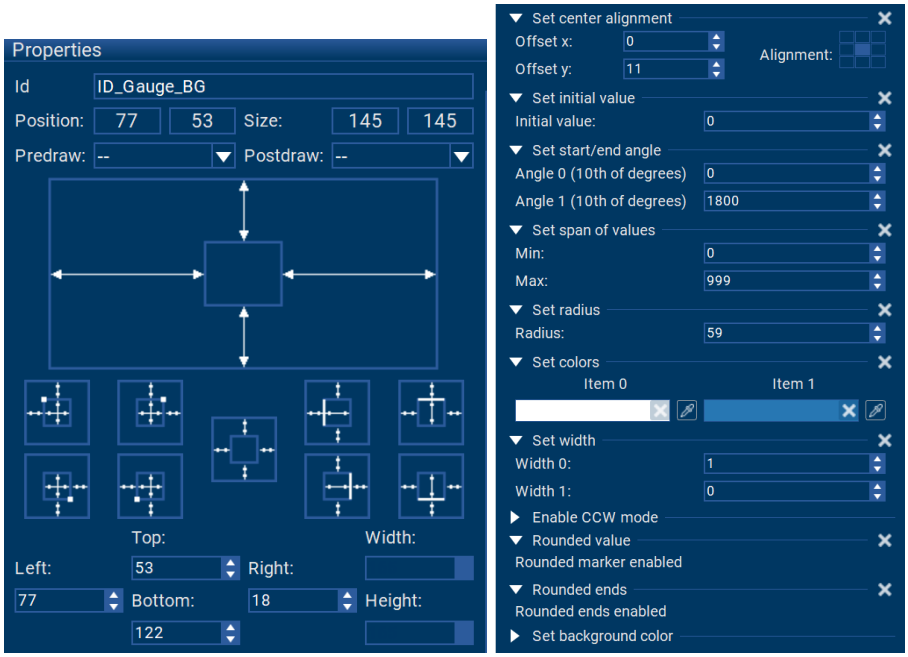


Figure 2-65 Set Properties of ID_Gauge_BG

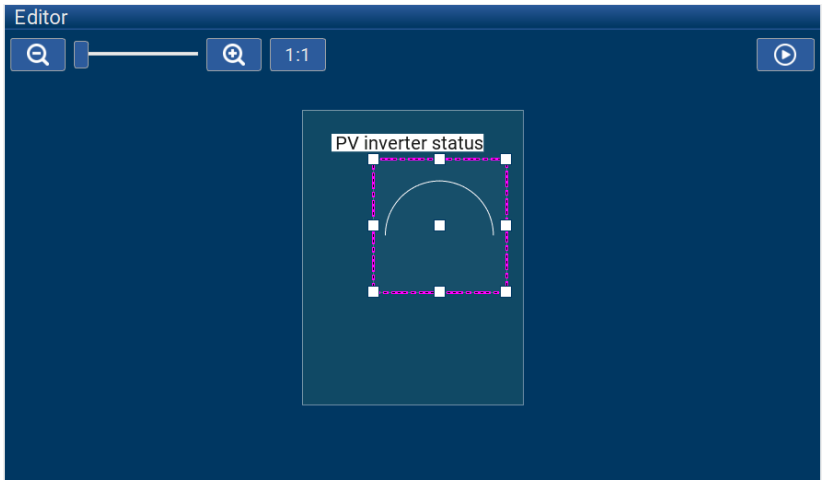


Figure 2-66 Appearance of ID_Gauge_BG

3. Add a second Gauge (ID_Gauge_BAR) object as a value display.

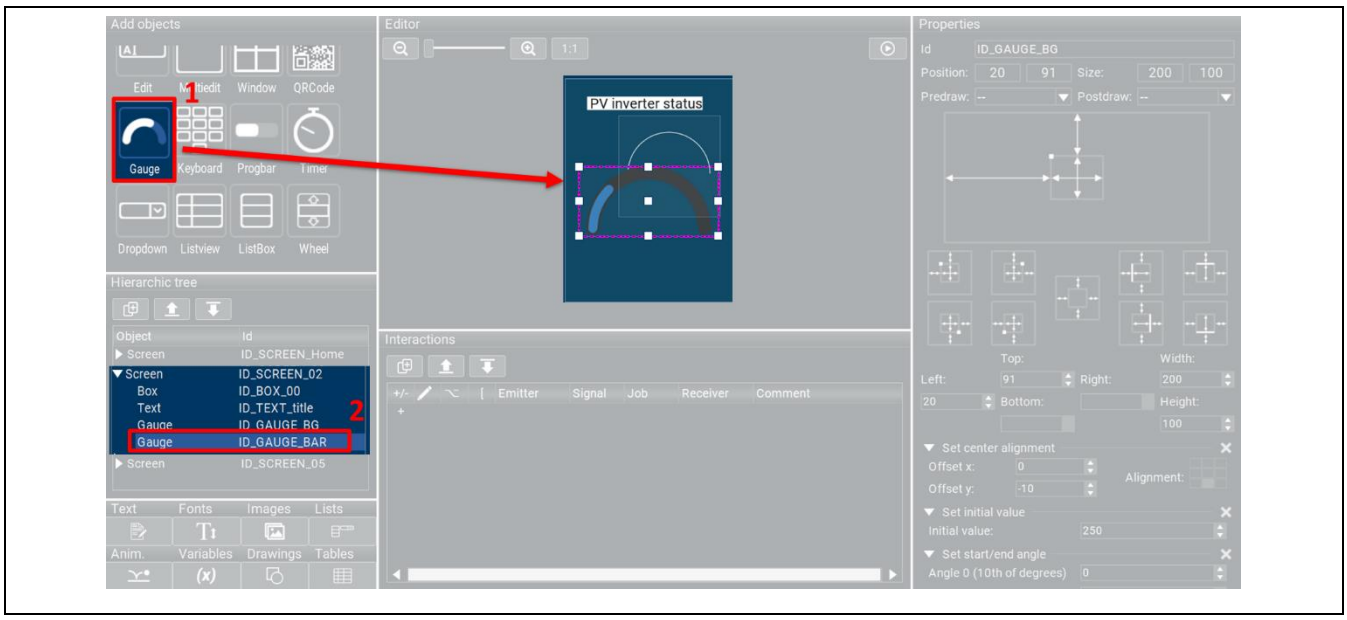


Figure 2-67 Add Gauge Object ID_Gauge_BAR

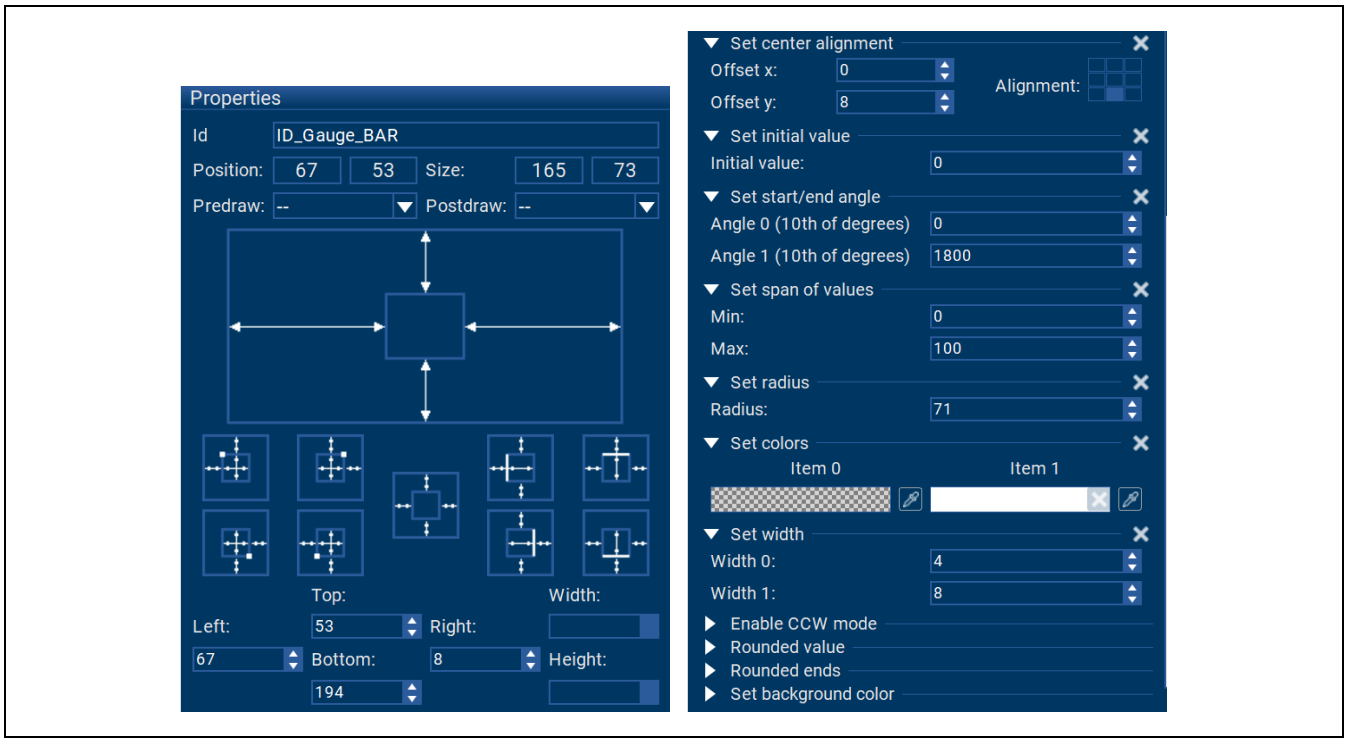


Figure 2-68 Set Properties of ID_Gauge_BAR

4. Click on "Rotary " in "Add objects" to the value changing, and choose images *Rotary_Background.png* and *Rotary_Marker.png* as bitmaps of background and marker.

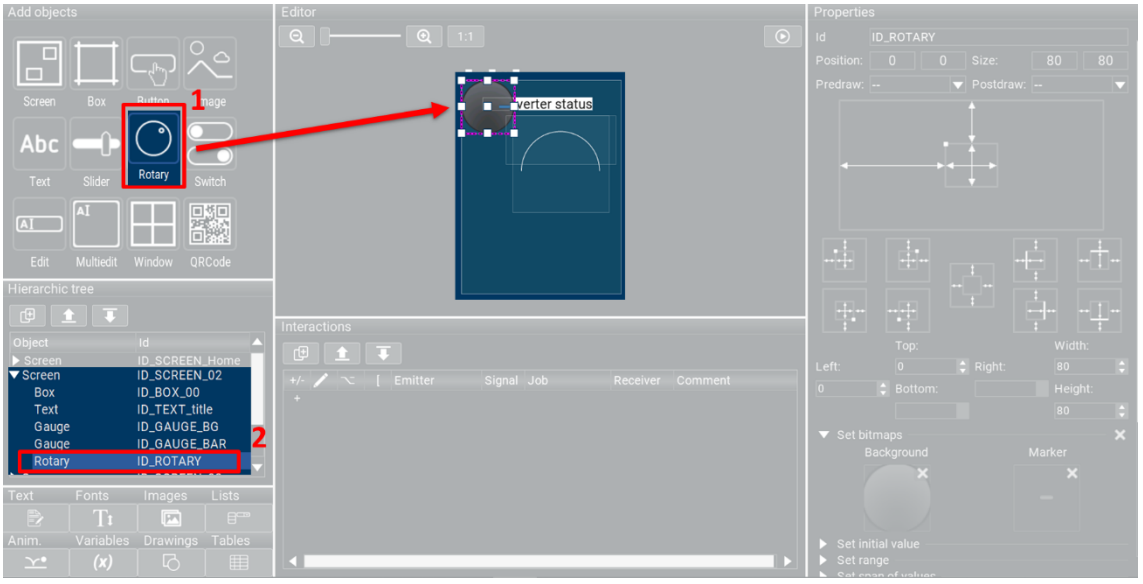


Figure 2-69 Add Rotary Object (ID_Rotary)

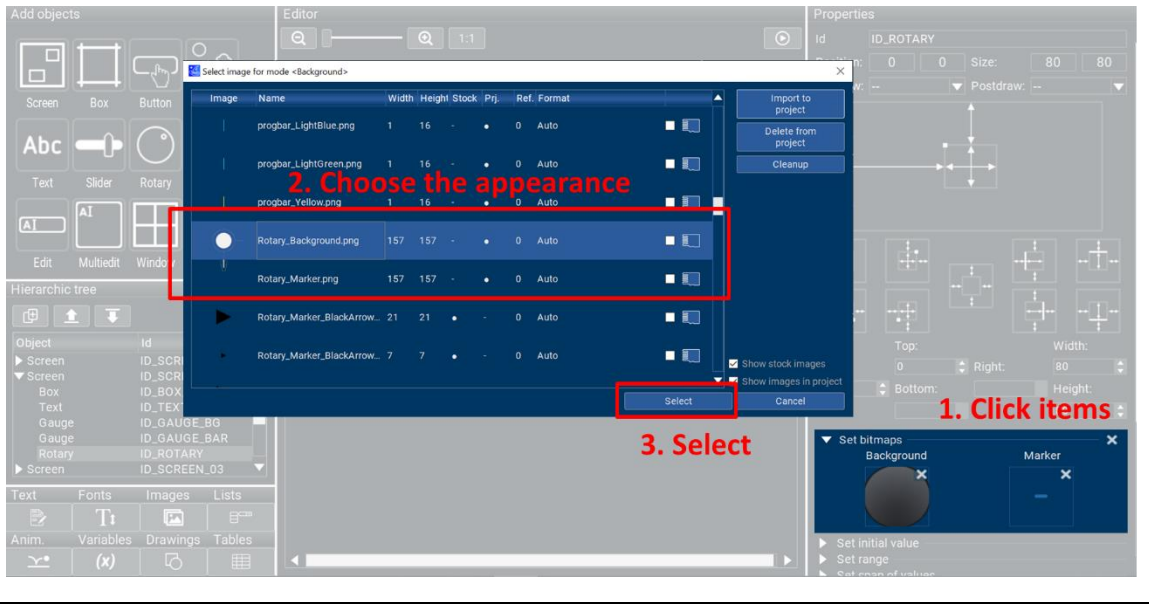


Figure 2-70 Bitmaps Settings of ID_Rotary

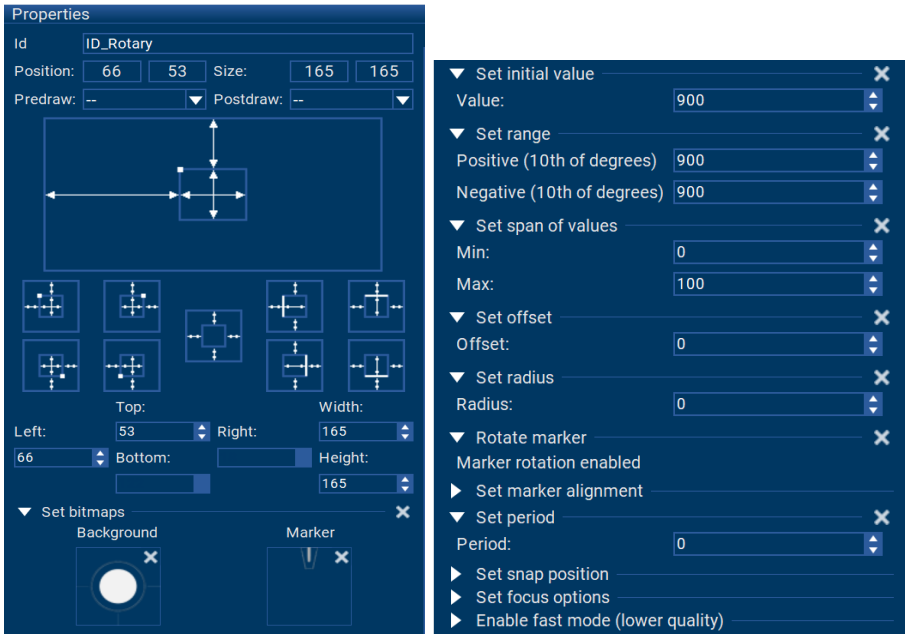


Figure 2-71 Set Properties of ID_Rotary

5. Add a Text object (TEXT_value) to display the value of Rotary (ID_Rotary), and set its properties according to Figure 2-72.

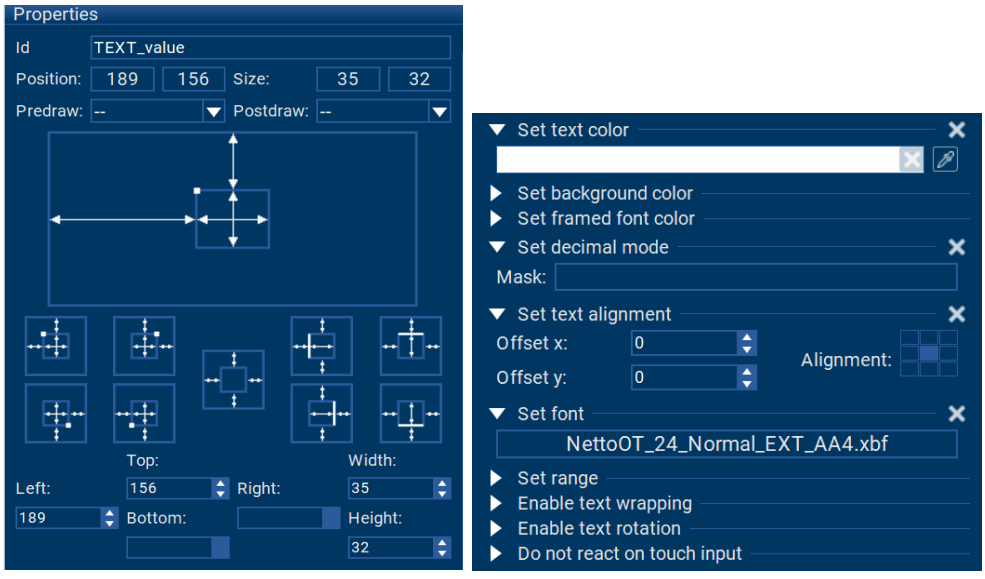


Figure 2-72 Set Properties of TEXT_value

6. Add two Switch objects (SWITCH_01, SWITCH_02) as function switches, then choose *images Swtich_Pressed_83x22.png* and *Swtich_Unpressed_83x22.png* as bitmaps of BG-Left and BG-Right respectively. Setting properties according to Figure 2-74 to Figure 2-77.

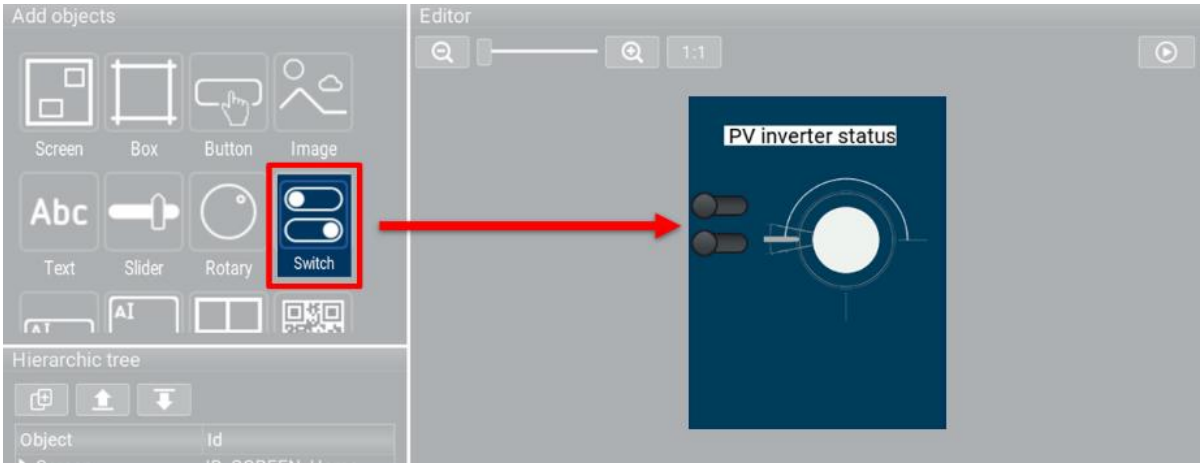


Figure 2-73 Add Switch Object

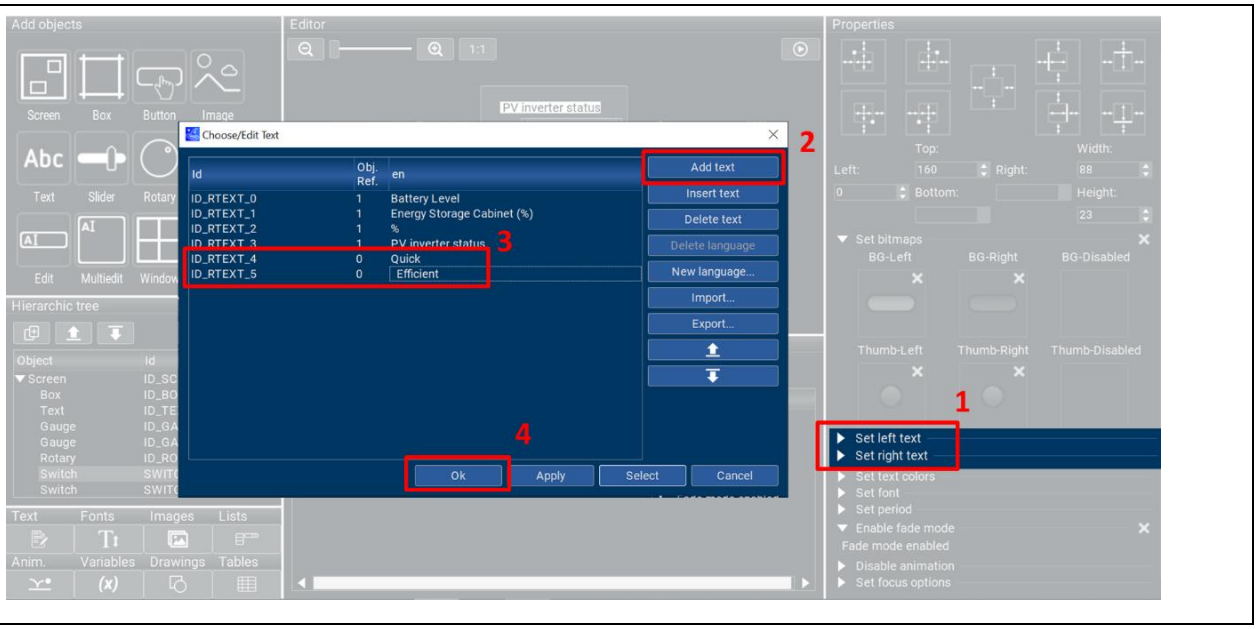


Figure 2-74 Add the Display Content of Switches

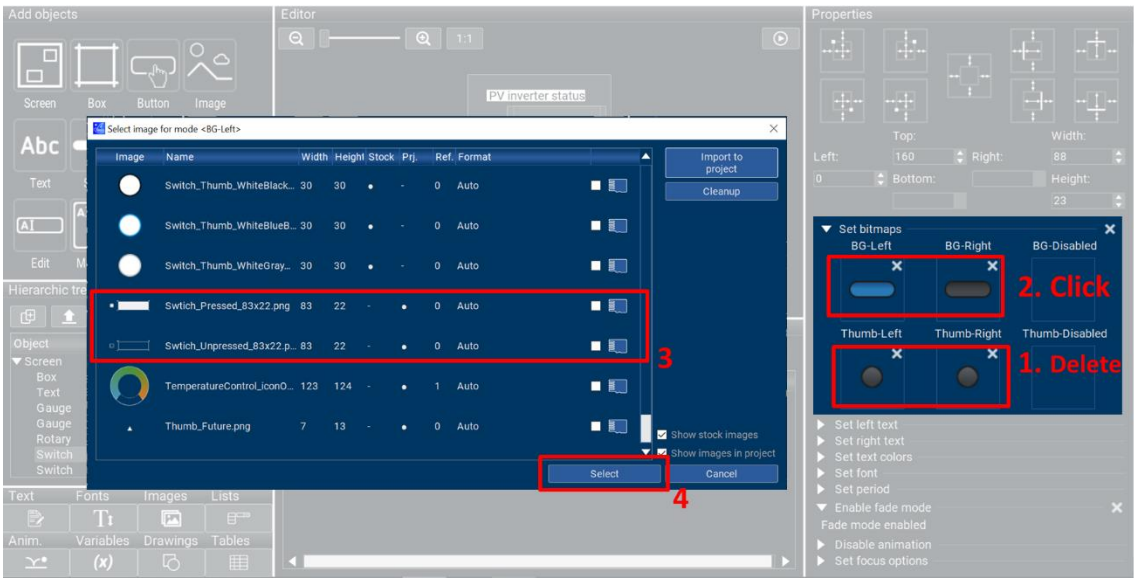


Figure 2-75 Set the Bitmap of Switch

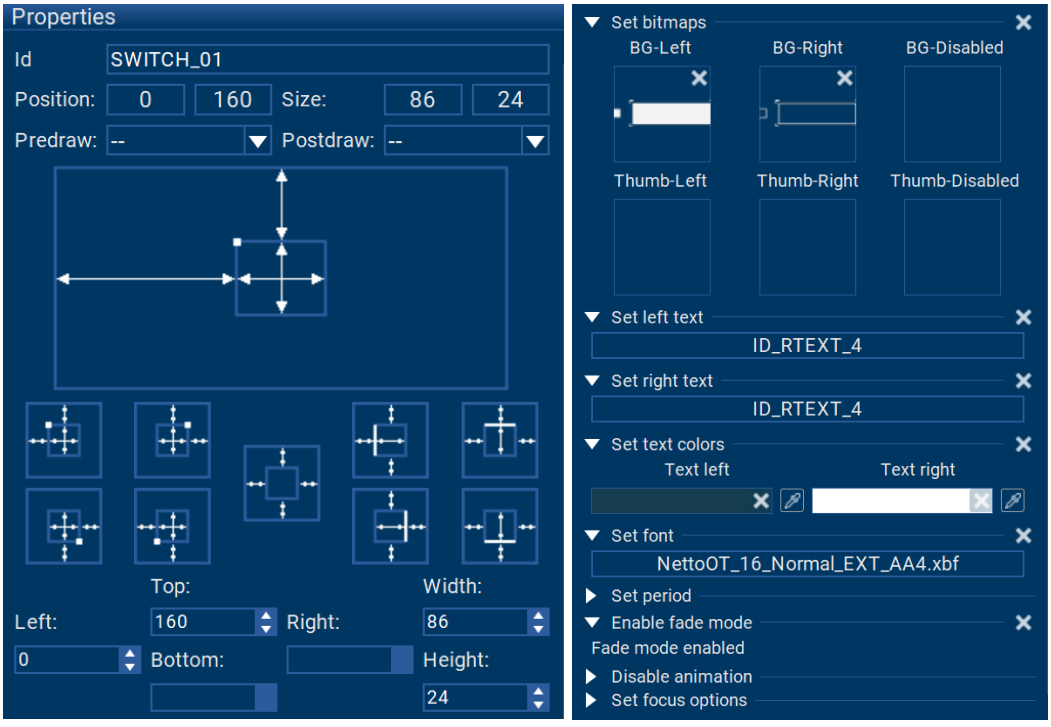


Figure 2-76 Set Properties of SWITCH_01

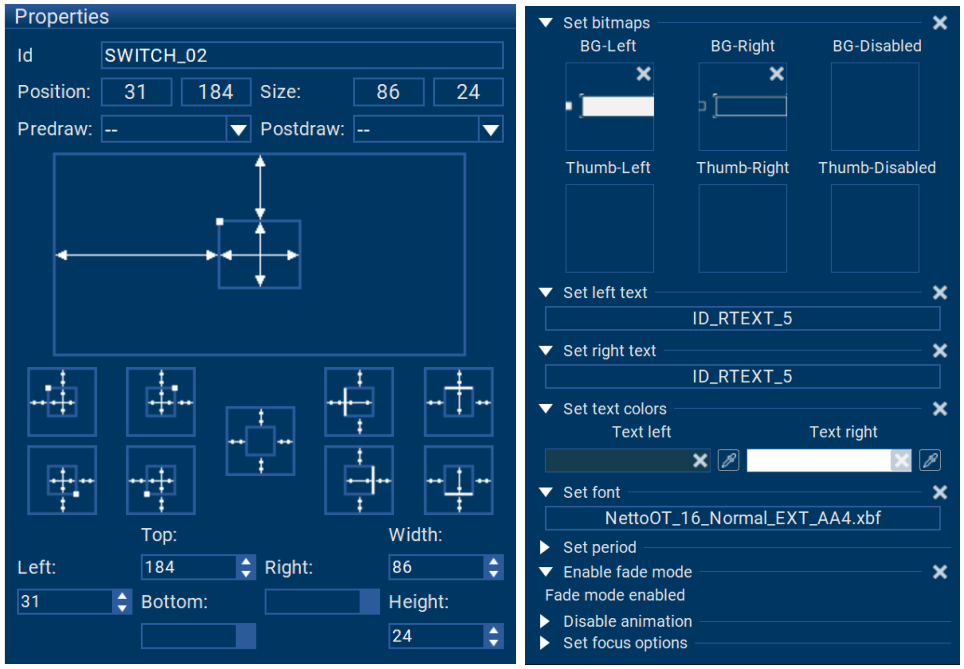


Figure 2-77 Set Properties of SWITCH_02

7. Follow the steps in Figure 2-78 to Figure 2-89 to add the control switches for the thrust bearing and guide bearing.

   1) Add two sets of Switch and Progbar objects (SWITCH_TB, PROGBAR_TB, SWITCH_GB, and PROGBAR_GB), and choose images *Future_Switch_Button_Pressed_20x84.png* and *Future_Switch_Button_Unpressed_84x23.png* as the bitmaps of Switches.

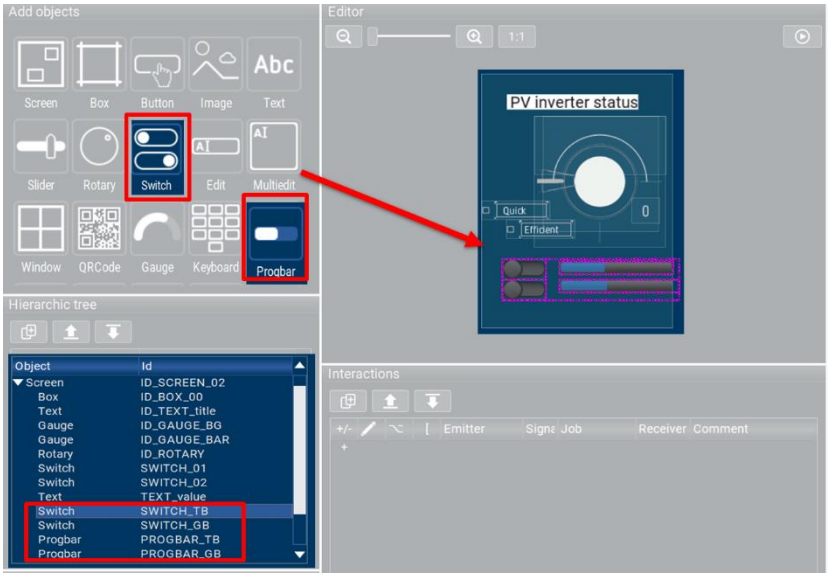


Figure 2-78 Add Two Sets of Object Switch and Progbar

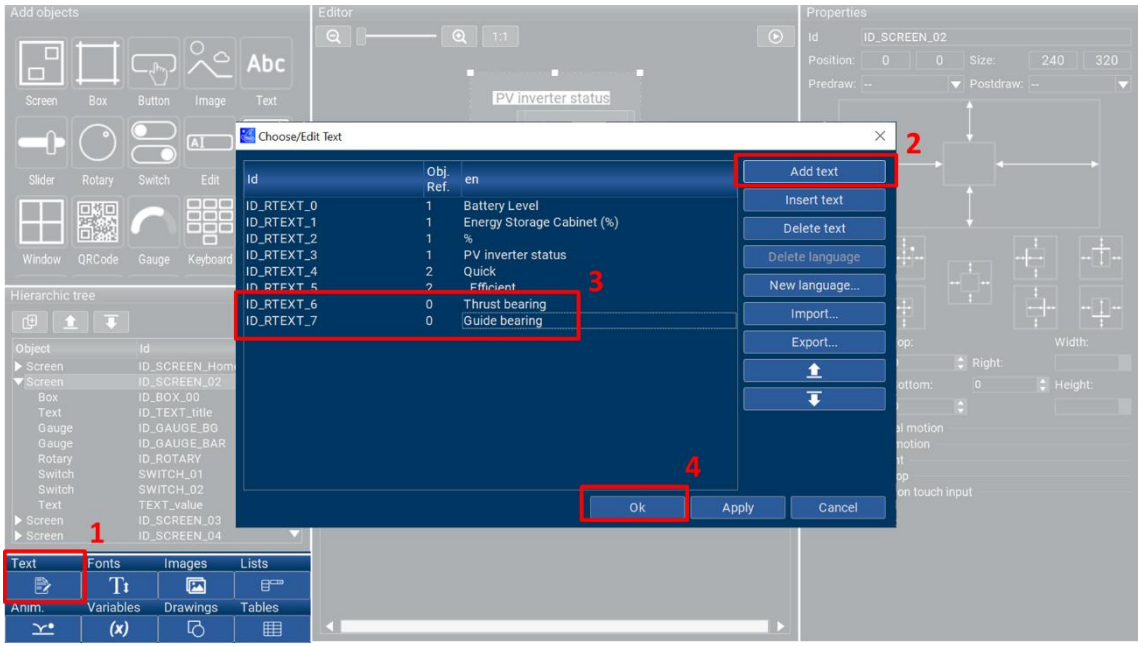


Figure 2-79 Add the Display Content of Switches

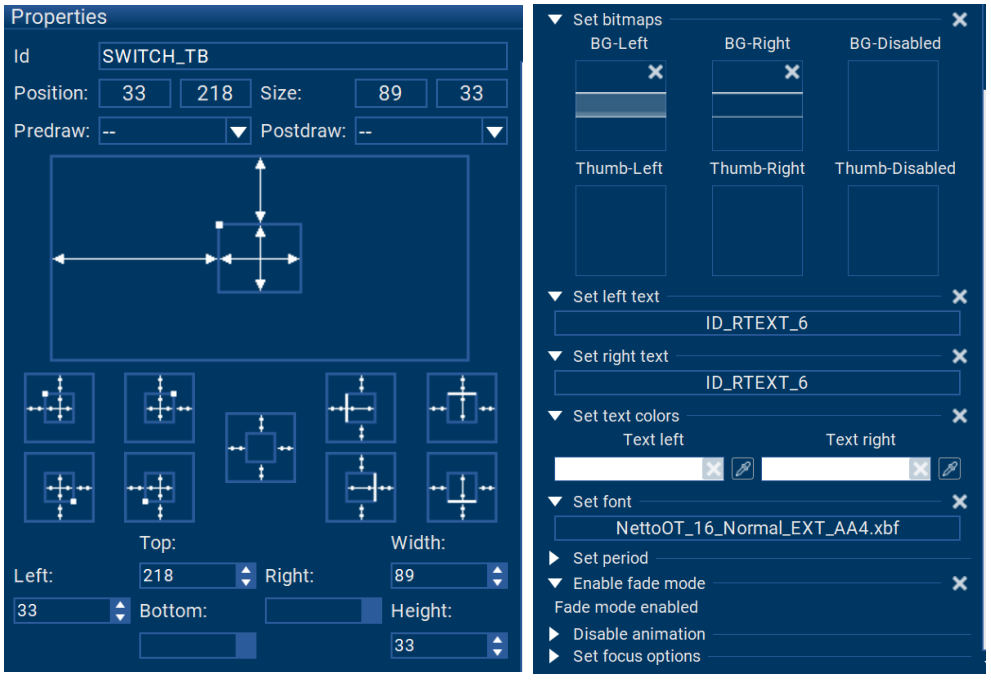


Figure 2-80 Set Properties of SWITCH_TB

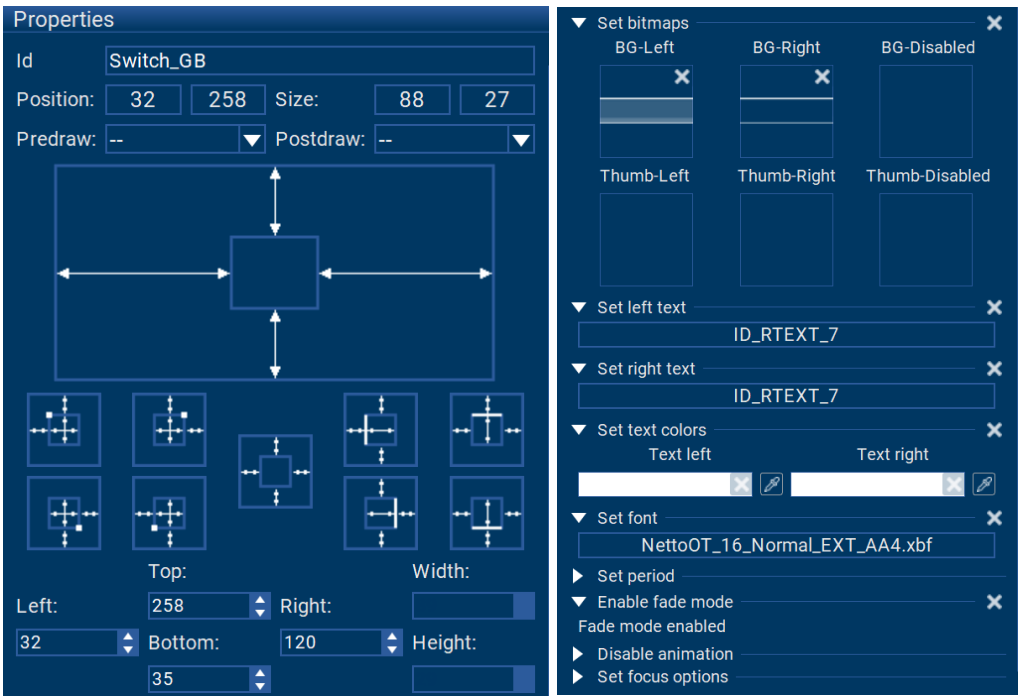


Figure 2-81 Set Properties of SWITCH_GB

2) Select image *Progbar_Future_START_80x23.png* as the bitmaps of the Progbar objects (PROGBAR_TB and PROGBAR_GB).

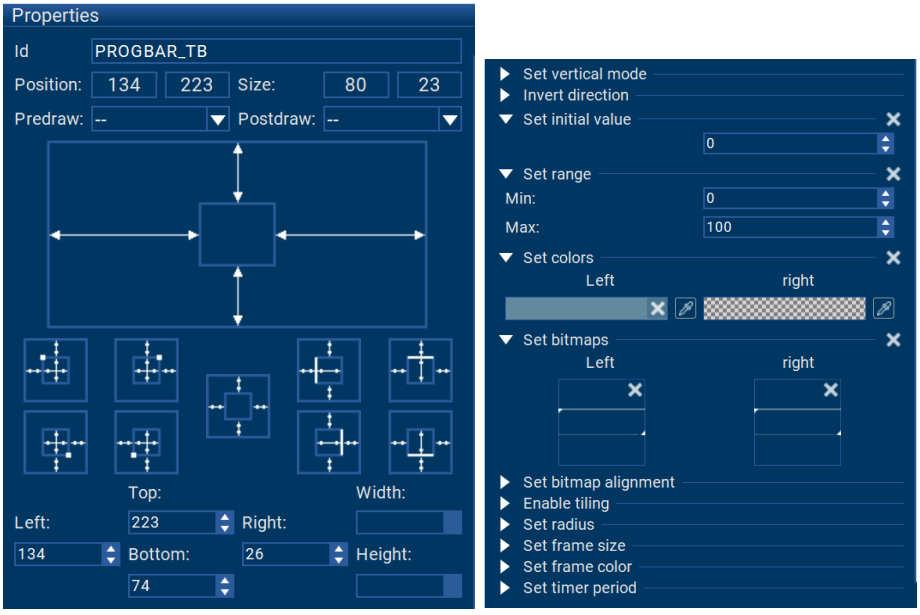


Figure 2-82 Set Properties of PROGBAR_TB

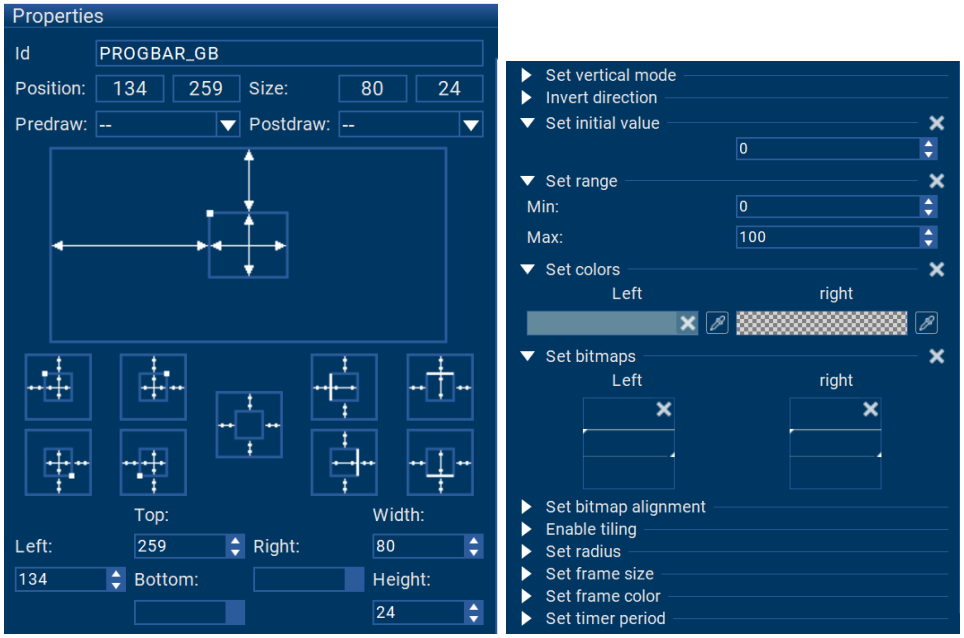


Figure 2-83 Set Properties of PROGBAR_GB

3) Add four Text objects (TEXT_num1, TEXT_unit1, TEXT_num2, and TEXT_unit2) to display values and units for a set of Switch and Progbar respectively.

4) Set the Text of Progbar (PROGBAR_TB), using TEXT_num1 and TEXT_unit1 to represent the value and unit respectively, and stack them on Progbar.

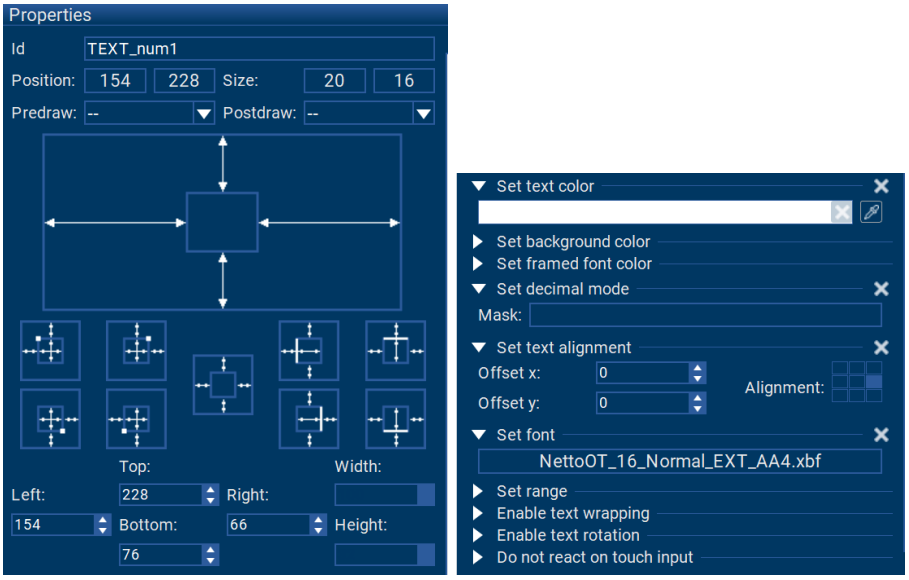


Figure 2-84 Set Properties of TEXT_num1

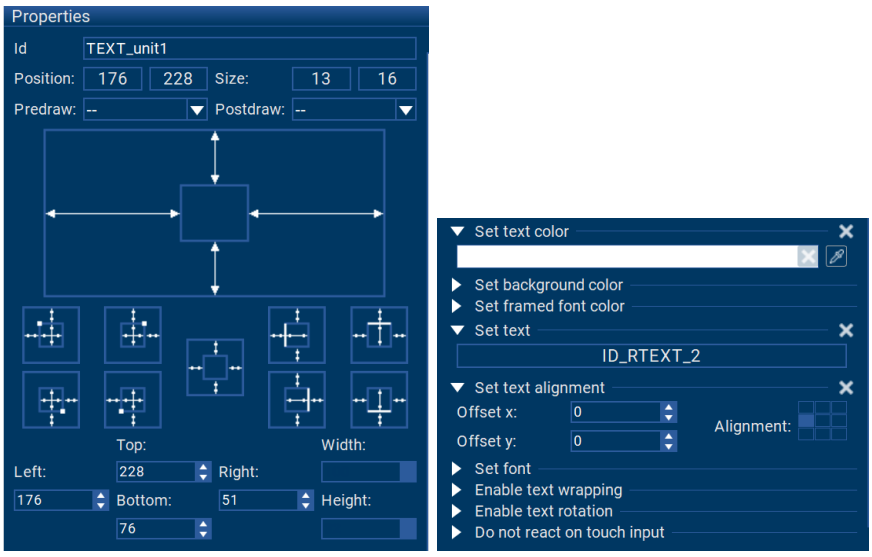


Figure 2-85 Set Properties of TEXT_unit1

5) Set the Text of Progbar (PROGBAR_GB), using TEXT_num2 and TEXT_unit2 to represent the value and unit respectively, and stack them on Progbar.

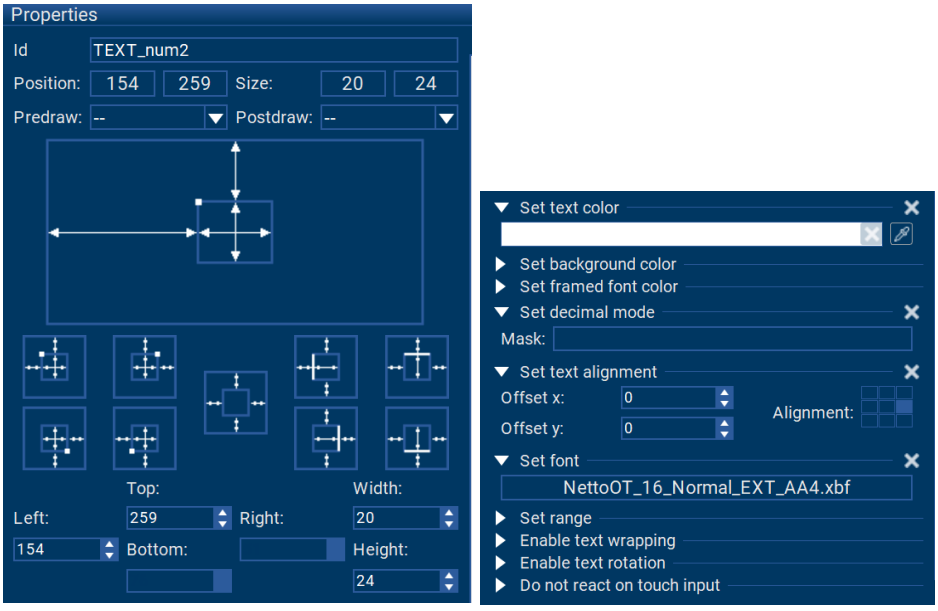


Figure 2-86 Set Properties of TEXT_num2

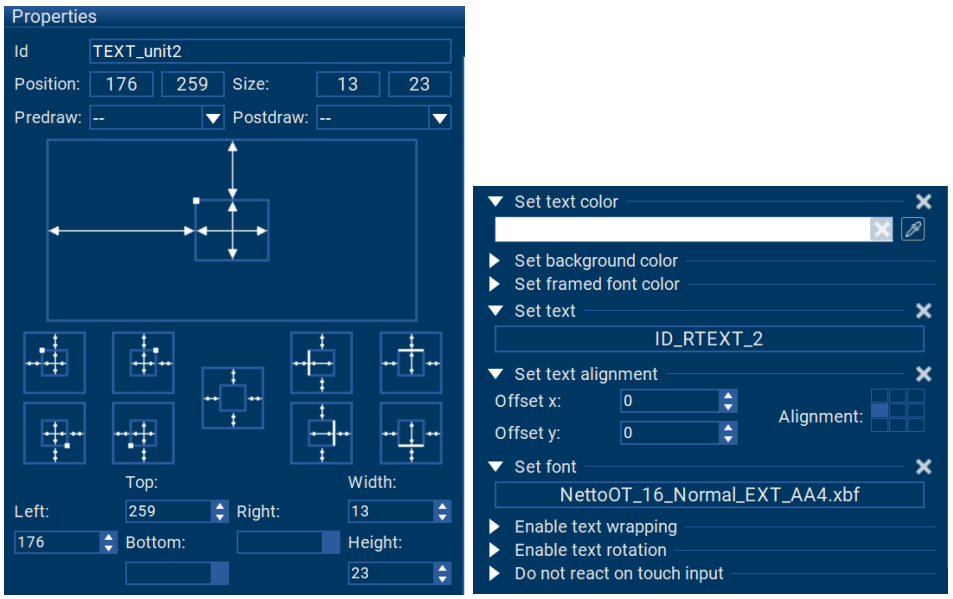


Figure 2-87 Set Properties of TEXT_unit2

6) Click on "Slider" in "Add objects" to create an adjusting switch, then choose images *Slider_Start_Future.png*, *Slider_End_Future.png*, and *Thumb_Future.png* as the bitmaps of Slider's Shaft left, Shaft right, and Thumb up.

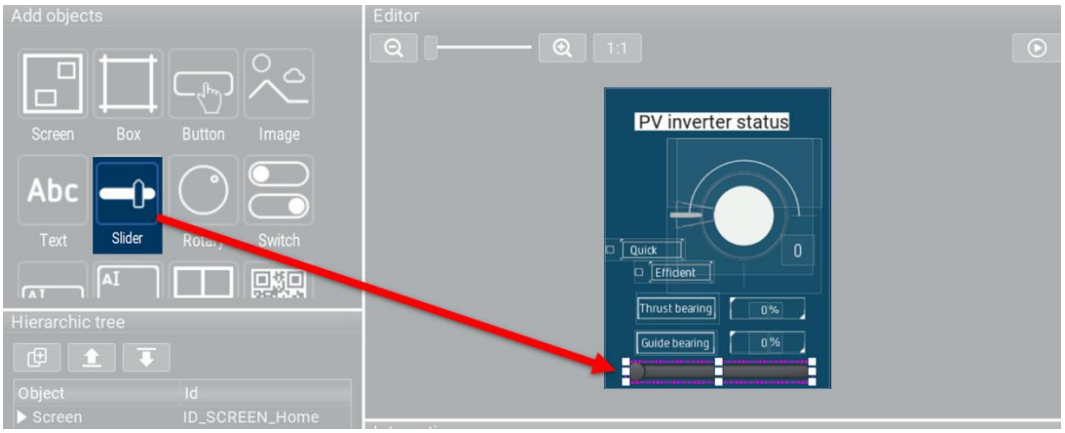


Figure 2-88 Add Slider Object

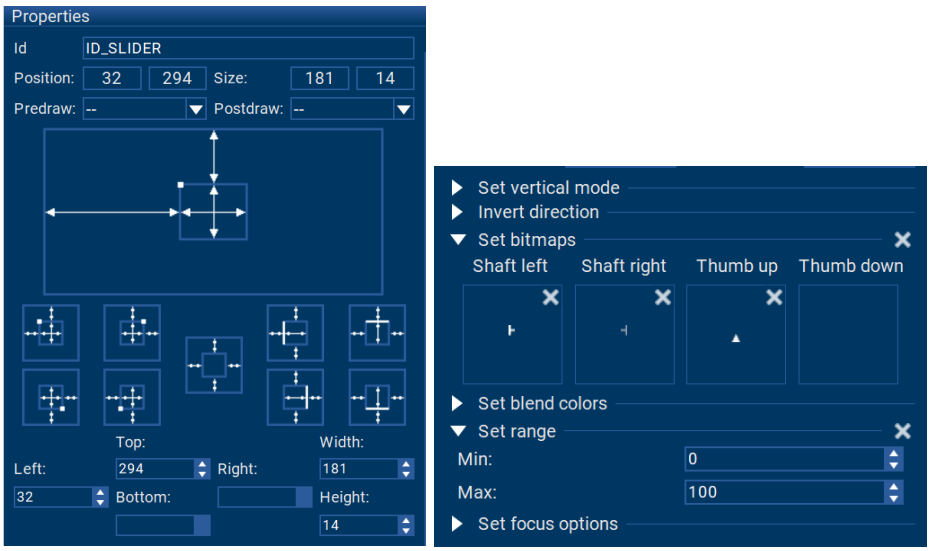


Figure 2-89 Set Properties of ID_SLIDER

8. Double-click on "Button" in "Add objects" to create screen change buttons for the previous page (btn_p2_back) and the next page (btn_p2_next). Set the images *arrow2_20x20_new.png* and *arrow1_20x20_new.png* as bitmaps for buttons.

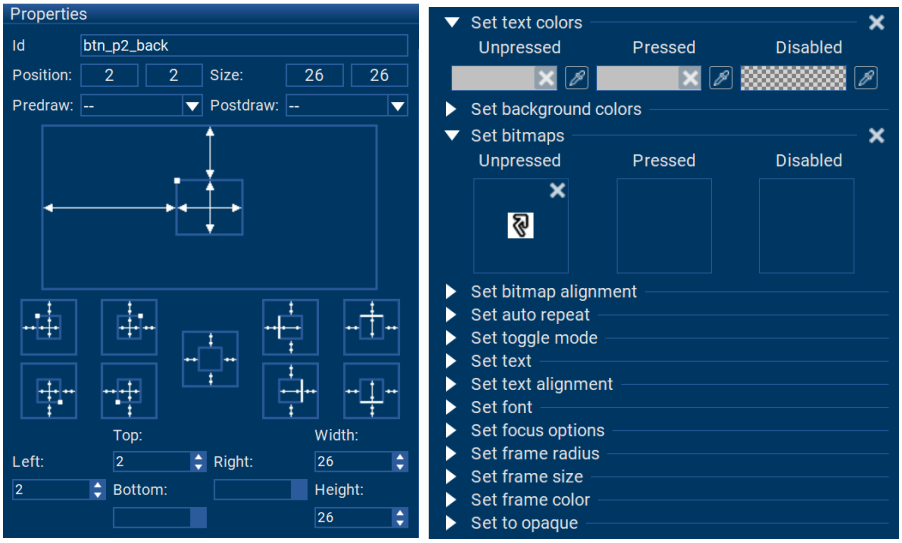


Figure 2-90 Set Properties of btn_p2_back

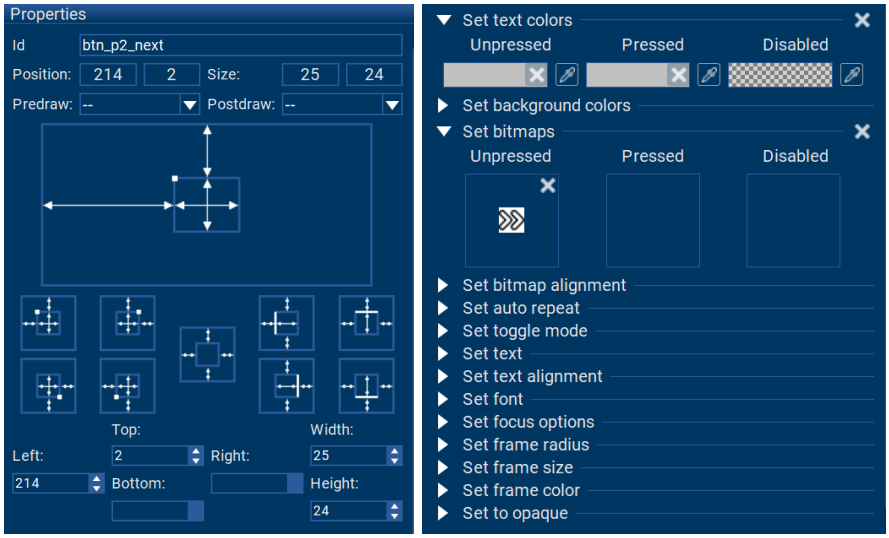


Figure 2-91 Set Properties of btn_p2_next

2.3.2.2 Interaction Settings

The completed design is shown in Figure 2-92. Next, users need to establish interactions between objects so that corresponding behaviors will be displayed when operating the objects.

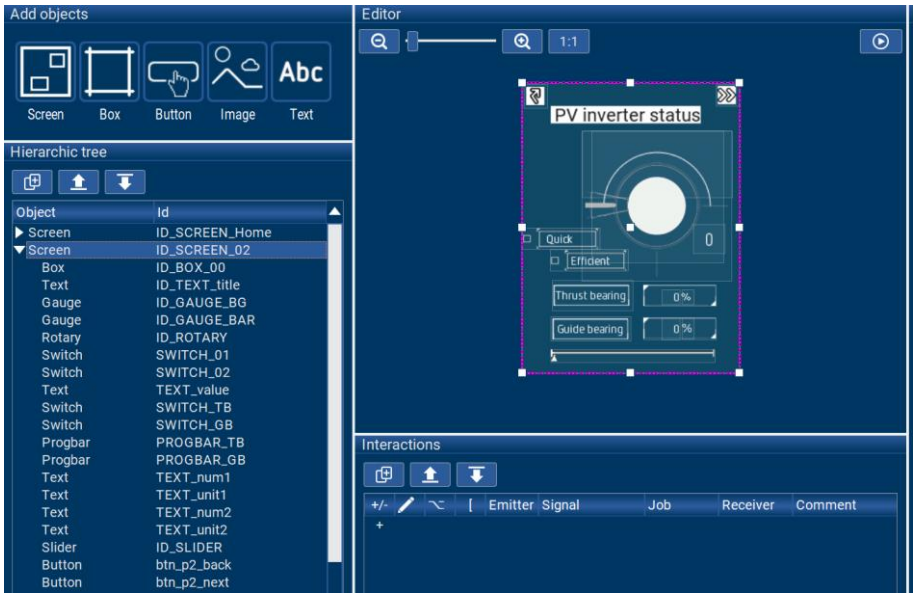


Figure 2-92 Completed Appearance of Screen 2

1. Add interactions between Rotary (ID_ROTARY), Gauge (ID_GAUGE_BAR), and TEXT (TEXT_value) objects, let the Gauge value change as the Rotary rotates, and display the value by Text simultaneously.

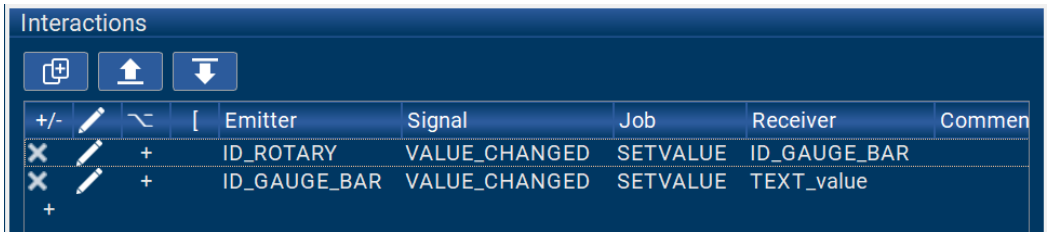


Figure 2-93 Interactions between Rotary, Gauge, and TEXT Objects

2. Set interactions between two sets of Progbar (PROGBAR_TB, PROGBAR_GB) and Text (TEXT_num1, TEXT_num2) objects, and let the display value of Text change follow Progbar.

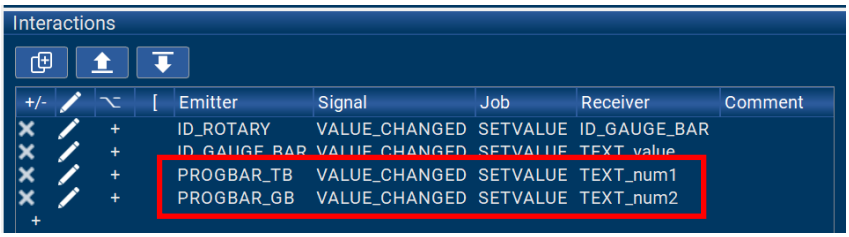


Figure 2-94 Interactions between Progbar and TEXT Object

3. Set interactions between Slider(ID_SLIDER), Progbar (PROGBAR_TB, PROGBAR_GB), and Switch(SWITCH_TB, SWITCH_GB) objects, let users to adjust the Progbar value through Slider, and use Switch to change the Progbar which is to be mapped the value.

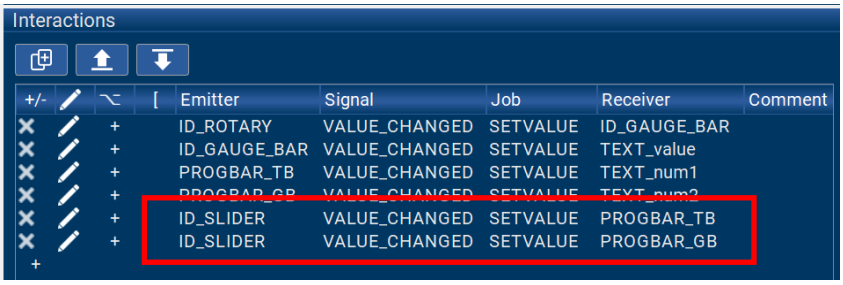


Figure 2-95 Interactions between Slider and Progbar Object

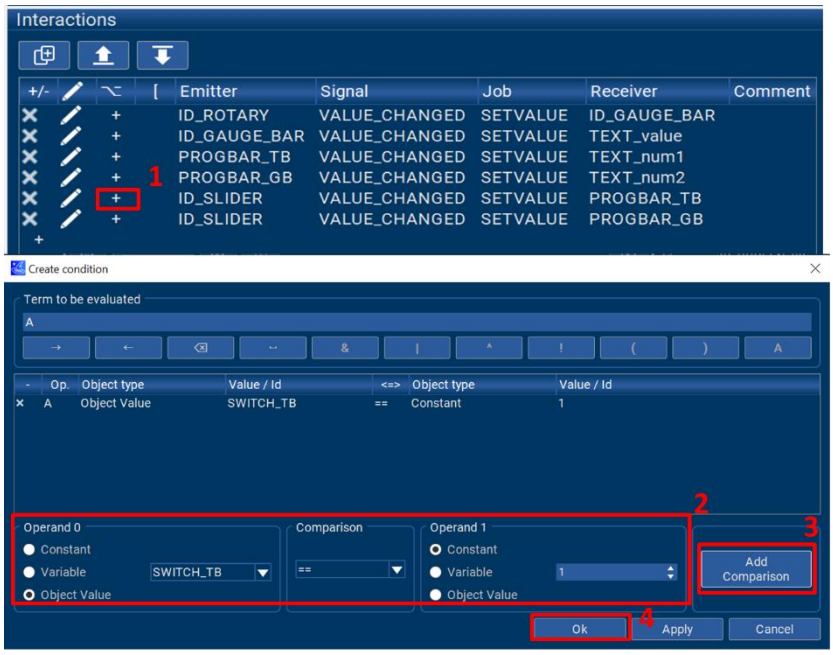


Figure 2-96 Switch Setting of the Slider and Switch Objects (1)

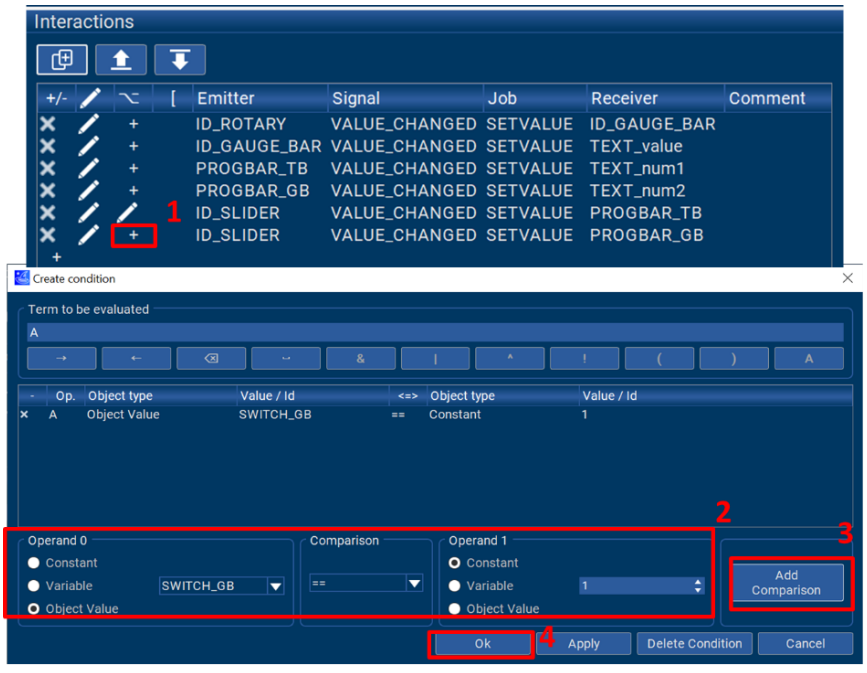


Figure 2-97 Switch Setting of the Slider and Switch Objects (2)

4. Set the screen-switching interaction by button btn_p2_back and btn_p2_next, make users can switch previous (ID_SCREEN_Home) and next (ID_SCREEN_03) screens.

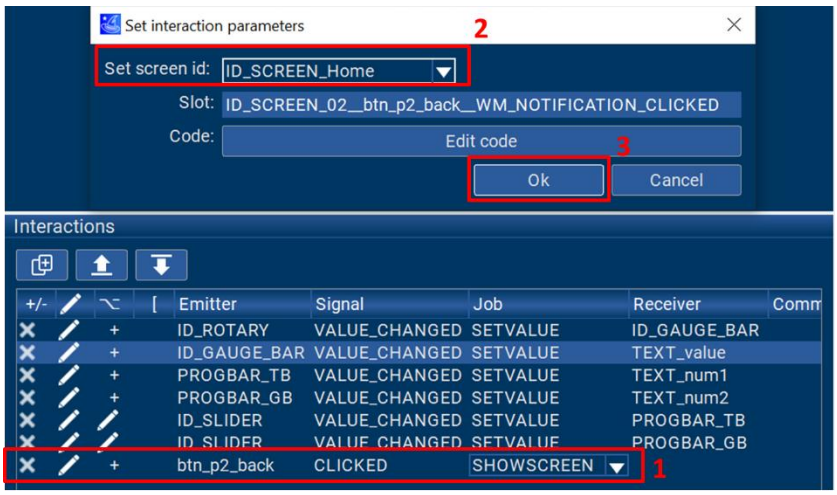


Figure 2-98 Button Settings of Screen Switching(btn_p2_back)

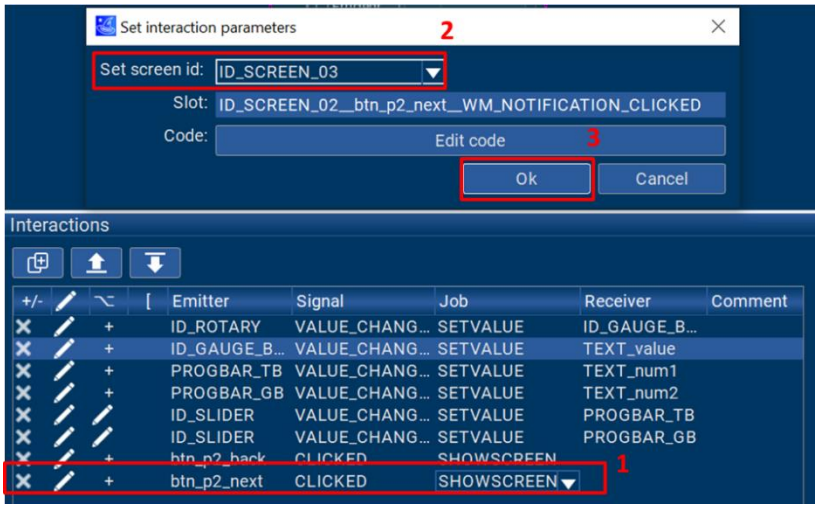


Figure 2-99 Button Settings of Screen Switching (btn_p2_next)

2.3.3 Design the Display of Three-phase Current (Screen 3)

Most three-phase current applications need to display power waveforms. Since AppWizard currently does not support Graphic objects, the waveform display needs to be designed with GUI_Builder through the emWin library, and then integrated the emWin dialog into the AppWizard project. This section will introduce how to embed an emWin dialog into AppWizard project in details. The widgets used in this screen are shown in Table 2-4 to Table 2-5.

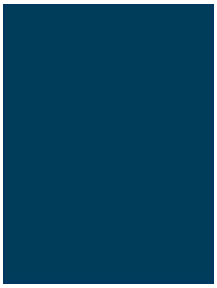
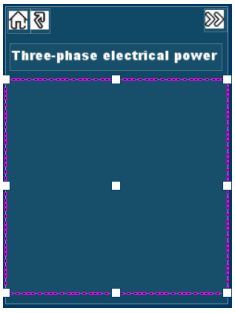
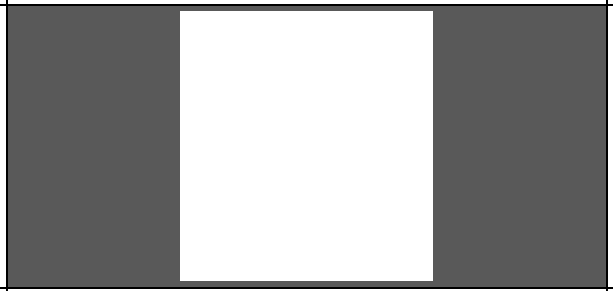
New Energy Gateway Description	Application GUI	Widget Type
Background (240x320)		Box
Text of title		Text
Reserved space of emWin dialog		window
Button of home page		Button
Button of previous page		
Button of next page		

Table 2-4 Widgets of Display of Three-Phase Current (AppWizard)

New Energy Gateway Description	Application GUI	Object Type
Dialog base(240x230)		window
Options for showing waveform		Check Box

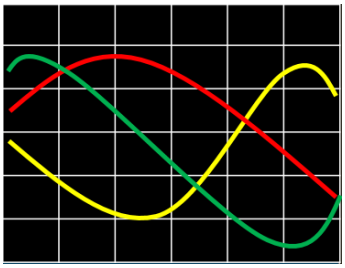
Waveform showing		Graphic

Table 2-5 Widgets of Display of Three-Phase Current (emWin)

2.3.3.1 Widgets Configuration of EmWin Dialog

1. Open *Gui_Builder.exe* and add a Window object with a size of 240*220 as the base of the emWin dialog. (IDE Location: *\EC_NUC980_AppWizard_New_Energy_Gateway_V1.00\Software Tools\GUI_builder*)

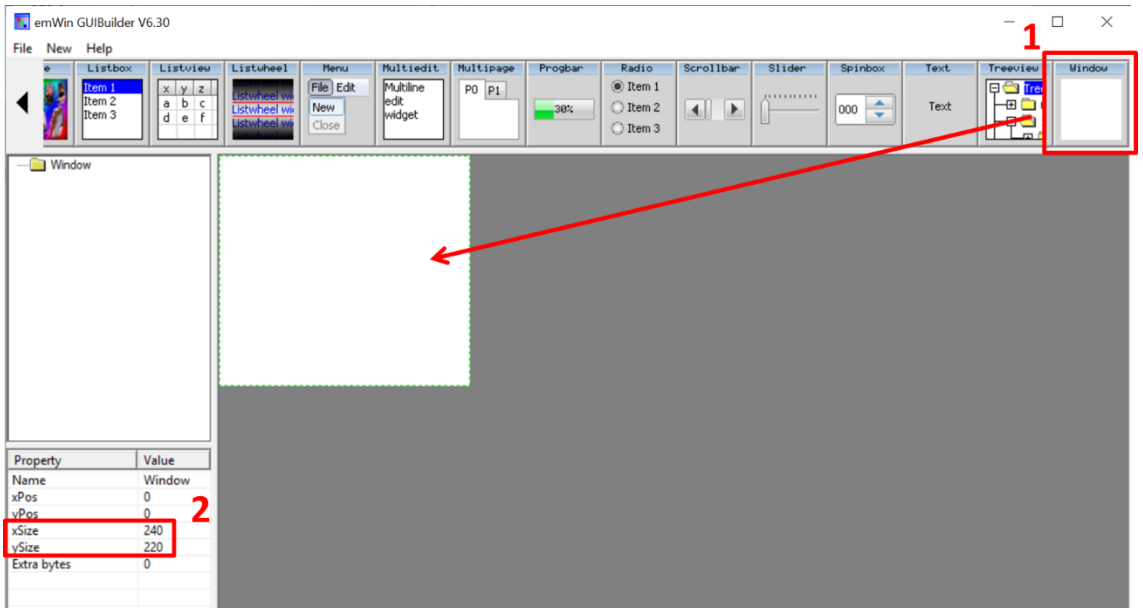


Figure 2-100 Add Window Object and Adjust Size

2. Add a new Graphic object with size 240x174 for drawing waveforms, and align its position below the Window object.

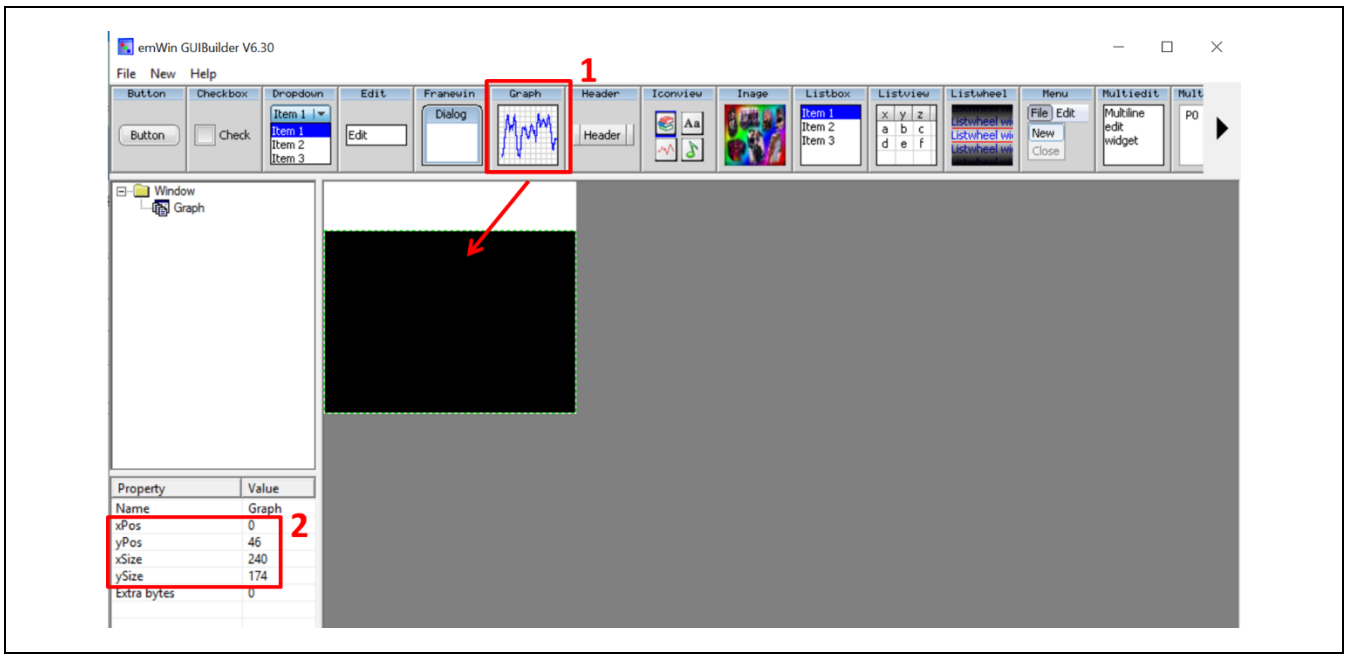


Figure 2-101 Add Graphic Object and Adjust Position

3. Add three Checkbox objects as function switches for each waveform display.

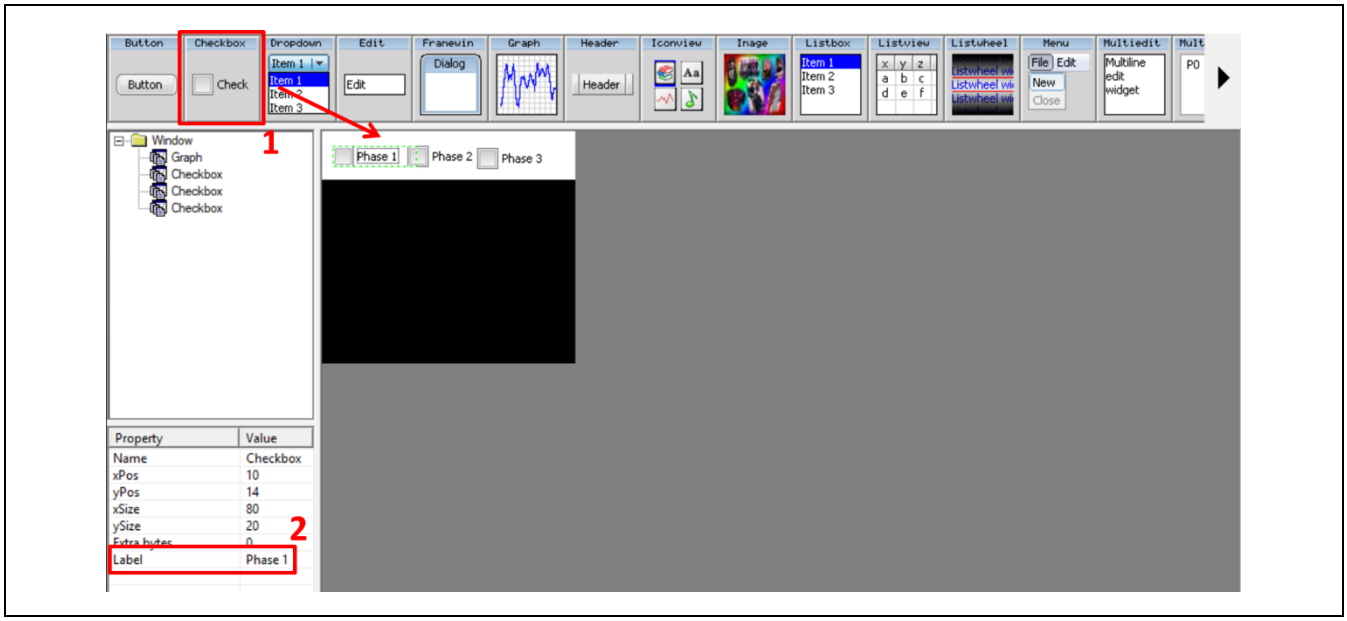


Figure 2-102 Add Checkbox Objects and Adjust Display Name by Label

4. After the design of the GUI, saving dialog by toolbar. The file will be saved in the same directory as GUI_Builder.exe automatically.

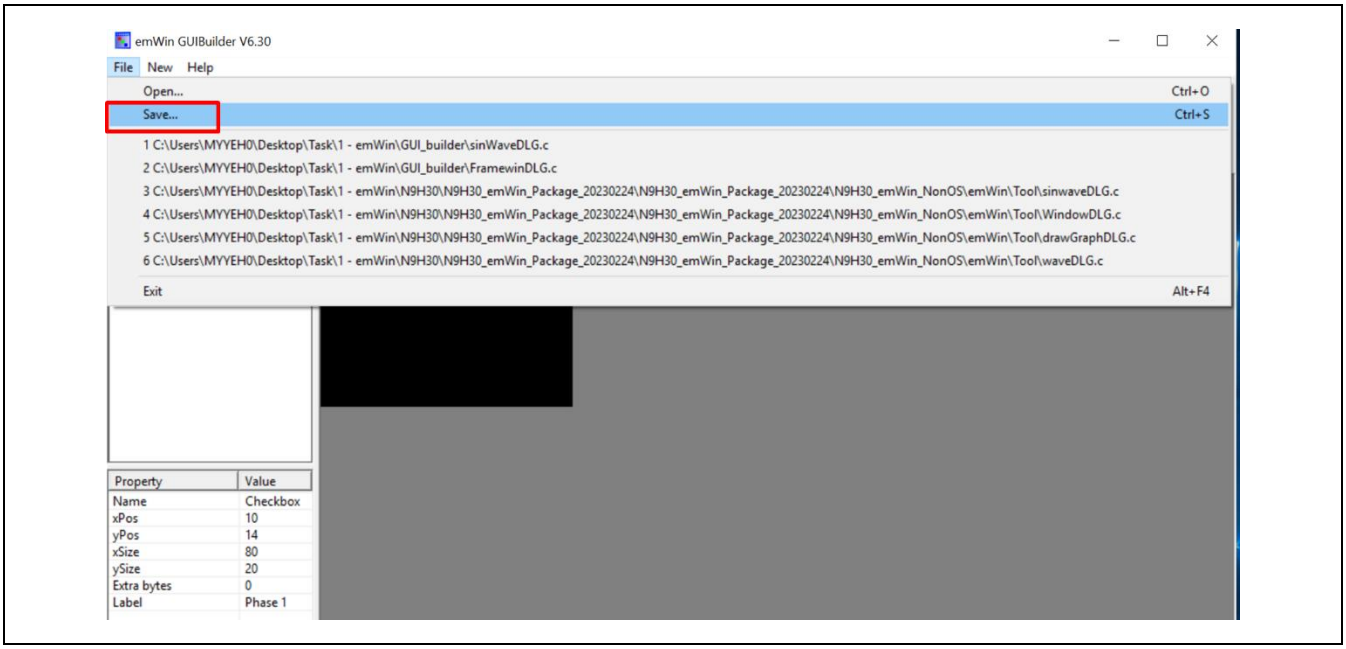


Figure 2-103 Save the emWin Dialog

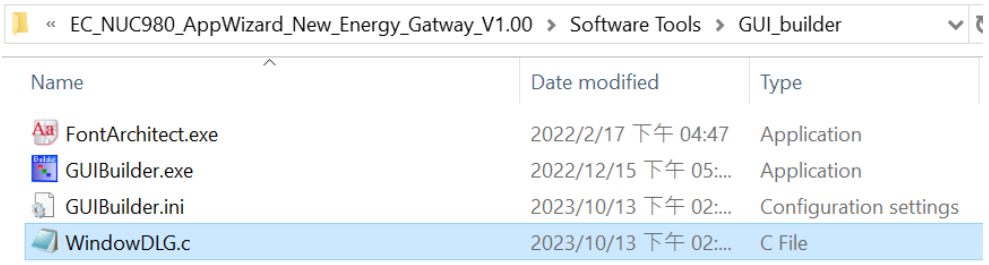


Figure 2-104 Default Storage Location of EmWin Dialog

5. Open the emWin dialog file (*DLG.c*). Add code to control the drawing of the waveform in the _cbDialog function and set the function to display the specified waveline by switching the checkbox.

   1) Declare the array ahData[3] to control the waveform. (If want to display the value of waveforms, users can add another array to store and then declarations-related variables in AppWiazrd. For the details of the method, please refer to the AppWizard official example: *HowTo_AddDialog*)

```
static void _cbDialog(WM_MESSAGE * pMsg) {
  WM_HWIN hItem;
  int     NCode;
  int     Id;
  // USER START (Optionally insert additional variables)
  static GRAPH_DATA_Handle ahData[3];
  static int               a;
  int                      v;
```

```
  //U16                   aIds[3] = { ID_VAR_V1, ID_VAR_V2, ID_VAR_V3 };  //Optional
  int                     i;
  // USER END
```

2) Set the initial value of the variable in “case WM_INIT_DIALOG”.

```
switch (pMsg->MsgId) {
  case WM_INIT_DIALOG:
    …

    hItem = WM_GetDialogItem(pMsg->hWin, ID_CHECKBOX_2);
    CHECKBOX_SetText(hItem, "Phase 3");

    // USER START (Optionally insert additional code for further widget initialization)
    hItem     = WM_GetDialogItem(pMsg->hWin, ID_GRAPH_0);
    v         = WM_GetWindowSizeX(pMsg->hWin);
    ahData[0] = GRAPH_DATA_YT_Create(GUI_RED,    v, NULL, 0);
    ahData[1] = GRAPH_DATA_YT_Create(GUI_GREEN,  v, NULL, 0);
    ahData[2] = GRAPH_DATA_YT_Create(GUI_YELLOW, v, NULL, 0);
    GRAPH_SetGridVis(hItem, 1);
    WM_CreateTimer(pMsg->hWin, 0, 25, 0);
    // USER END
    break;
```

3) Add the "case WM_TIMER" in the "switch (pMsg->MsgId)", and the waveform value will be updated over time.

```
static void _cbDialog(WM_MESSAGE * pMsg) {
 …

switch (pMsg->MsgId) {
    …
    // USER START (Optionally insert additional message handling)
    case WM_TIMER:
       a++;
       if (a == 360) { a = 0; }
       for (i = 0; i < (int)GUI_COUNTOF(ahData); i++) {
           v = (50 * GUI__SinHQ((a + (i * 90)) * 1000)) >> 16;
           v += 115 / 2;
           GRAPH_DATA_YT_AddValue(ahData[i], v);
           //APPW_SetVarData(aIds[i], a + (i * 90));  //Optional
       }
       WM_RestartTimer((WM_HTIMER)pMsg->Data.v, 0);
       break;
    // USER END
…
```

4) Add a Switch-Case in "case WM_NOTIFY_PARENT" to enable the program to update the value of the GRAPHIC object according to the checkbox status.

```
switch (pMsg->MsgId) {
    …
  case WM_NOTIFY_PARENT:
    …
    switch(Id) {
        case ID_CHECKBOX_0: // Notifications sent by 'Checkbox'
          switch(NCode) {
                …
                case WM_NOTIFICATION_VALUE_CHANGED:
```

```
            // USER START
            hItem = WM_GetDialogItem(pMsg->hWin, ID_GRAPH_0);
            if (CHECKBOX_GetState(pMsg->hWinSrc) == 1) {
              GRAPH_AttachData(hItem, ahData[0]);
            } else {
              GRAPH_DetachData(hItem, ahData[0]);
            }
            // USER END
            break;
    case ID_CHECKBOX_1: // Notifications sent by 'Checkbox'
      switch(NCode) {
            …
            case WM_NOTIFICATION_VALUE_CHANGED:
              // USER START
              hItem = WM_GetDialogItem(pMsg->hWin, ID_GRAPH_0);
                GRAPH_AttachData(hItem, ahData[1]);
              } else {
              GRAPH_DetachData(hItem, ahData[1]);
              }
              // USER END
              break;

    case ID_CHECKBOX_2: // Notifications sent by 'Checkbox'
      switch(NCode) {
            …
            case WM_NOTIFICATION_VALUE_CHANGED:
                // USER START
               hItem = WM_GetDialogItem(pMsg->hWin, ID_GRAPH_0);
                if (CHECKBOX_GetState(pMsg->hWinSrc) == 1) {
                    GRAPH_AttachData(hItem, ahData[2]);
                } else {
                   GRAPH_DetachData(hItem, ahData[2]);
                }
                // USER END
            break;
                // USER START
                // USER END
```

5) Place the updated emWin dialog file under the path of the AppWizard project (*/Source/CustomCode*), the location is shown in Figure 2-105.

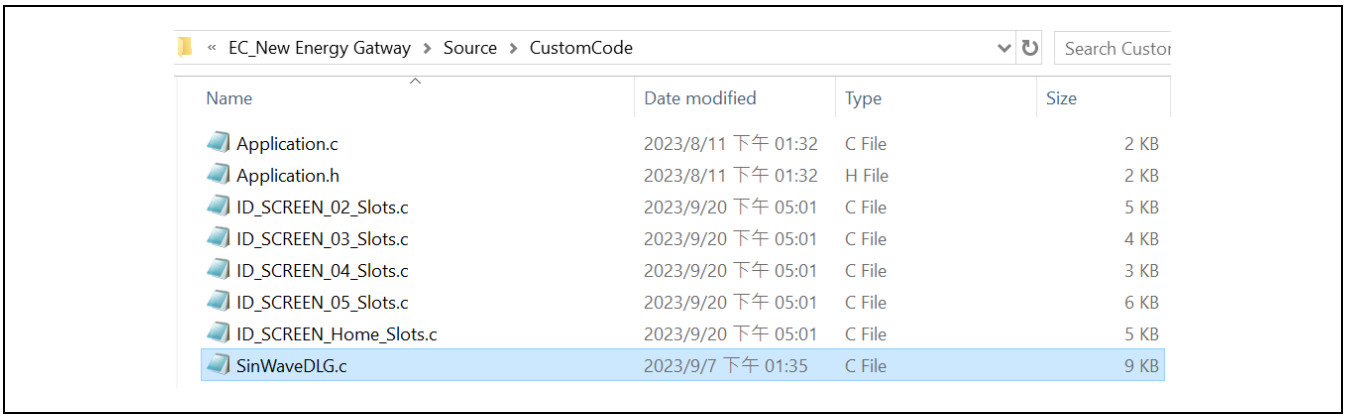


Figure 2-105 The Location to Place the EmWin Dialog File

2.3.3.2 Widgets Configuration of AppWizard

1. Choose ID_SCREEN_03 to start editing screen 3, click on "Button" in "Add objects" to create screen change buttons for the previous page (btn_p3_back), the next page (btn_p3_next), and the home page (btn_p3_home). Set the images *home_20x20_new.png*, *arrow2_20x20_new.png*, and *arrow1_20x20_new.png* as their bitmap and set the properties according to Figure 2-106 to Figure 2-108.

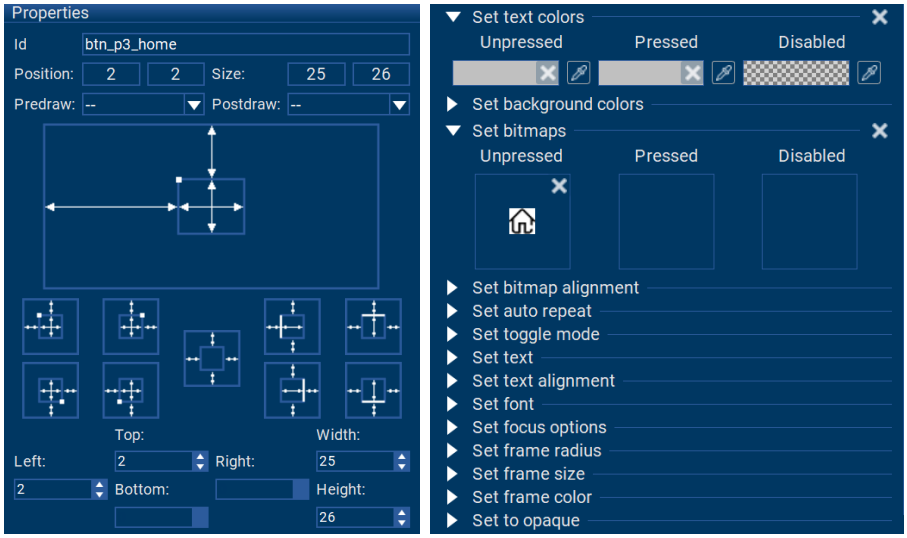


Figure 2-106 Set Properties of btn_p3_home

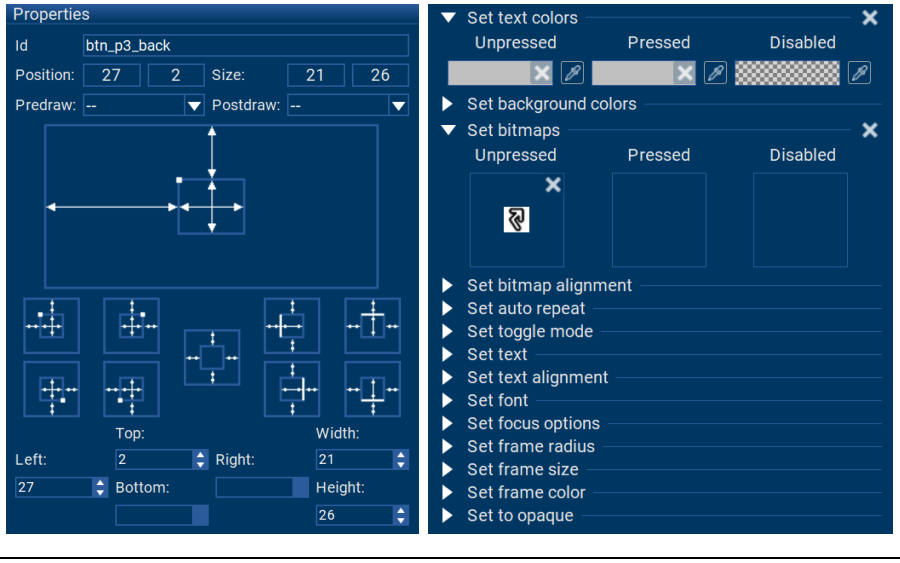


Figure 2-107 Set Properties of btn_p3_back

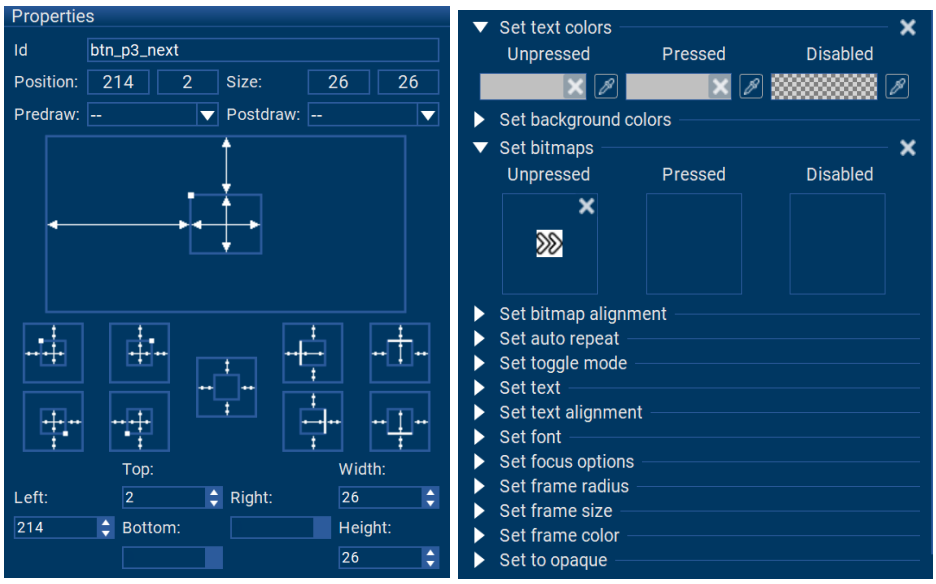


Figure 2-108 Set Properties of btn_p3_next

2. To embed the emWin dialog in AppWizard, add a window object (emwin_window) with the same size as the emWin dialog, and set the object properties as shown in Figure 2-110.

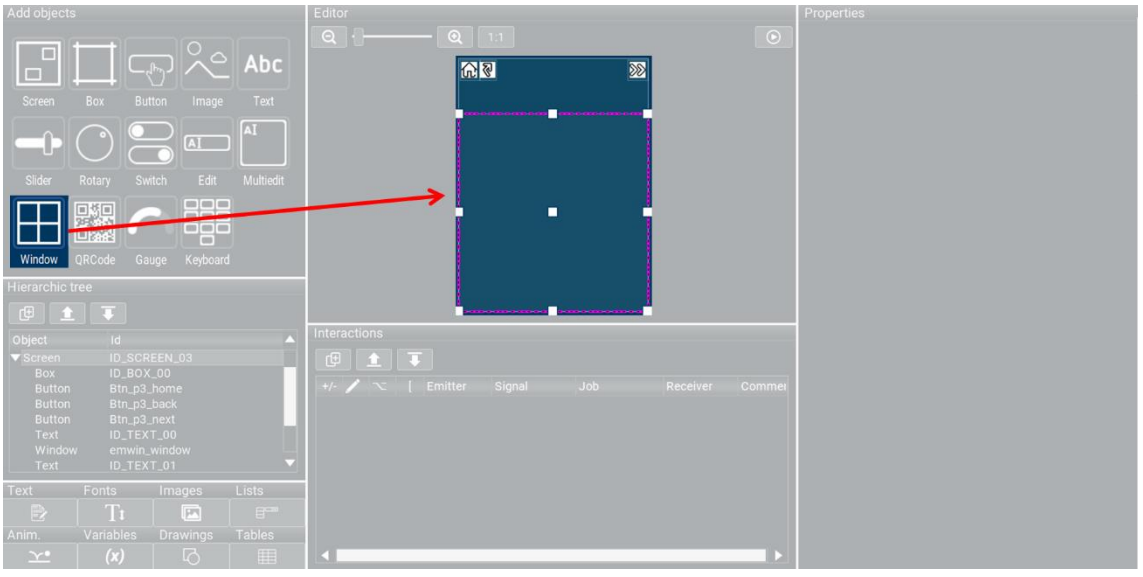


Figure 2-109 Add Window Object

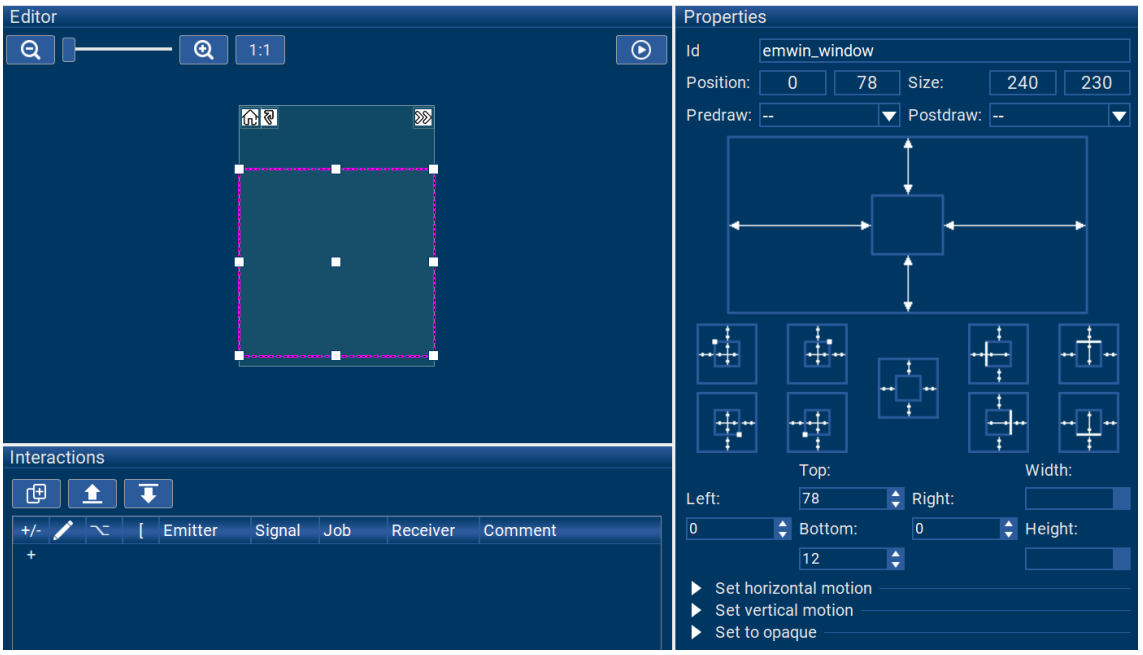


Figure 2-110 Set Properties of emwin_window

3. Click on "Text" (ID_TEXT_00) in "Add objects" to create a display title, and set the display content as "Three-phase electrical power" and properties by Figure 2-111 to Figure 2-113.

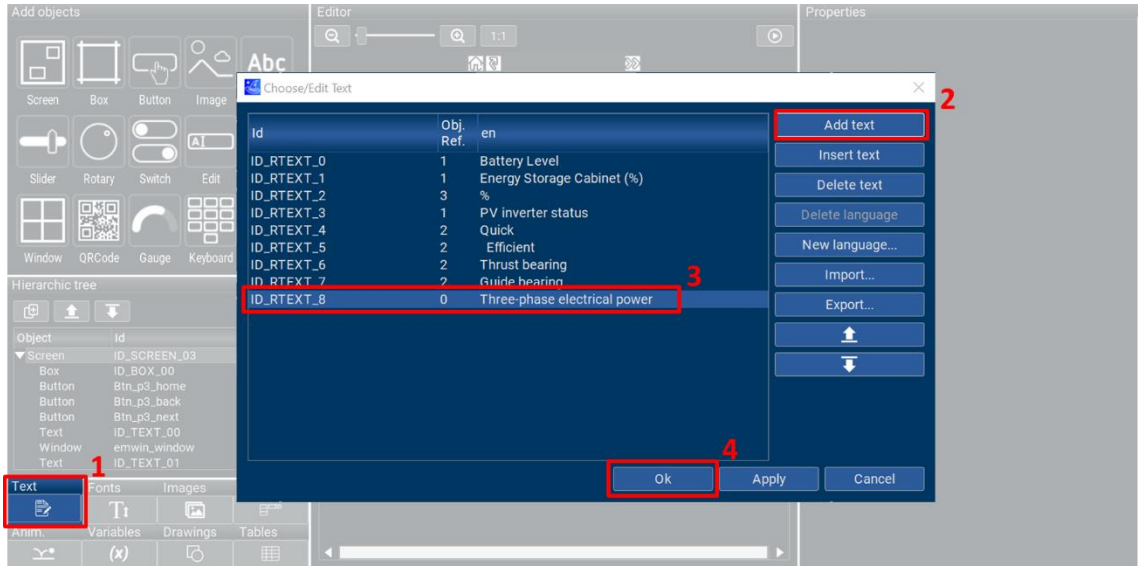


Figure 2-111 Add Display Content (ID_RTEXT_8)

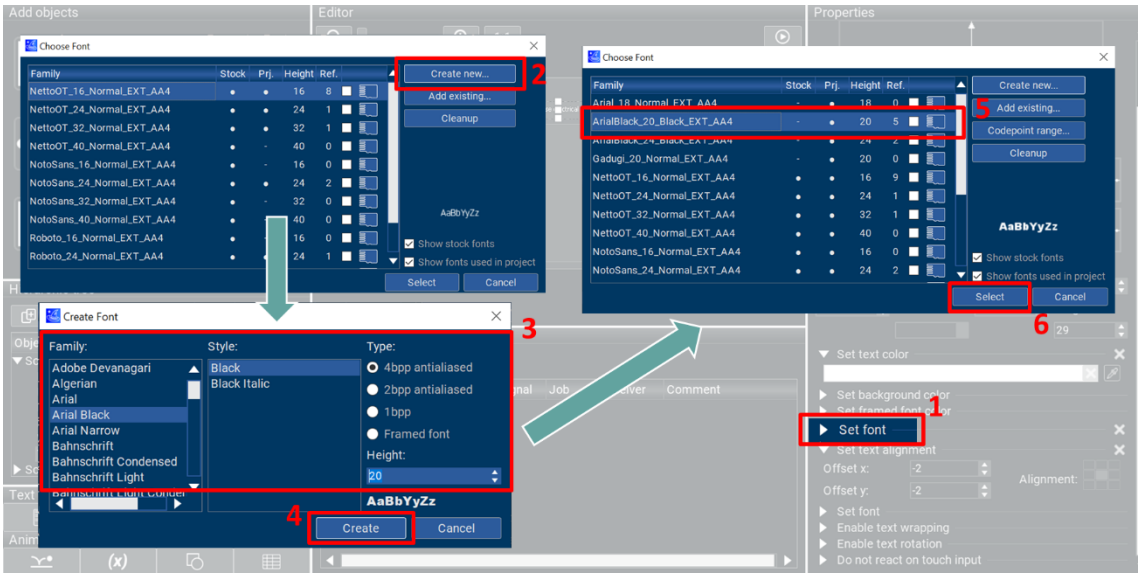


Figure 2-112 Add Custom Format of Text

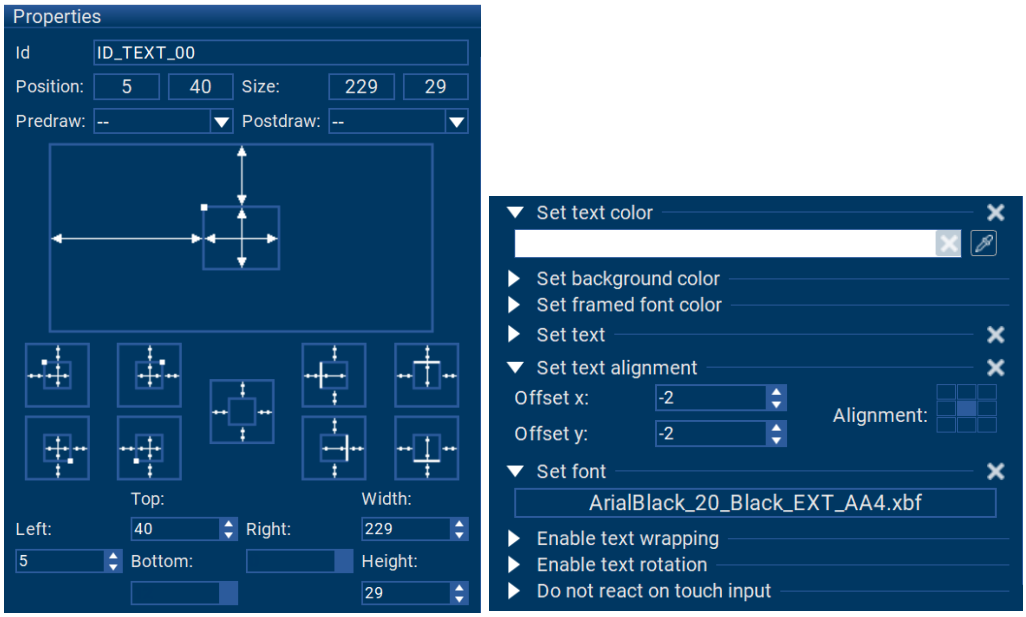


Figure 2-113 Set Properties of ID_TEXT_00

2.3.3.3 Embed an EmWin Dialog into AppWizard Project

It's necessary to embed emWin dialog into an AppWizard project by operating the interactions of AppWizard. The detailed process is as follows.

1. Add new interactions with Emitter "ID_SCREEN_03" (the screen ID to be embedded in the emwin dialog), Signal as "INITDIALOG", and Job as "NULL", then set the slot name as ID_SCREEN_03_Init. (Slot represents the function name of this action in ID_SCREEN_03_slot.)

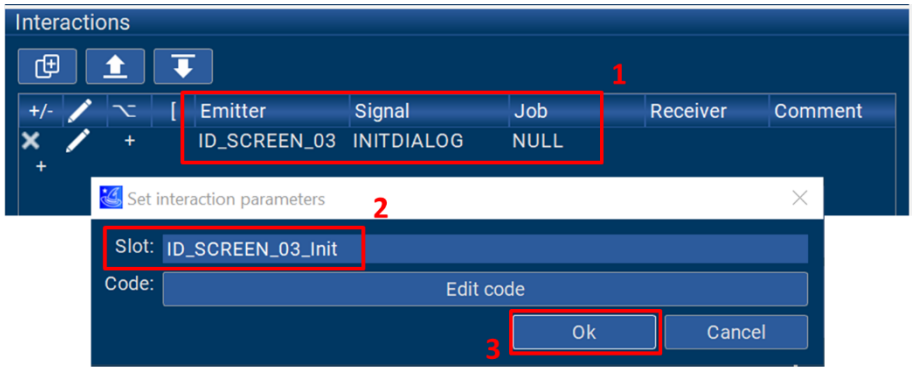


Figure 2-114 Interaction Settings of EmWin Dialog

2. Export the current project so that the program can integrate the emWin dialog settings into the AppWizard project.

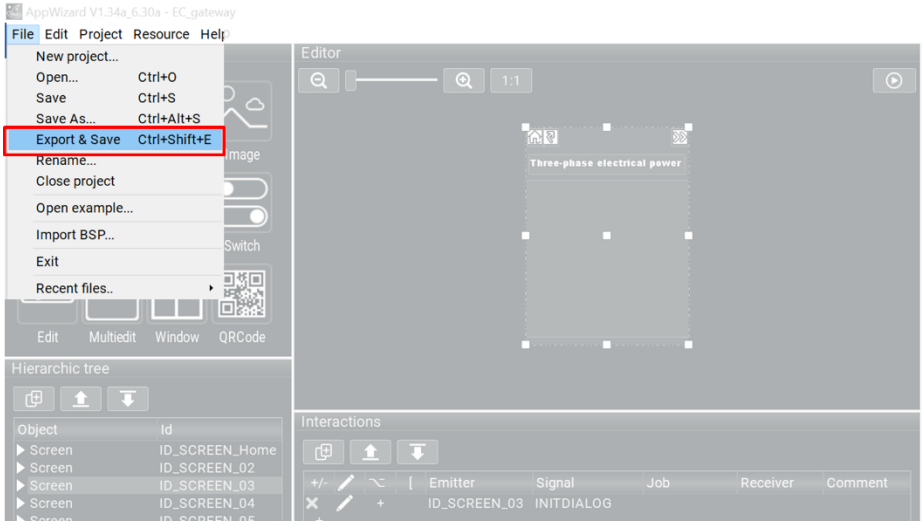


Figure 2-115 Export the Current Project

3. Open the file *ID_SCREEN_03_Slots.c* in location *\Source\CustomCode* and embed the emWin dialog. After finding the slot function name (ID_SCREEN_03_Init) which is set in Step 1, make the following modifications to the content:

```
void ID_SCREEN_01_Init(APPW_ACTION_ITEM * pAction, WM_HWIN hScreen, WM_MESSAGE * pMsg, int
* pResult) {
  WM_HWIN hWindow;

  GUI_USE_PARA(pAction);
  GUI_USE_PARA(hScreen);
  GUI_USE_PARA(pMsg);
  GUI_USE_PARA(pResult);
```

```
  //assign window ID
  hWindow = WM_GetDialogItem(hScreen, emwin_window);
  CreateDialog(hWindow);
}
```

4. Open the emWin dialog program (*SinWaveDLG.c*) in the same path, modify it according to Figure 2-116 and save.

Figure 2-116 Add the Declaration

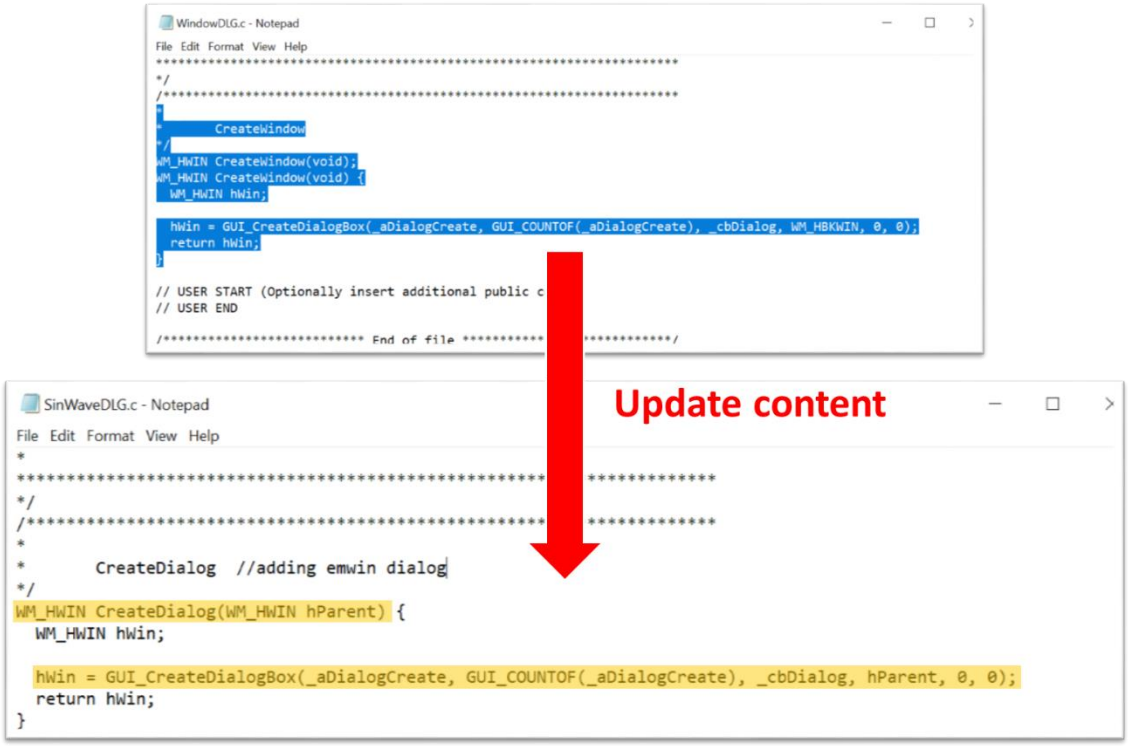


Figure 2-117 Update the Declaration of emWin Dialog

The content of emWin dialog:

```
…

#include "../Generated/Resource.h"
#include "../Generated/ID_SCREEN_03.h"

…

/*********************************************************************
*
*       CreateDialog   // emwin dialog
*/

WM_HWIN CreateDialog(WM_HWIN hParent) {
  WM_HWIN hWin;

  hWin = GUI_CreateDialogBox(_aDialogCreate, GUI_COUNTOF(_aDialogCreate), _cbDialog,
hParent, 0, 0);
  return hWin;
}
…
```

2.3.3.4 Interaction Settings

The completed design is shown in Figure 2-118. Next, users need to establish interactions between objects so that corresponding behaviors will be displayed when operating the objects. (**Note:** The effect of the emWin dialog cannot be previewed in AppWizard, it needs to be deployed to the evaluation board for confirmation)

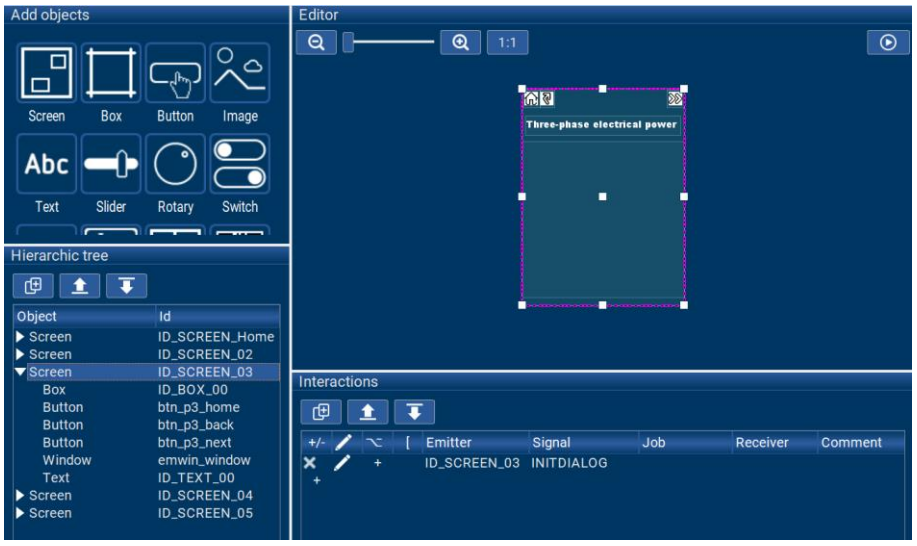


Figure 2-118 Completed Appearance of Screen 3

1. Set the screen switch by the Button object so that clicking the button can switch back to the homepage, previous page, and next page (btn_p3_home, btn_p3_back, btn_p3_next).

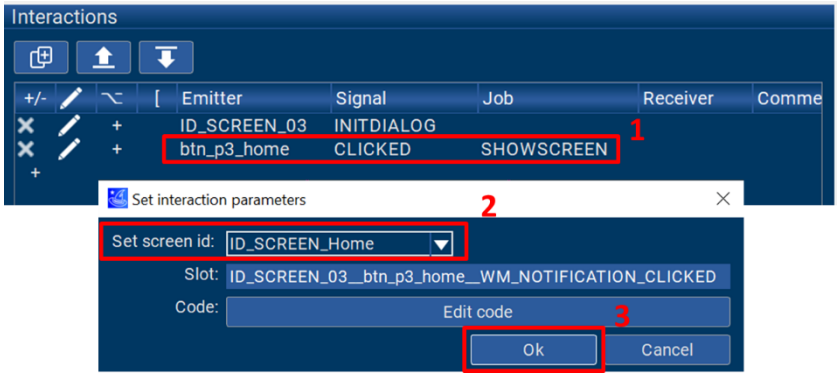


Figure 2-119 Button Settings of Screen Switching (btn_p3_home)

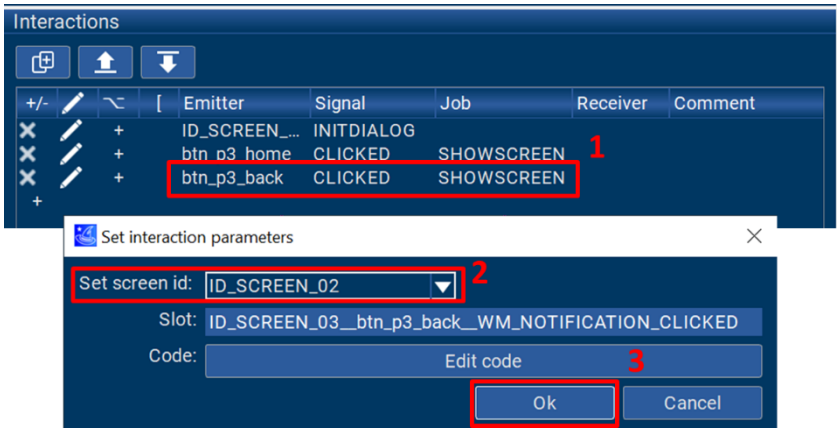


Figure 2-120 Button Settings of Screen Switching (btn_p3_back)

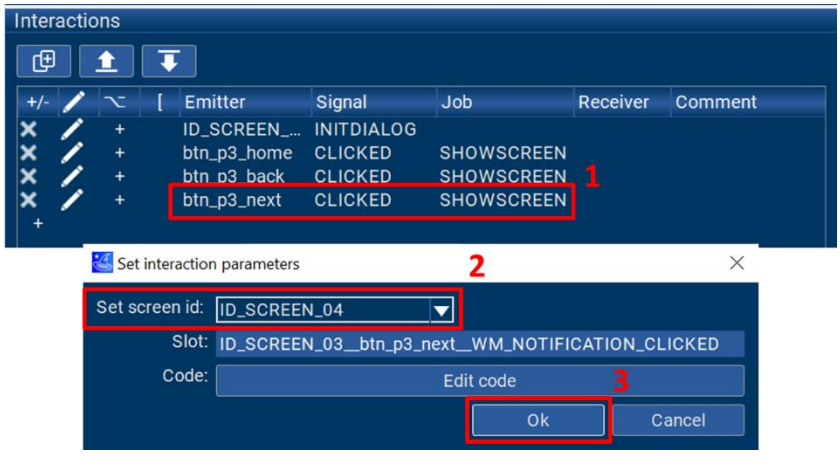


Figure 2-121 Button Settings of Screen Switching (btn_p3_next)

2.3.4 Design of Electricity Display (Screen 4)

Bar charts are one of the most common ways gateways measure or present electricity values. This page is also combined with the animation display function of AppWizard to make the page's content more rich and vivid, and users can go to the website by QR codes directly. (It can be used to display detailed report of power consumption, data analysis, and other information pages) The widgets used in this screen are shown in Table 2-6.

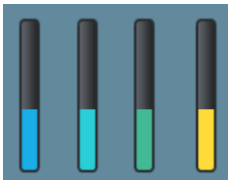
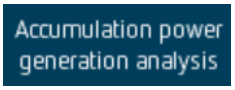
New Energy Gateway Description	Application GUI	Widget Type
Background (240x320)		Box
Background of animation (200x207)		Box
Cumulative power consumption bar chart		Progbar
QR code		QR code
Text of title		Text
Description of bar chart		
Description of QR code		
Button of home page		Button
Button of previous page		
Button of next page		

Table 2-6 Widgets of Electricity Display

2.3.4.1 Widgets Configuration of GUI

1. Choose ID_SCREEN_04 to start editing screen 4, click on "Button" in "Add objects" to create screen change buttons for the previous page (btn_p4_back), the next page (btn_p4_next), and the home page (btn_p4_home), then set the images *arrow2_20x20_new.png*, *arrow1_20x20_new.png*, and *home_20x20_new.png* as their bitmap.

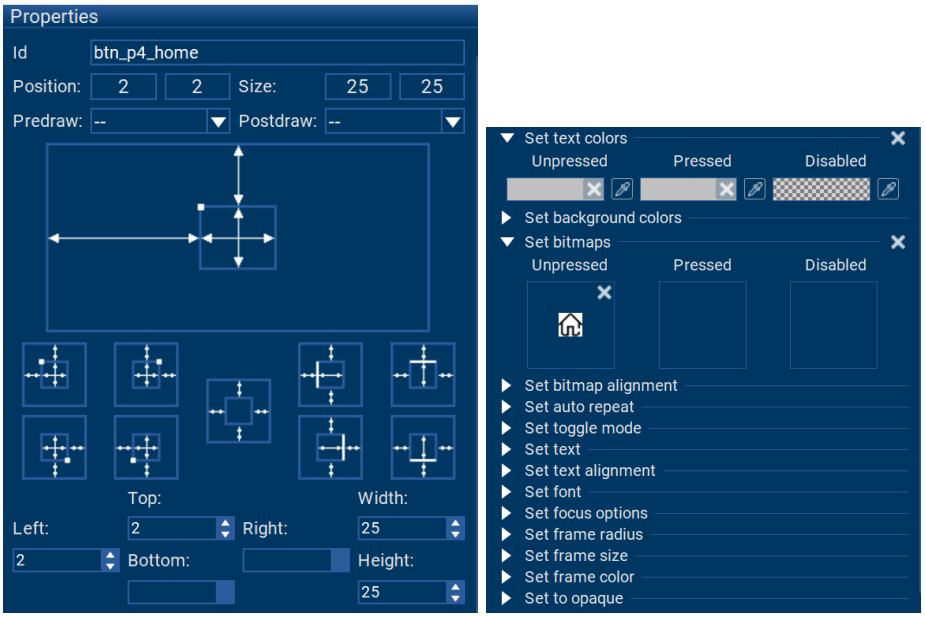


Figure 2-122 Set Properties of btn_p4_home

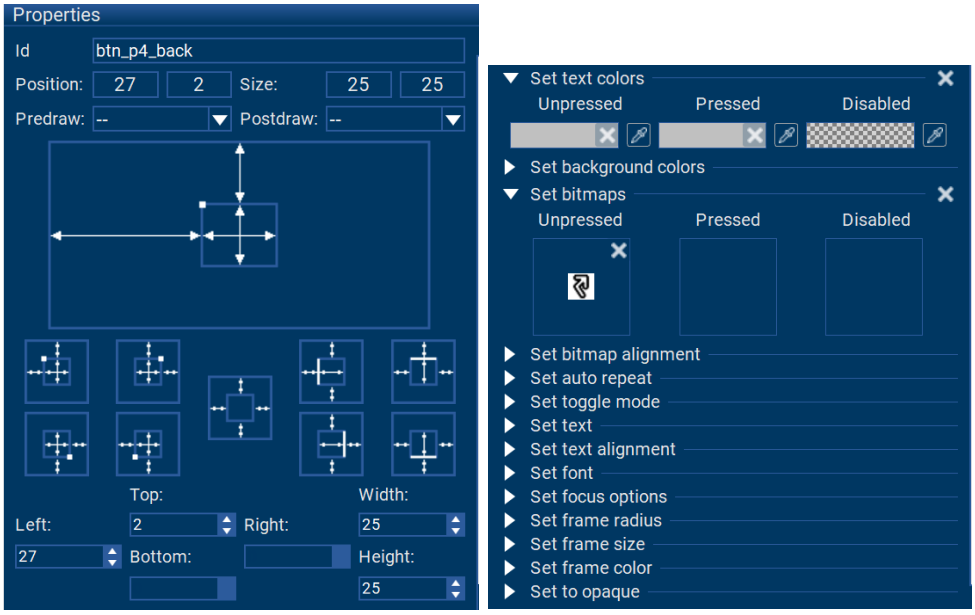


Figure 2-123 Set Properties of btn_p4_back

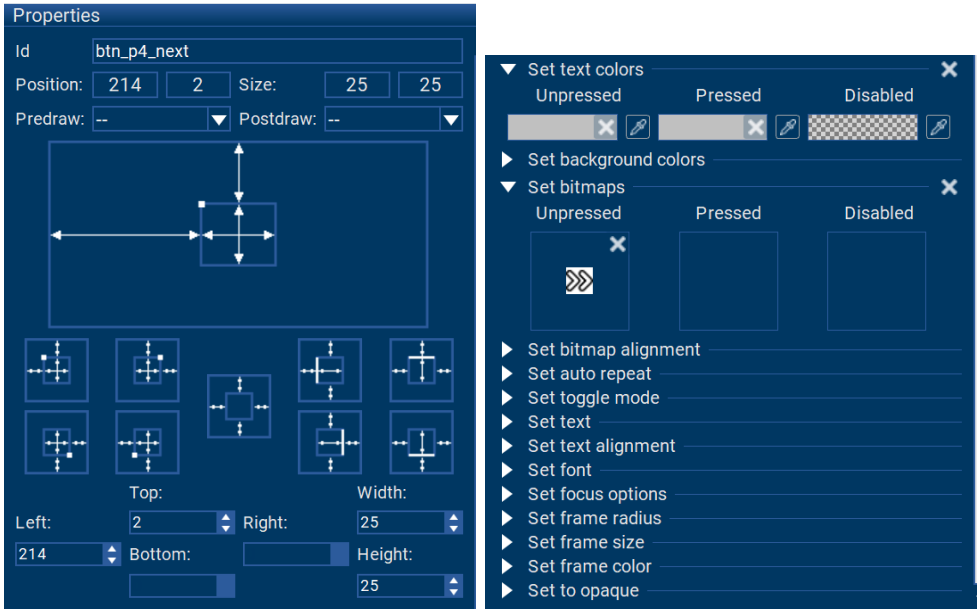


Figure 2-124 Set Properties of btn_p4_next

2. Add a BOX object (ID_BOX_01) by "Add object" as the background of the animation area, and set the properties as Figure 2-125.

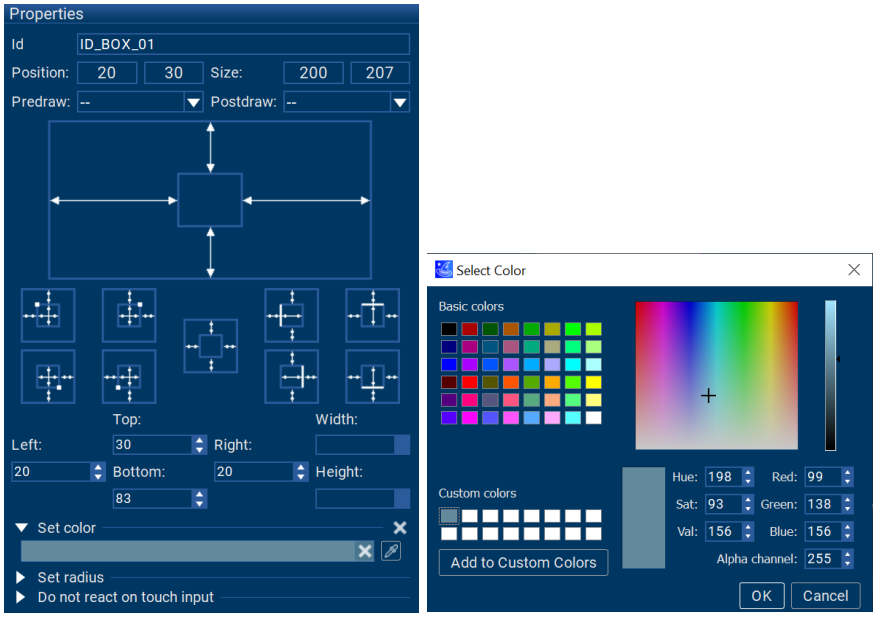


Figure 2-125 Set Properties of ID_BOX_01

3. Add four Progbar objects (PROGBAR_00~PROGBAR_03) by "Add object" to display different levels of power, and set the properties according to Figure 2-126 to Figure 2-129. First, turn on the vertical mode of Progbar, and use the images *progbar_LightBlue.png*, *progbar_LightGreen.png*, *progbar_Green.png* and *progbar_Yellow.png* as the bitmap bottom of Progbar respectively, and select the image *DARK_Progbar_Tile_V_Gray_16x1.png* as the bitmap top.

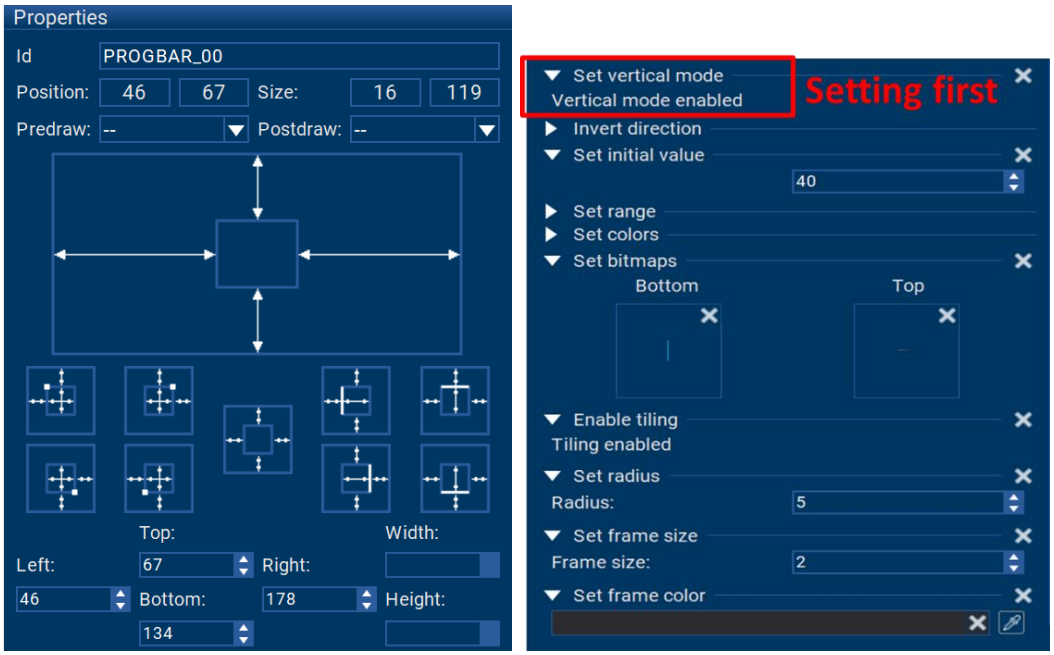


Figure 2-126 Set Properties of PROGB AR_00

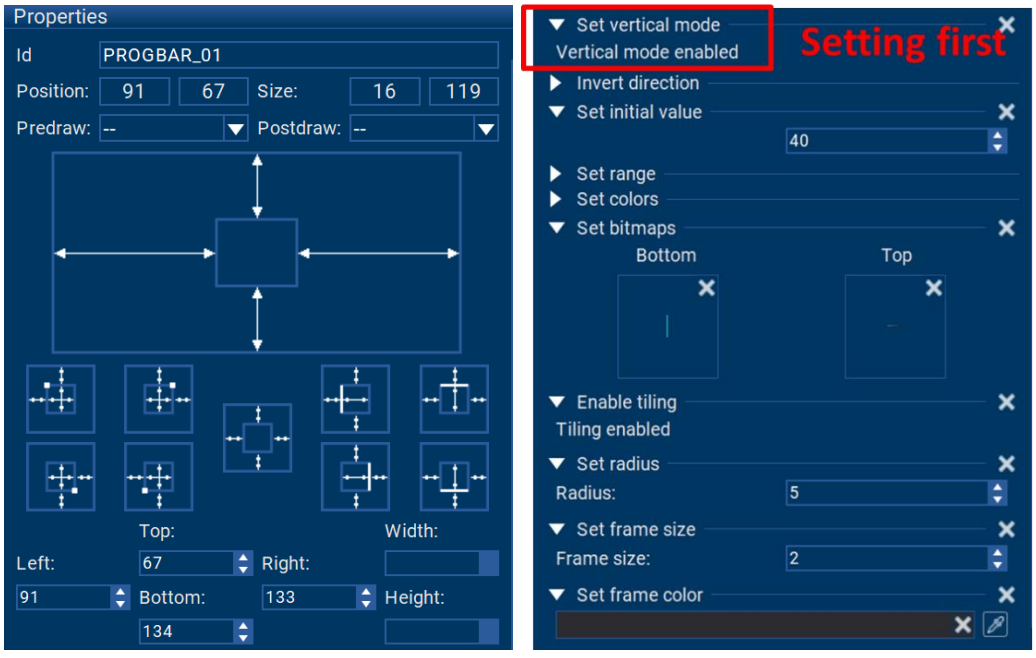


Figure 2-127 Set Properties of PROGBAR_01

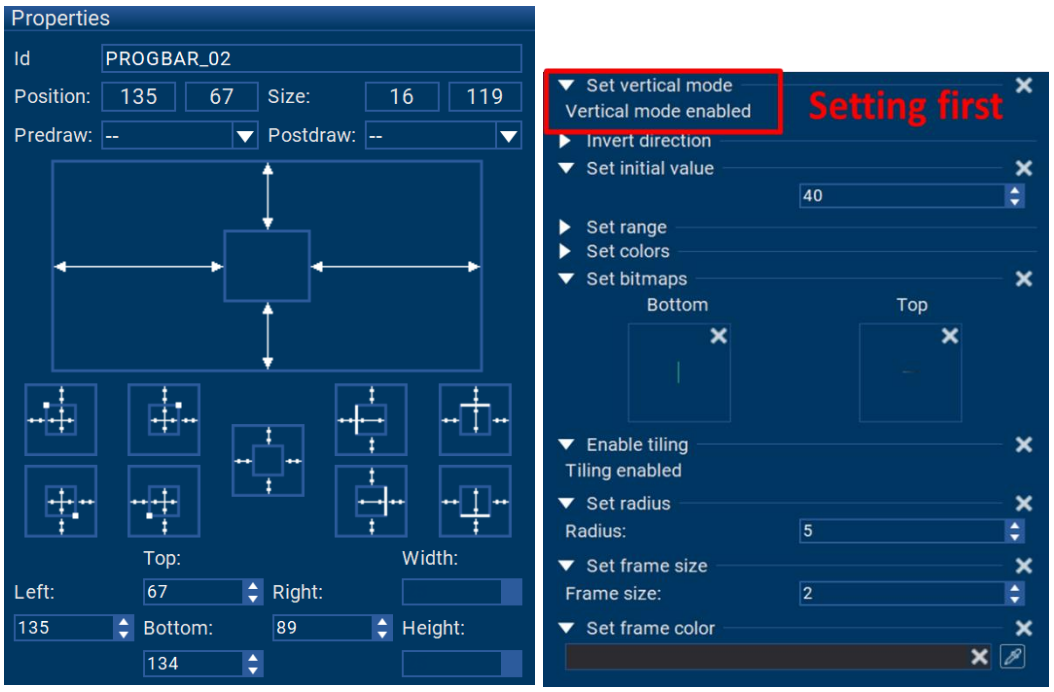


Figure 2-128 Set Properties of PROGBAR_02

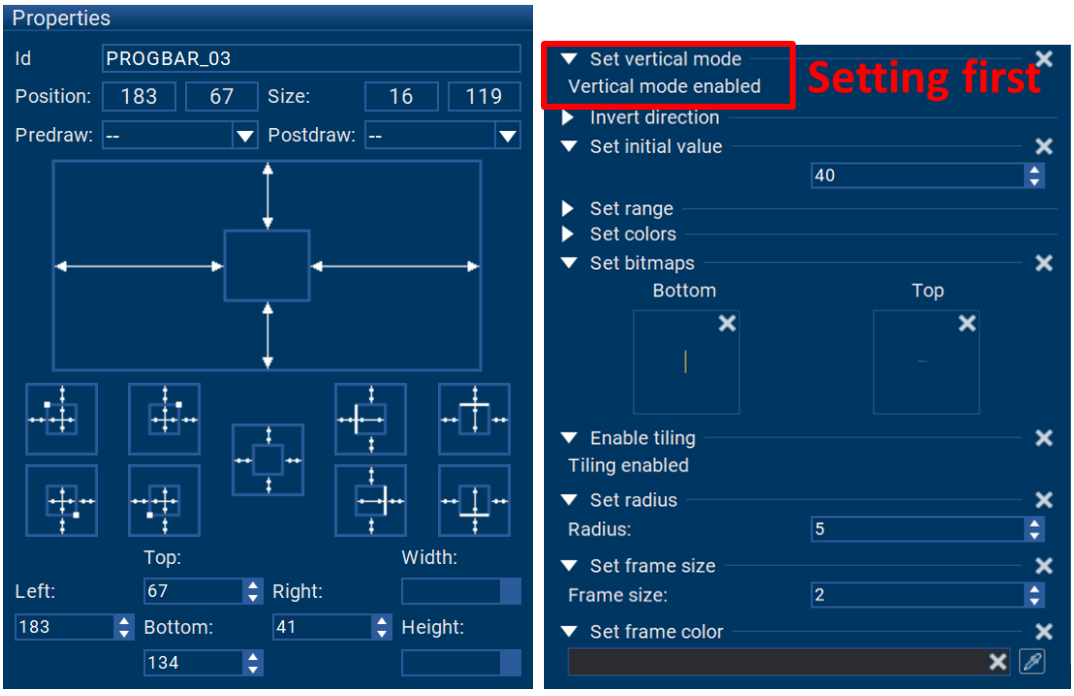


Figure 2-129 Set Properties of PROGBAR_03

4. Add two TEXT objects as title (ID_TEXT_00) and footer text (ID_TEXT_01), and set the displayed text as "Monitoring Report" and "Accumulation power generation analysis" respectively, then set the properties according to Figure 2-132 and Figure 2-133.

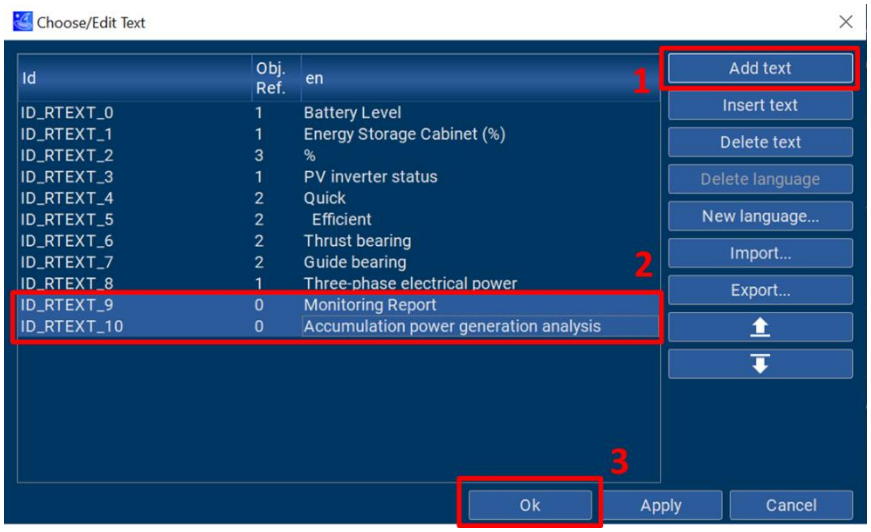


Figure 2-130 Add the Display Text

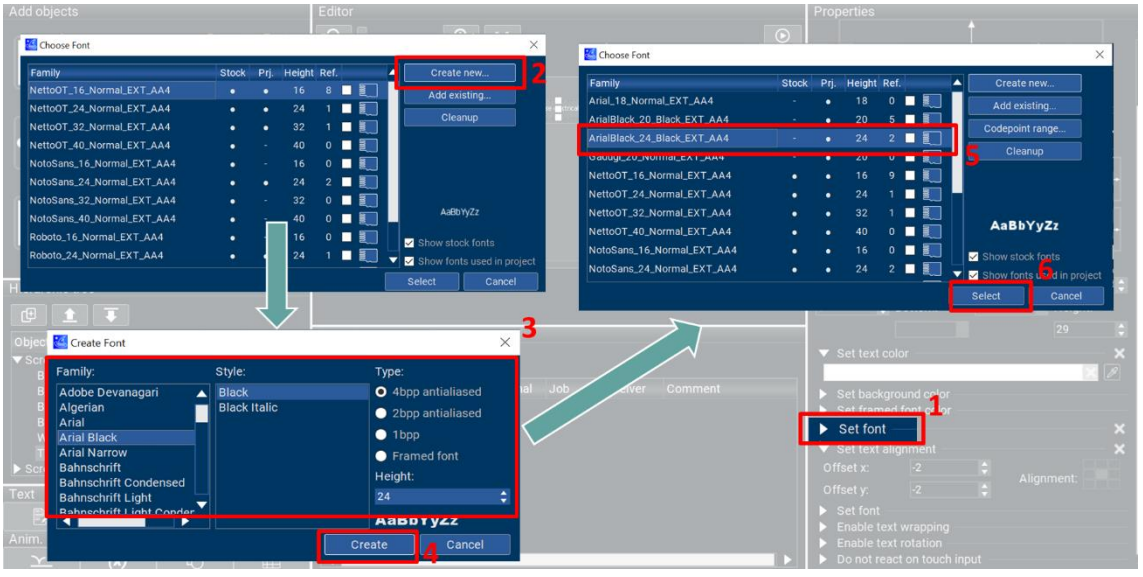


Figure 2-131 Set the Custom Font for ID_TEXT_00

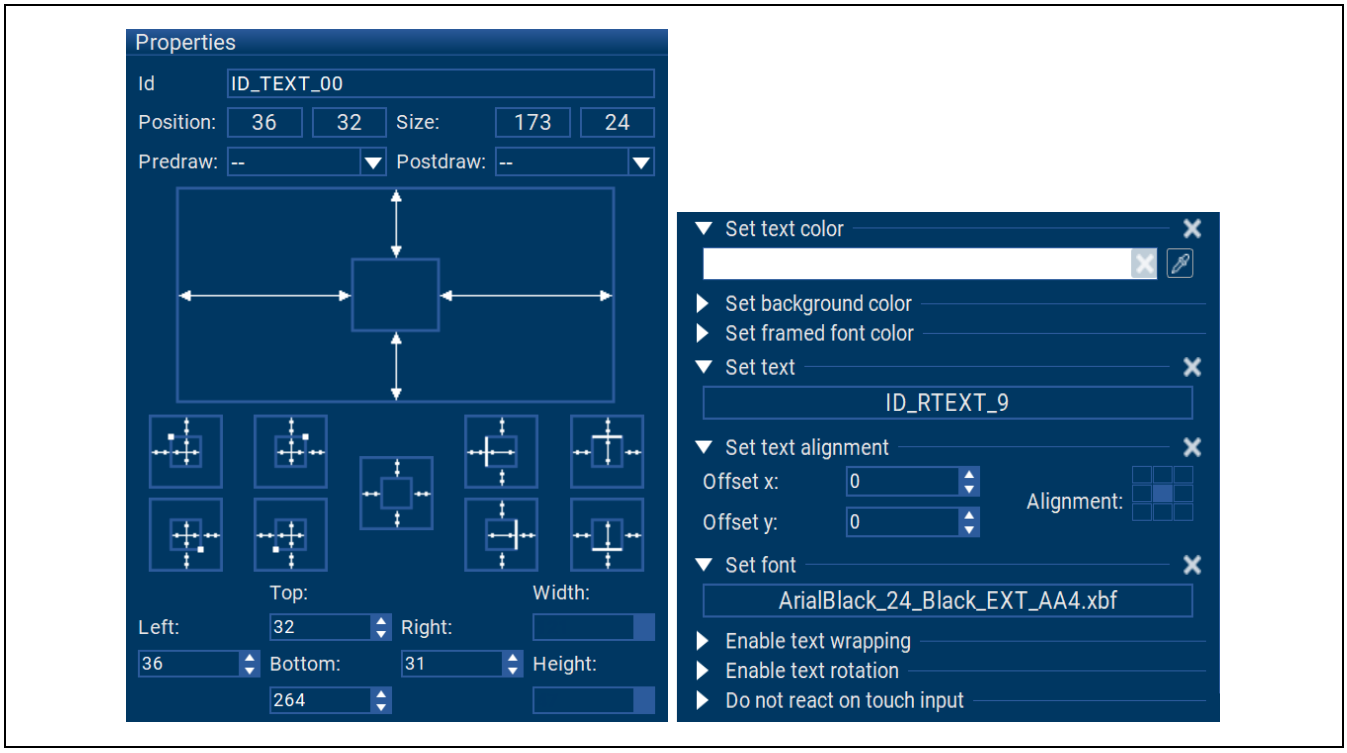


Figure 2-132 Set Properties of ID_TEXT_00

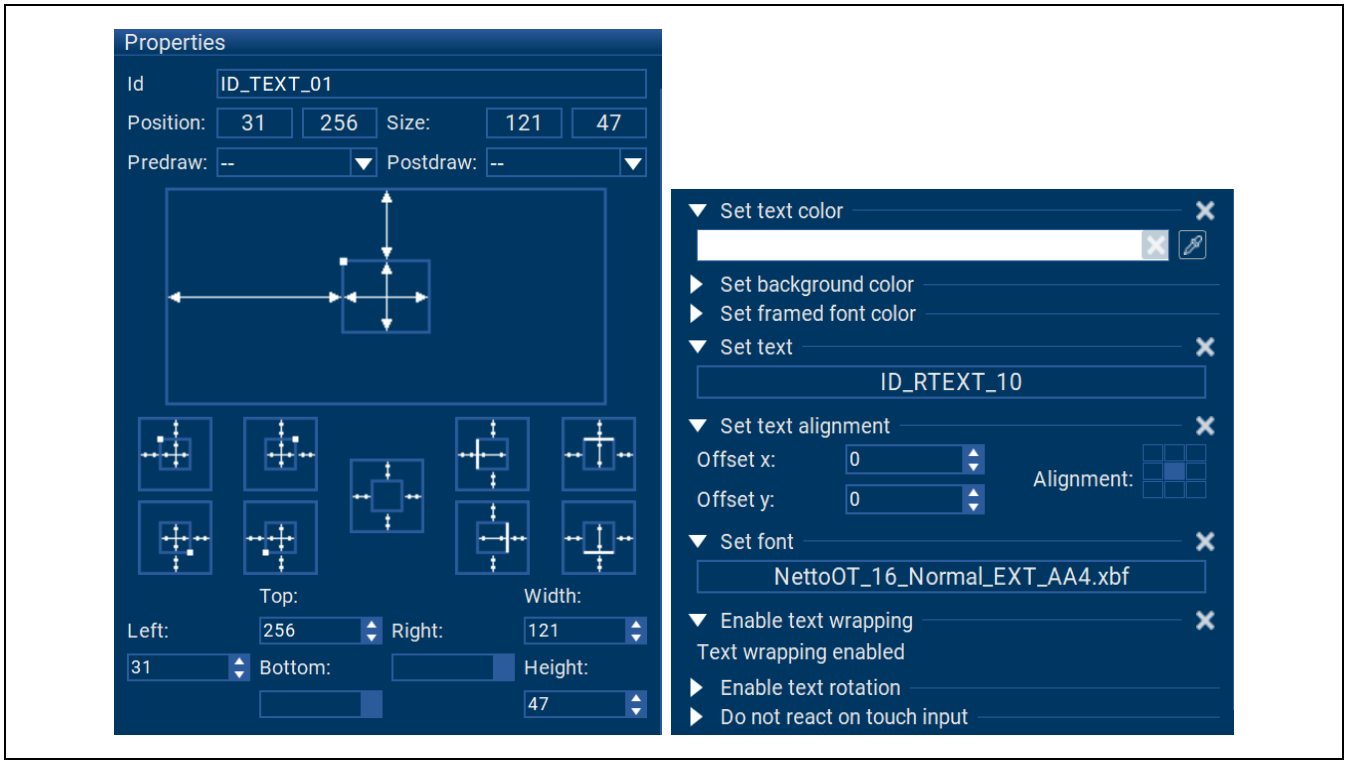


Figure 2-133 Set Properties of ID_TEXT_01

5. To set the captions for each bar chart, add four TEXT objects (ID_TEXT_02~ID_TEXT_04) and set the display text as "Instant Power (W)", "Daily (KW)", "Monthly (MW)" and "Cumulative power (GW)" respectively. Set the properties according to Figure 2-135 to Figure 2-138.

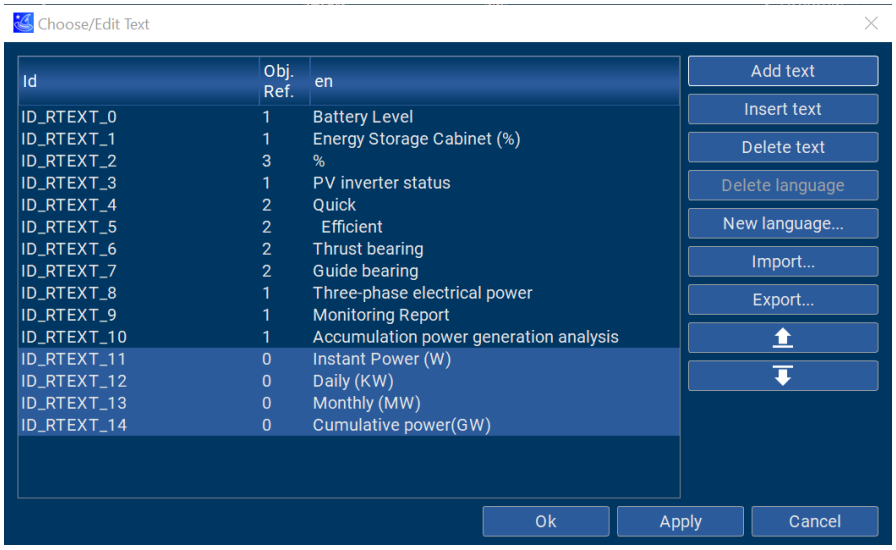


Figure 2-134 Add the Display Text (Progbar Caption)

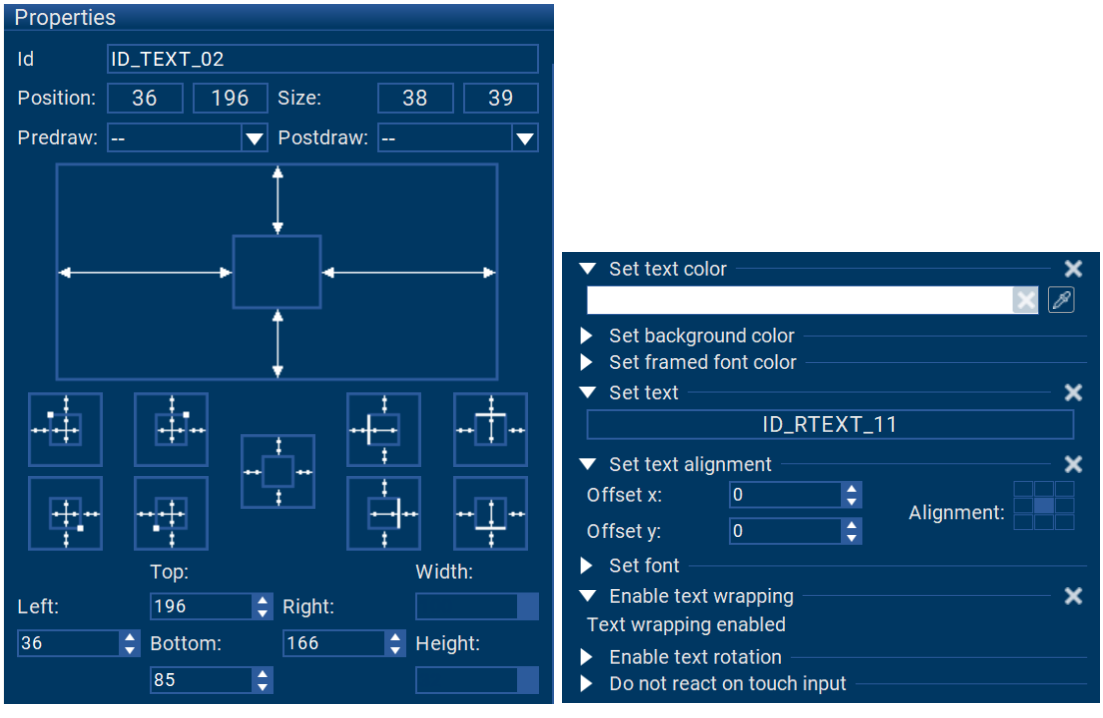


Figure 2-135 Set Properties of ID_TEXT_02

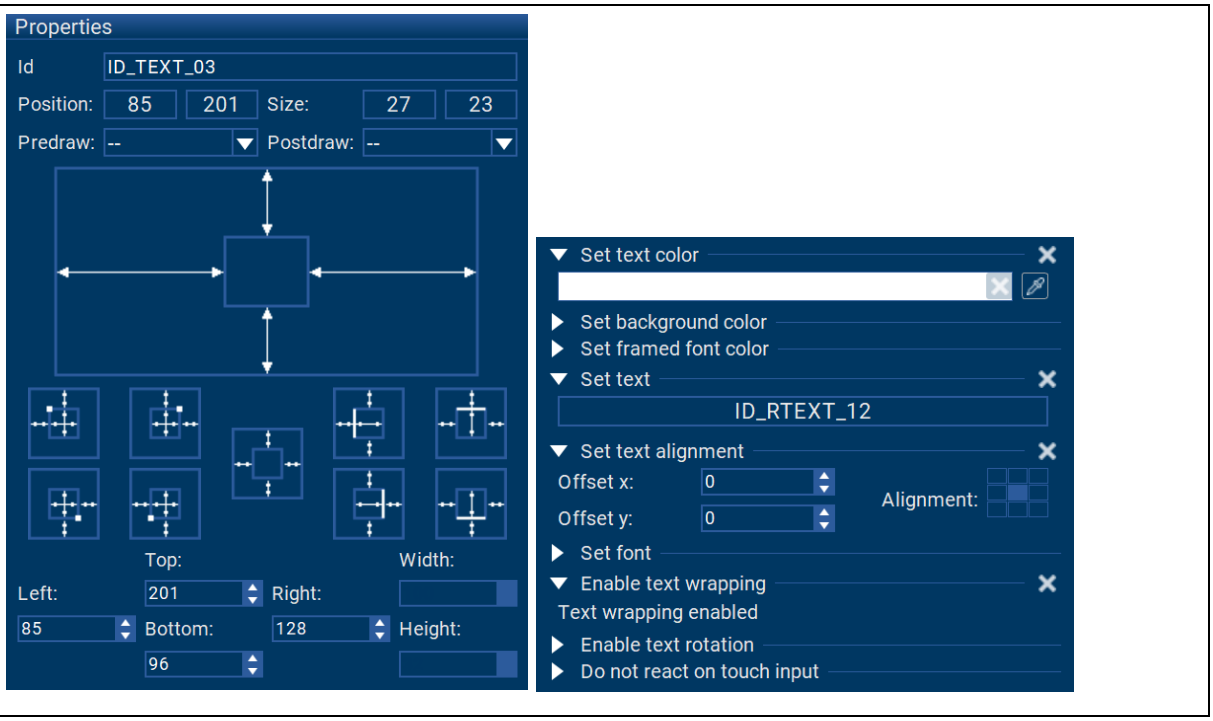


Figure 2-136 Set Properties of ID_TEXT_03

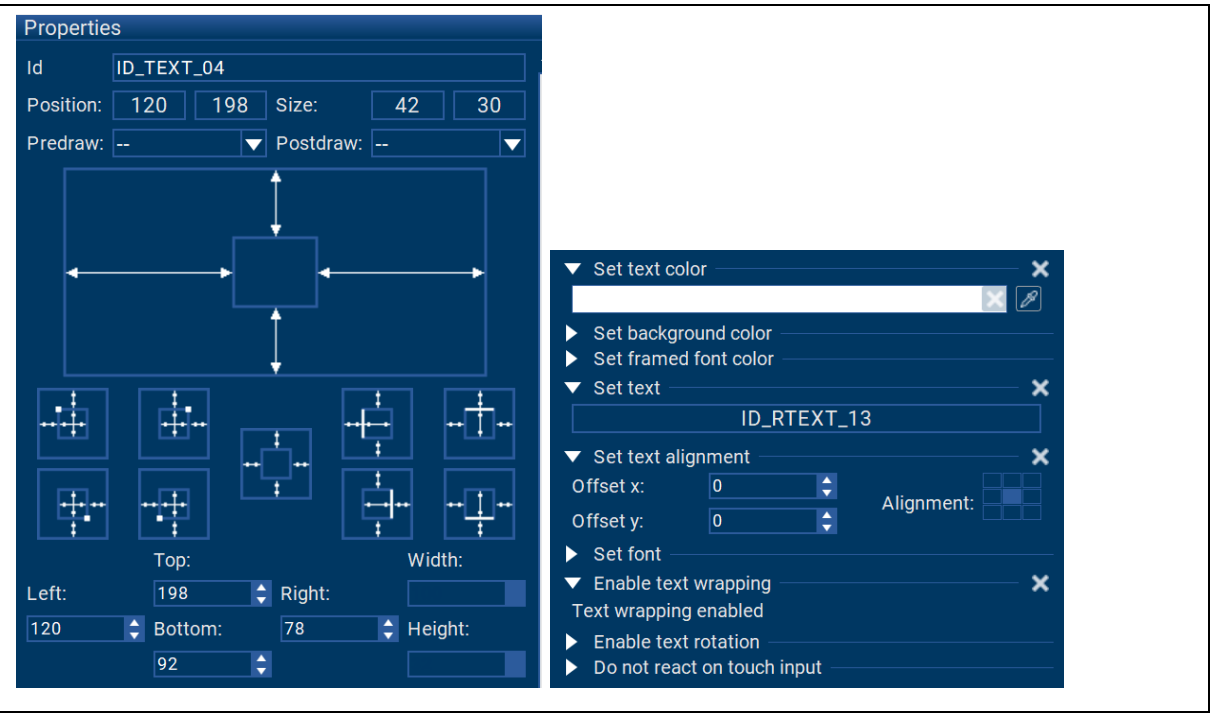


Figure 2-137 Set Properties of ID_TEXT_04

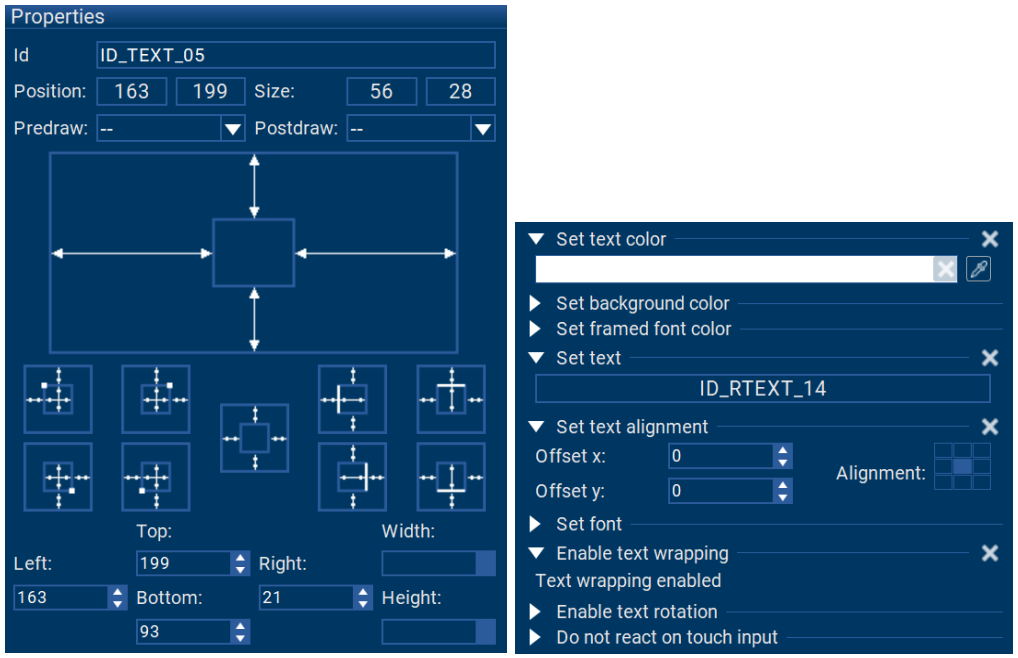


Figure 2-138 Set Properties of ID_TEXT_05

6. Add a QR code object (ID_QRCODE) for users to be led to a specific web page by scanning. Set the display text as URL (https://www.nuvoton.com/products/microprocessors/arm9-mpus/) and set the properties according to Figure 2-140 and Figure 2-141.

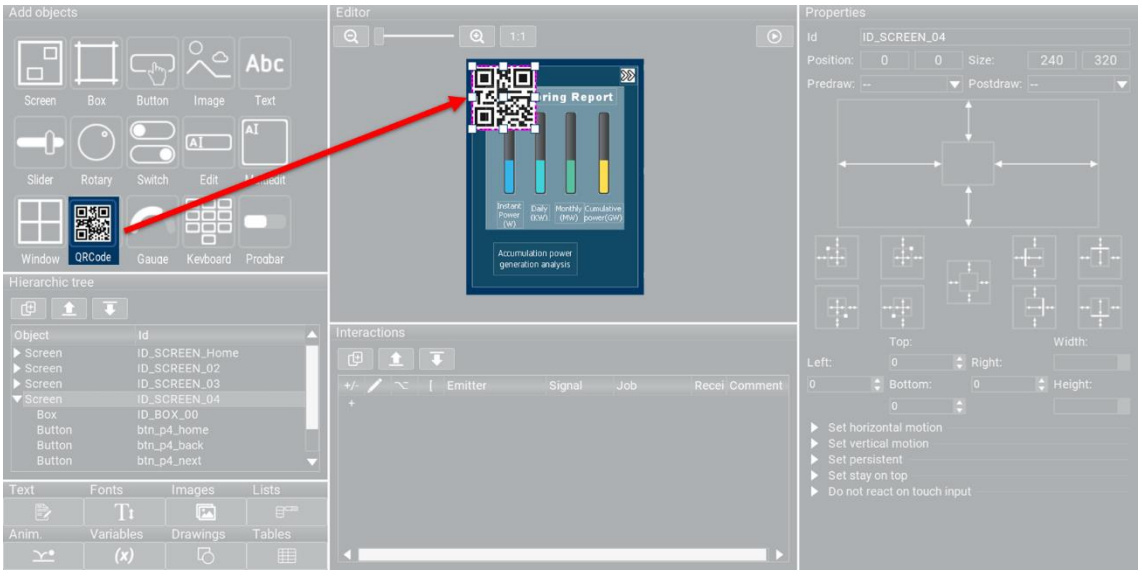


Figure 2-139 Add A QR Code Object

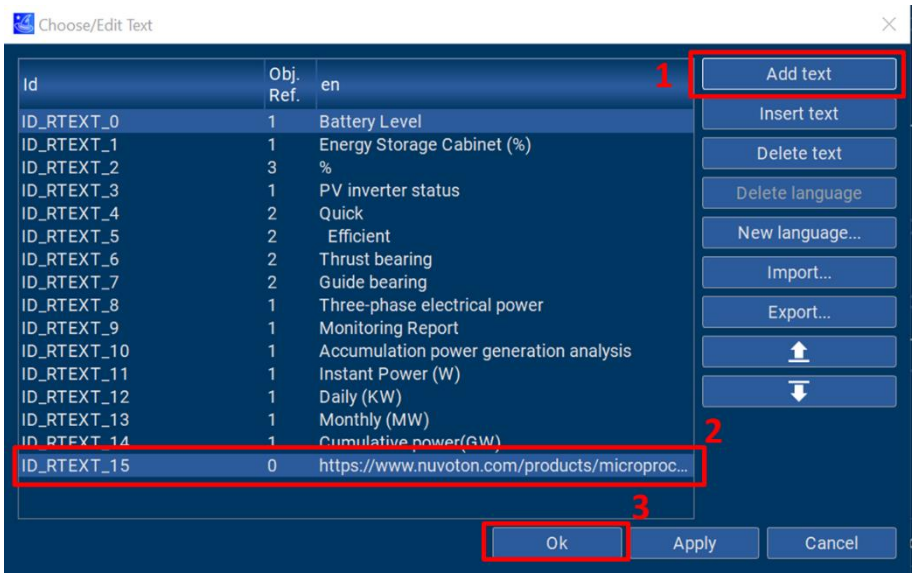


Figure 2-140 Add a URL as Content in Edit Text

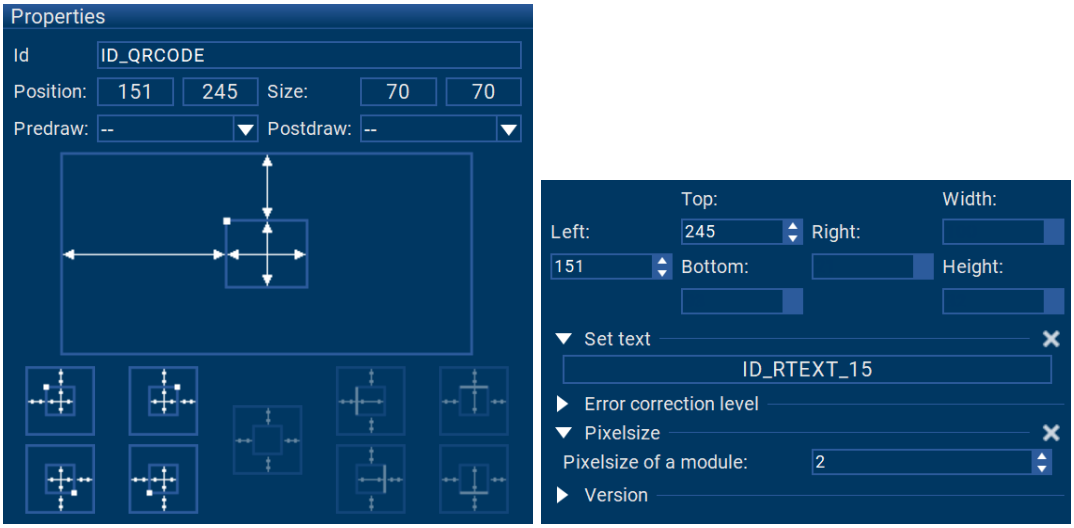


Figure 2-141 Set Properties of ID_QRCODE

2.3.4.2 Interaction Settings

The completed design is shown in Figure 2-142. Next, users need to establish interactions between objects so that corresponding behaviors will be displayed when operating the objects.

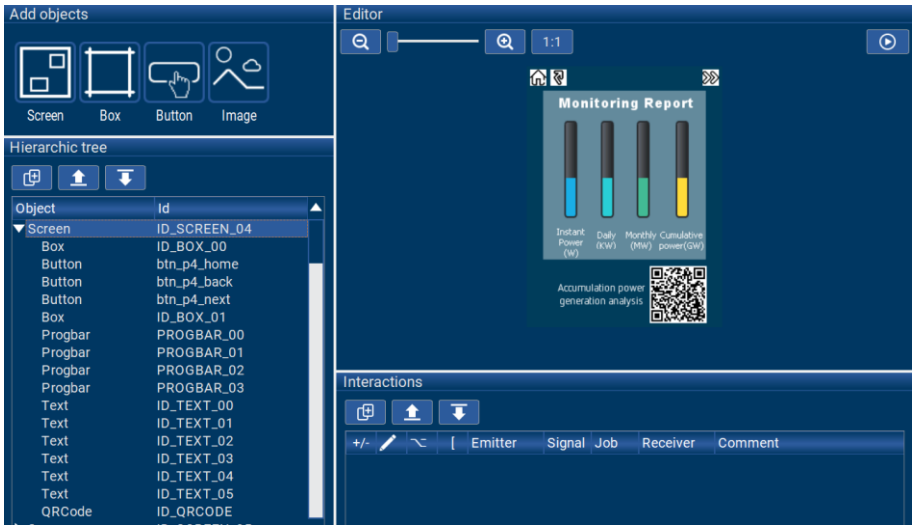


Figure 2-142 Completed Appearance of Screen 4

1. To make the values of the histogram dynamically present, click the Anim button in the lower left corner to add animation (ANIM_PW_LINE), and then set the animation details according to the steps to Figure 2-143 to Figure 2-151.

   1) After adding an animation, modify the ID as ANIM_PW_LINE and set the length(Period) of the animation to 2000 (ms).

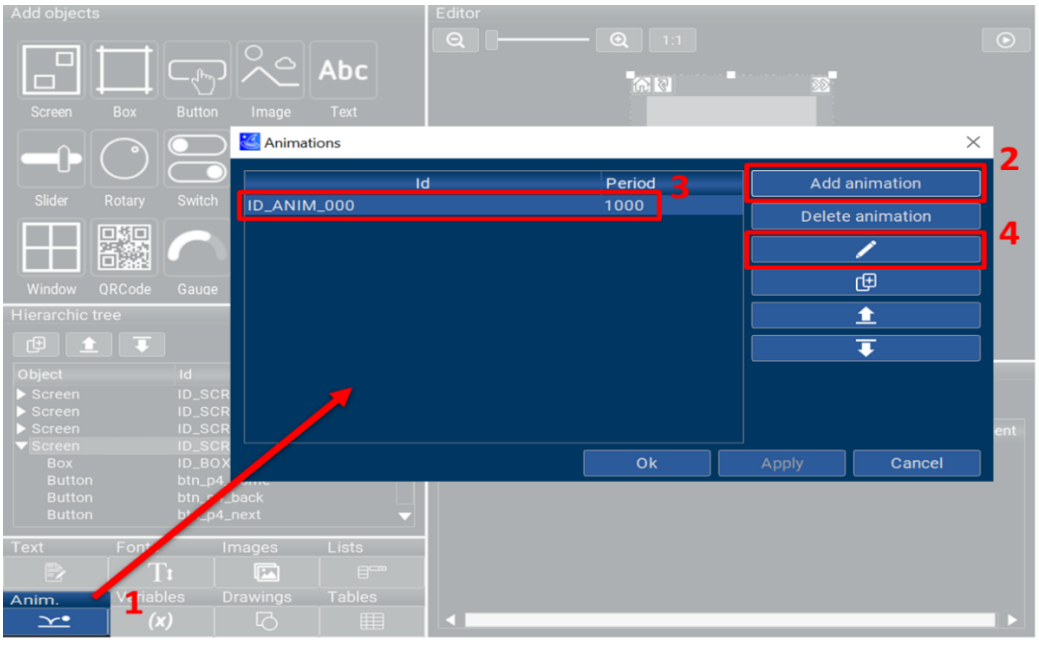


Figure 2-143 Add Animation

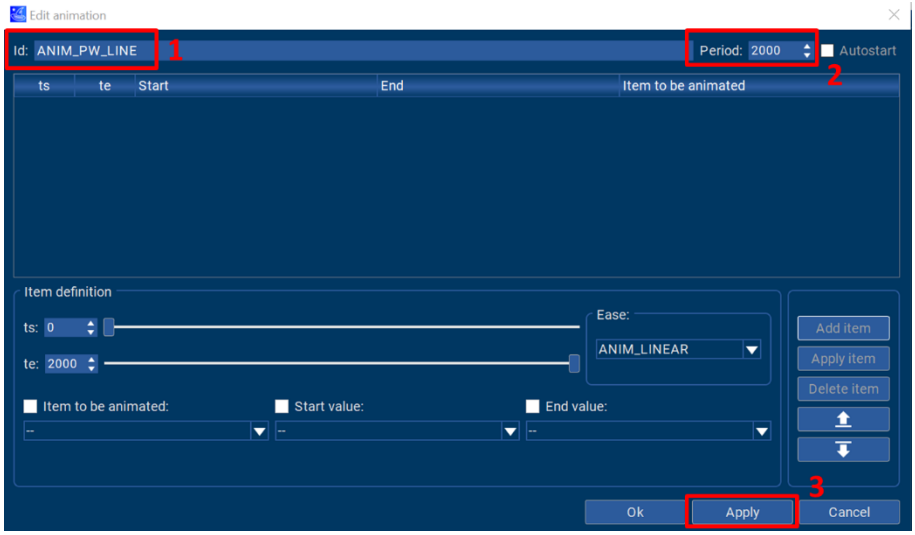


Figure 2-144 Set ID and Time-length of Animation

2) Set the start and end time of the animation through "ts" and "te".

3) Set the action object ID and change the action's starting value, end value, and Ease (ANIM_ACCELDECEL).

4) Click "Add item" to add this action to the ANIM_PW_LINE.

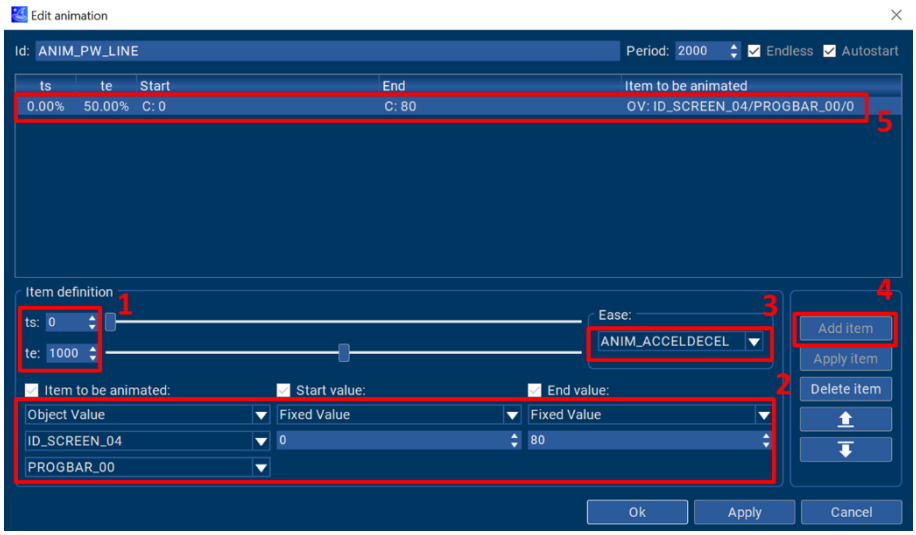


Figure 2-145 Add the Action of PROGBAR_00

5) Same as steps 1) to 4), set the actions of other Progbar objects (PROGBAR_01~ PROGBAR_03).

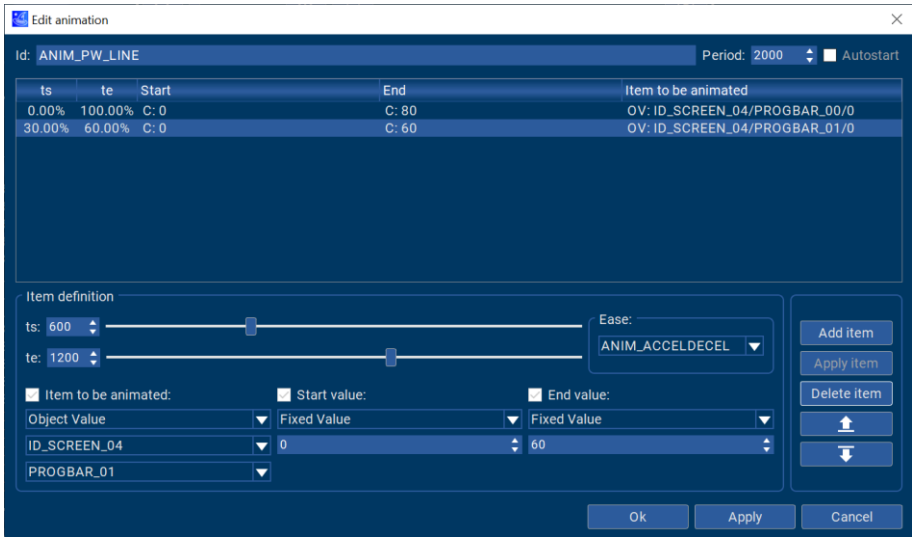


Figure 2-146 Add the Action of PROGBAR_01

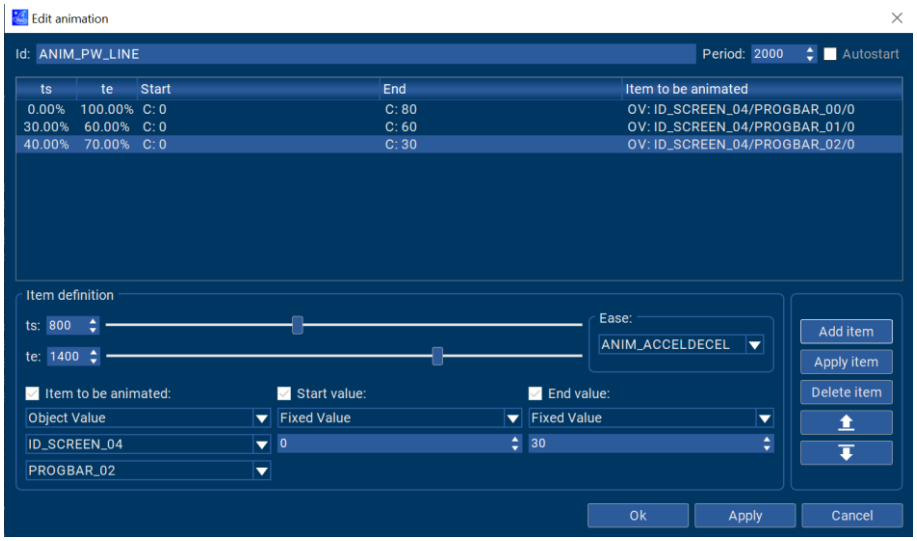


Figure 2-147 Add the Action of PROGBAR_02

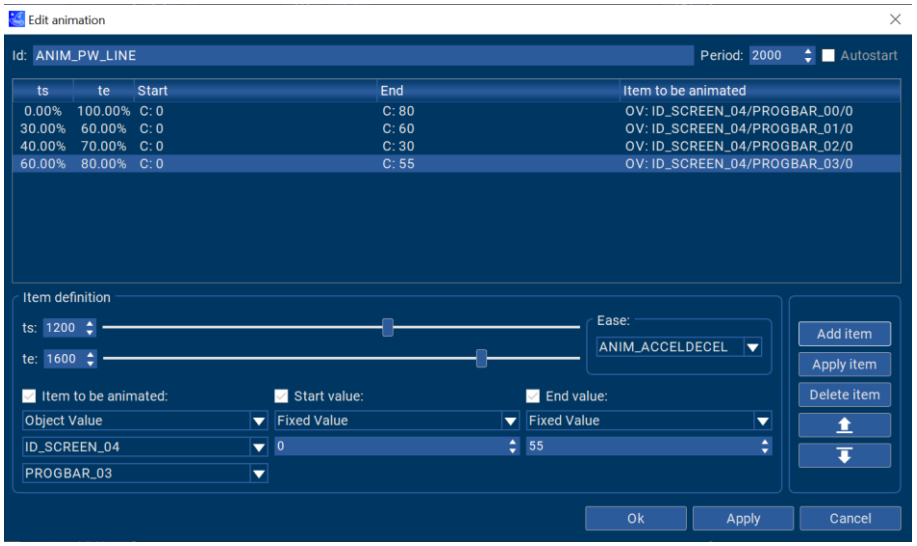


Figure 2-148 Add the Action of PROGBAR_03

6) After completing all action settings, tick "Autostart" and "Endless" to make the animation start automatically and loop.

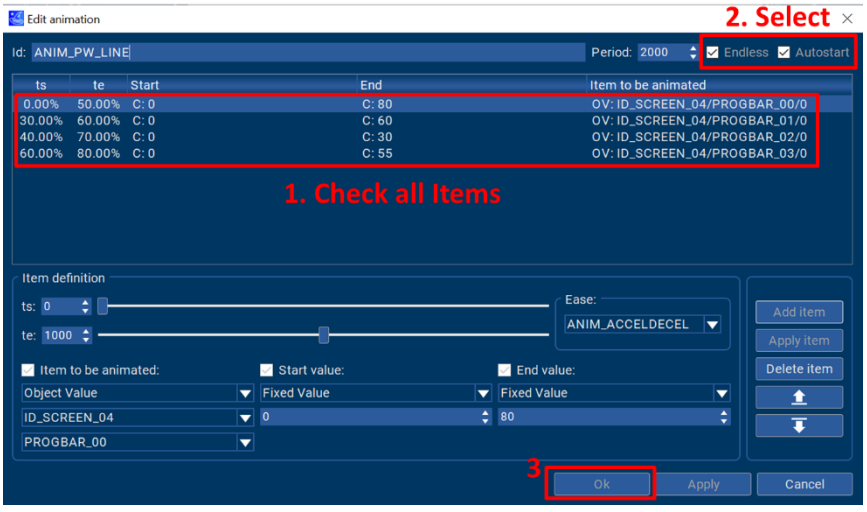


Figure 2-149 Set "AutoStart" and “Endless” of the Animation

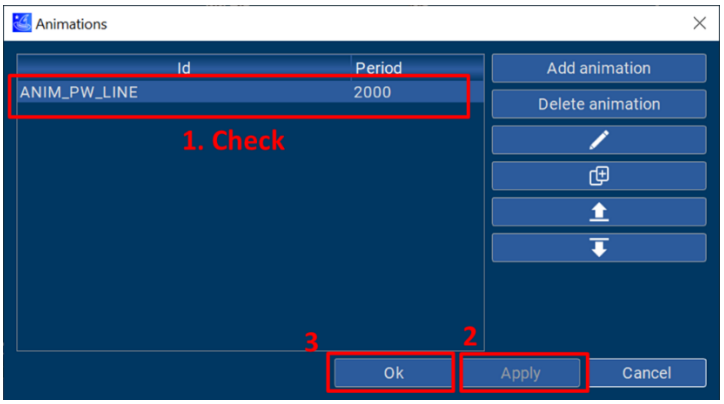


Figure 2-150 Save the Settings of ANIM_PW_LINE

7) Add an item in the interactions field: Emitter is "ID_SCREEN_04", Signal is "INITDIALOG", and Job is "ANIMCREATE", then set the animation id to ANIM_PW_LINE.

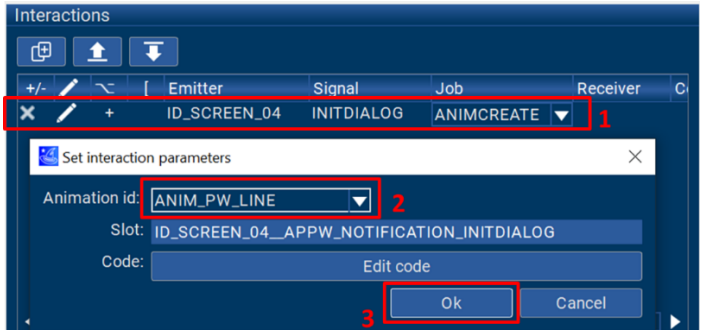


Figure 2-151 Set the Display of the Animation

2. Set the screen-switching function by the Button objects ((btn_p4_home, btn_p4_back, btn_p4_next)) so that clicking the button can switch to the home page, the previous page, and the next page.

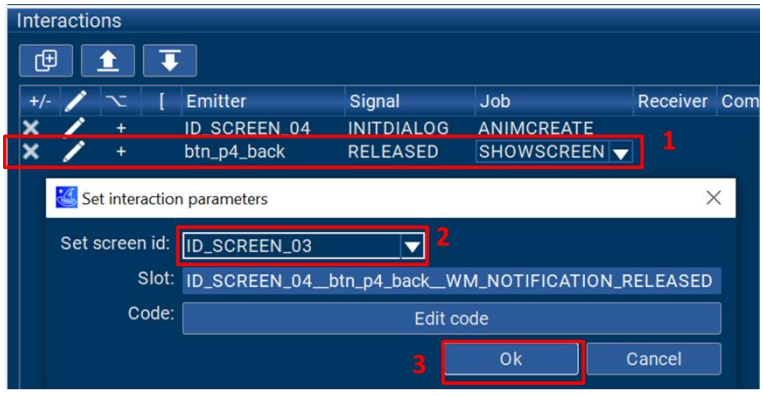


Figure 2-152 Interaction Settings of Screen Switching (btn_p4_back)

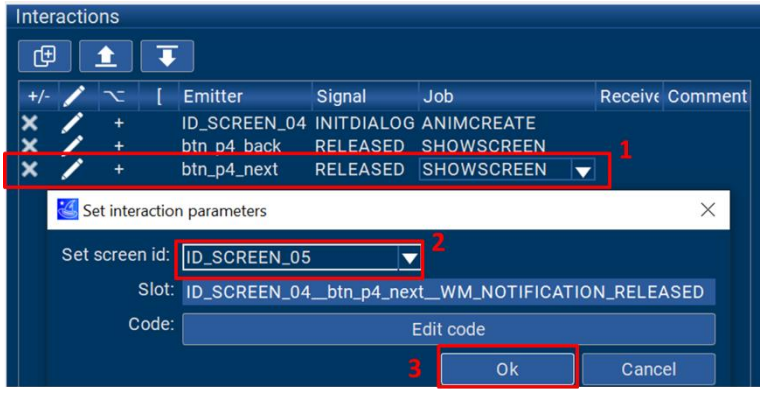


Figure 2-153 Interaction Settings of Screen Switching (btn_p4_next)

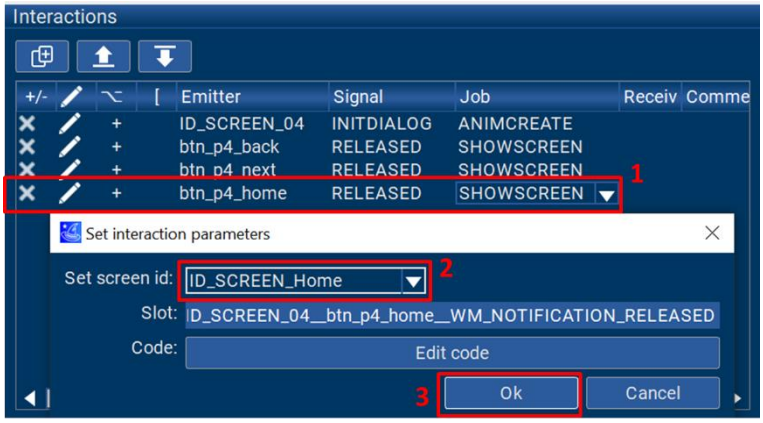
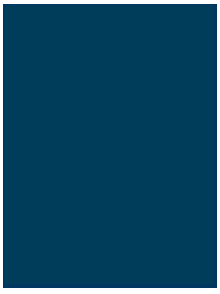


Figure 2-154 Interaction Settings of Screen Switching (btn_p4_home)

2.3.5 Design of Conversion Rate Adjustment (Screen 5)

There are many applications such as energy conversion and inverters in gateway products. In addition to adjusting the conversion rate through the Slider object, this design also sets the Switch object as a safety switch (locking the adjustment rate) to reduce the risk of accidental operation by personnel. Damage caused by accidental contact. The objects used in this design are shown in Table 2-7.

New Energy Gateway Description	Application GUI	Widget Type
Background (240x320)		Box

Background of sliders (150x254)		Box
Title		Text
Percent display		
Conversion range adjustment slot		Slider
Lock-switch		Switch
Upload button		Button
Button of home page		
Button of previous page		

Table 2-7 Widgets of Conversion Rate Adjustment

2.3.5.1 Widgets Configuration of GUI

1. Choose ID_SCREEN_05 to start editing screen 5, and click on "Box" in "Add objects" to create a second box as the background of sliders, then set the properties according to Figure 2-155.

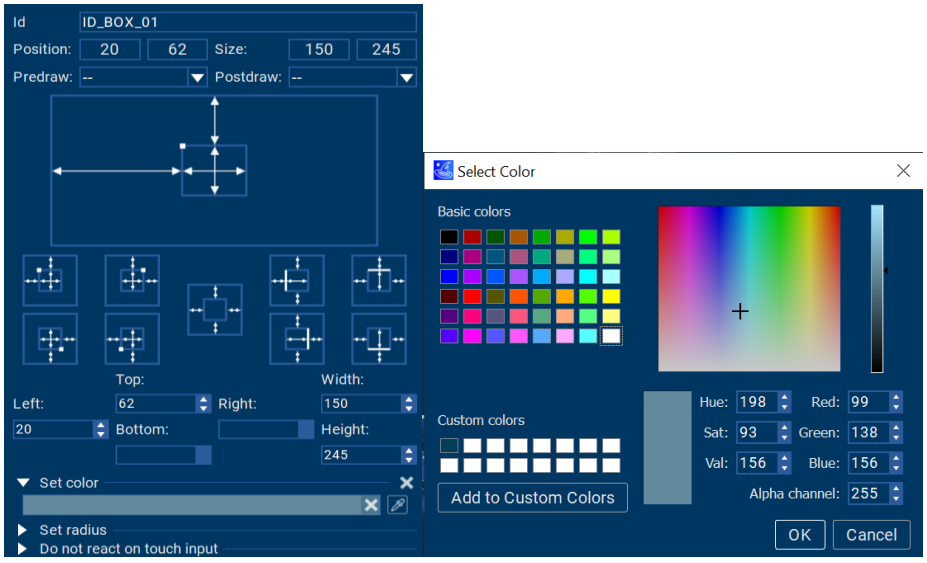


Figure 2-155 Set Properties of ID_BOX_01

2. Click on "Text" objects in "Add objects" to create five Text objects and display them as the title (ID_TEXT_00, the displayed text is Amplitude adjustment), two slot values (ID_TEXT_01, ID_TEXT_03) and two percentage symbols (ID_TEXT_02, ID_TEXT_04). Set the properties according to Figure 2-157 to Figure 2-161.

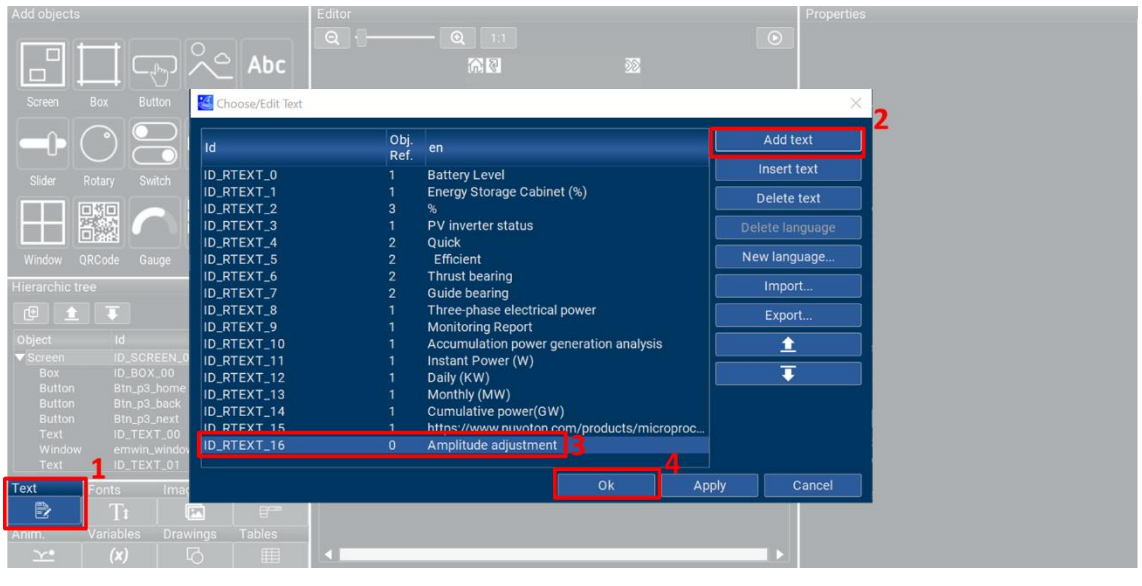


Figure 2-156 Add the display content of ID_TEXT_00

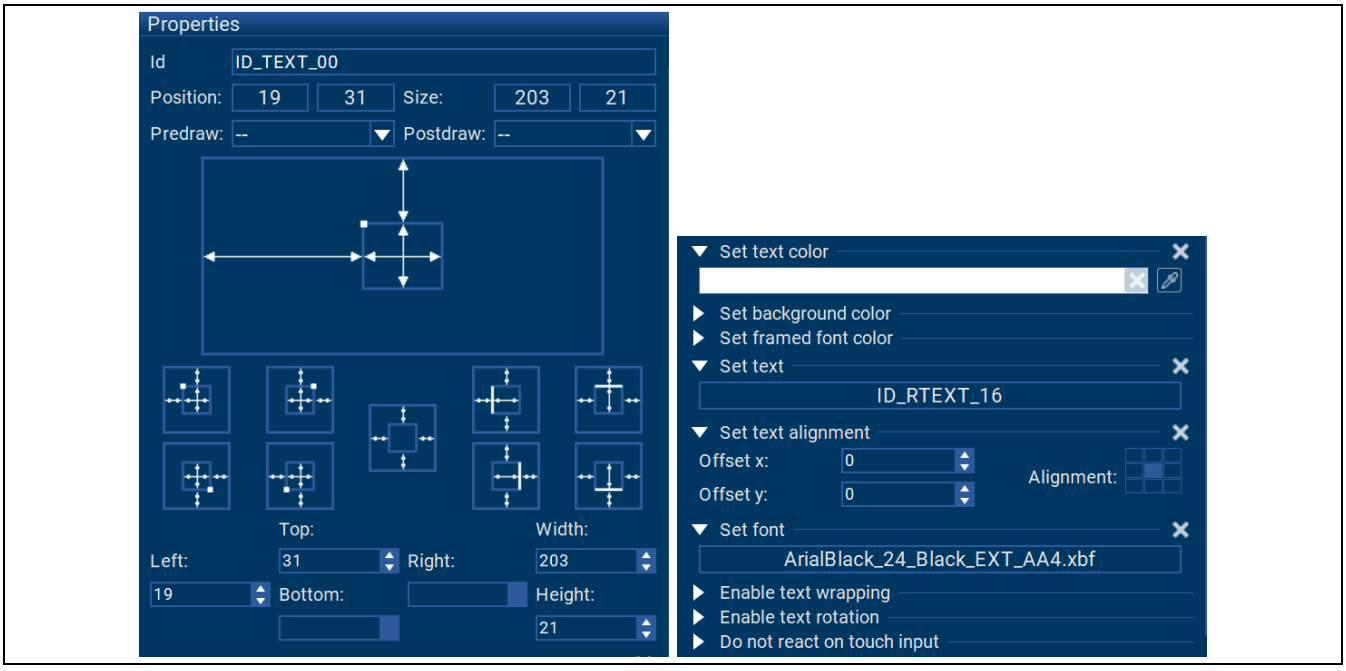


Figure 2-157 Set Properties of ID_TEXT_00

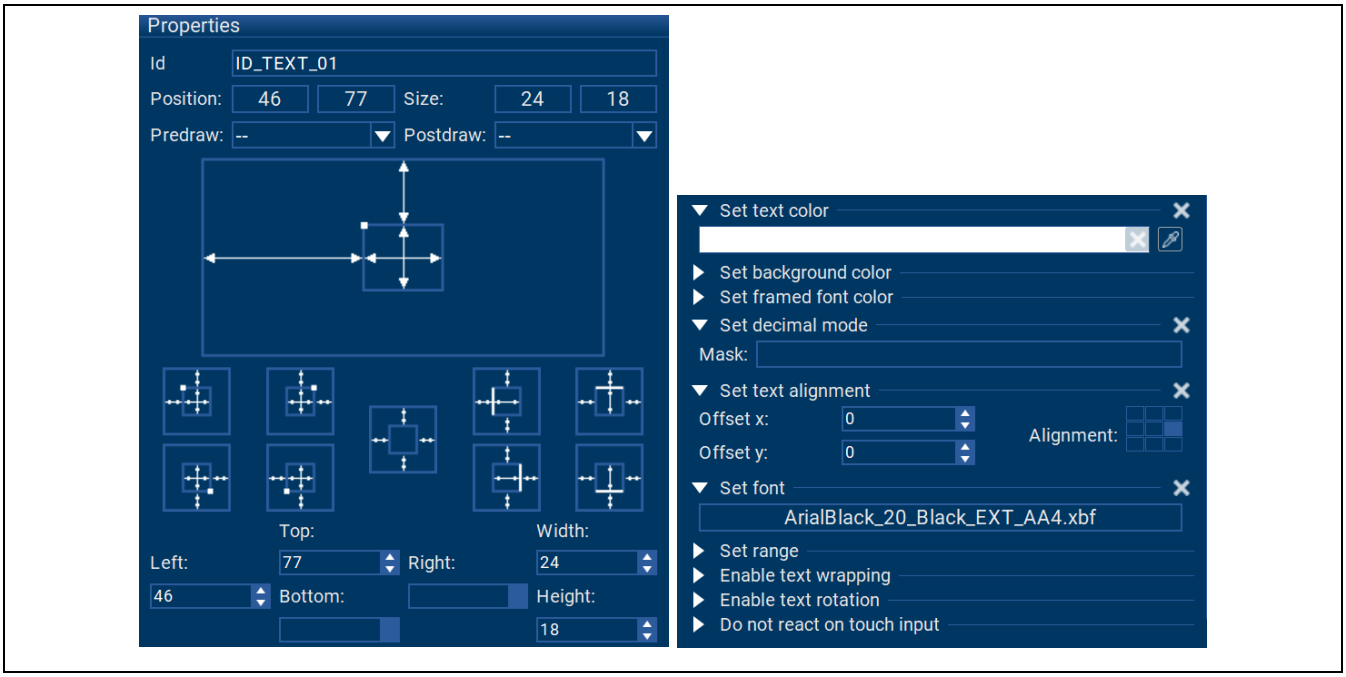


Figure 2-158 Set Properties of ID_TEXT_01

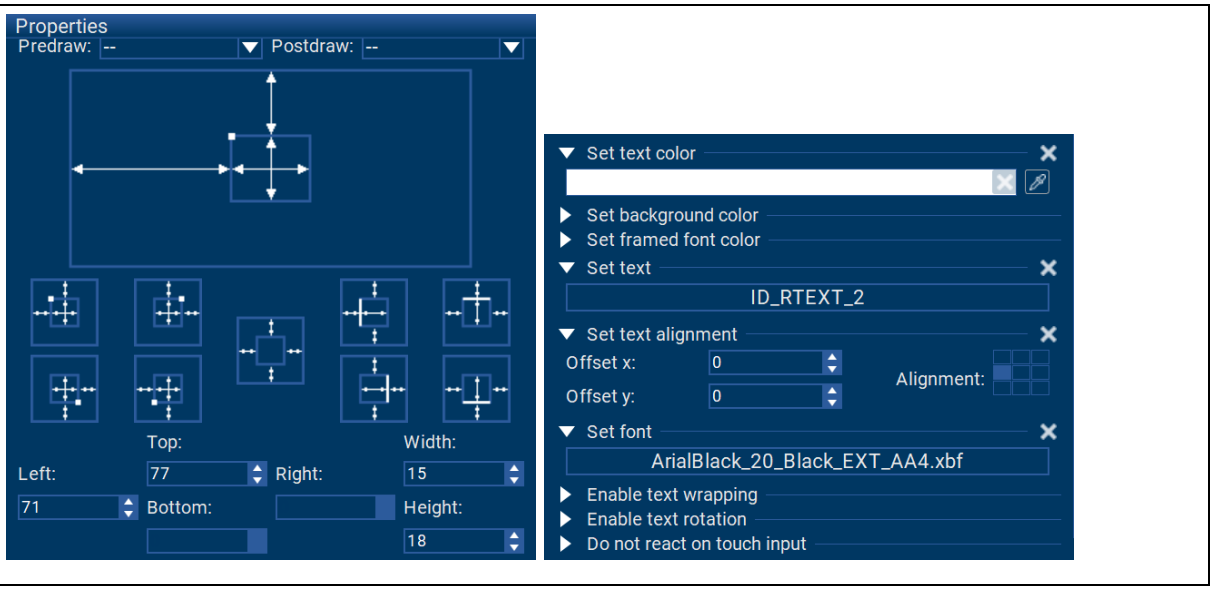


Figure 2-159 Set Properties of ID_TEXT_02

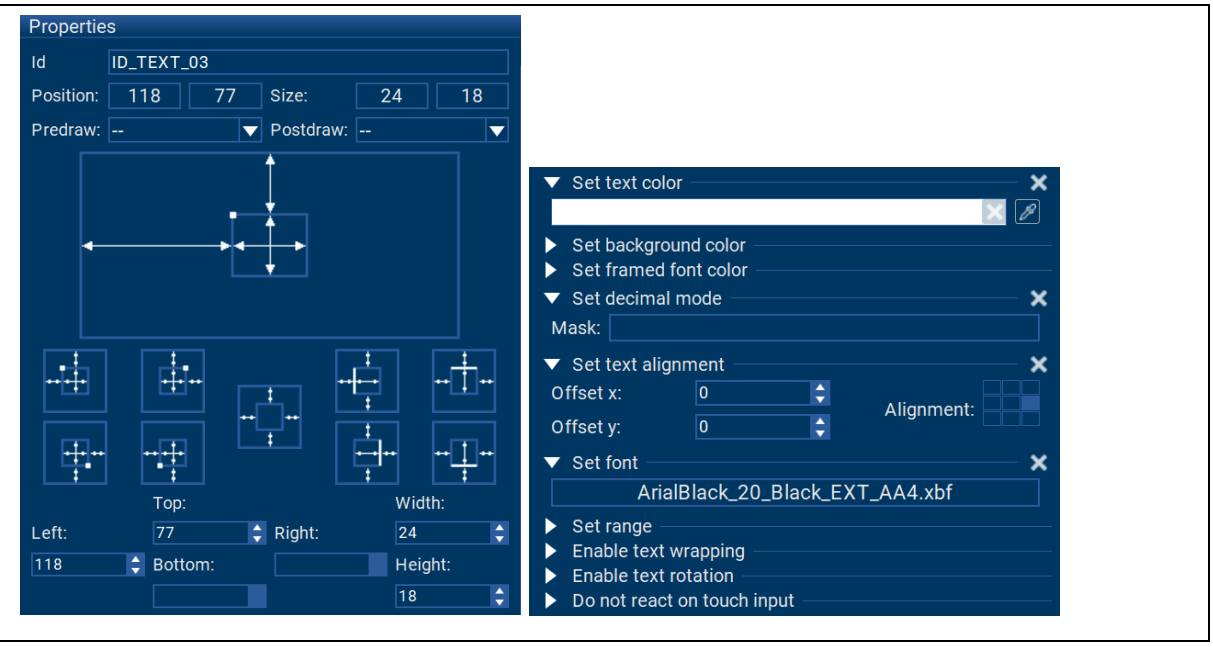


Figure 2-160 Set Properties of ID_TEXT_03

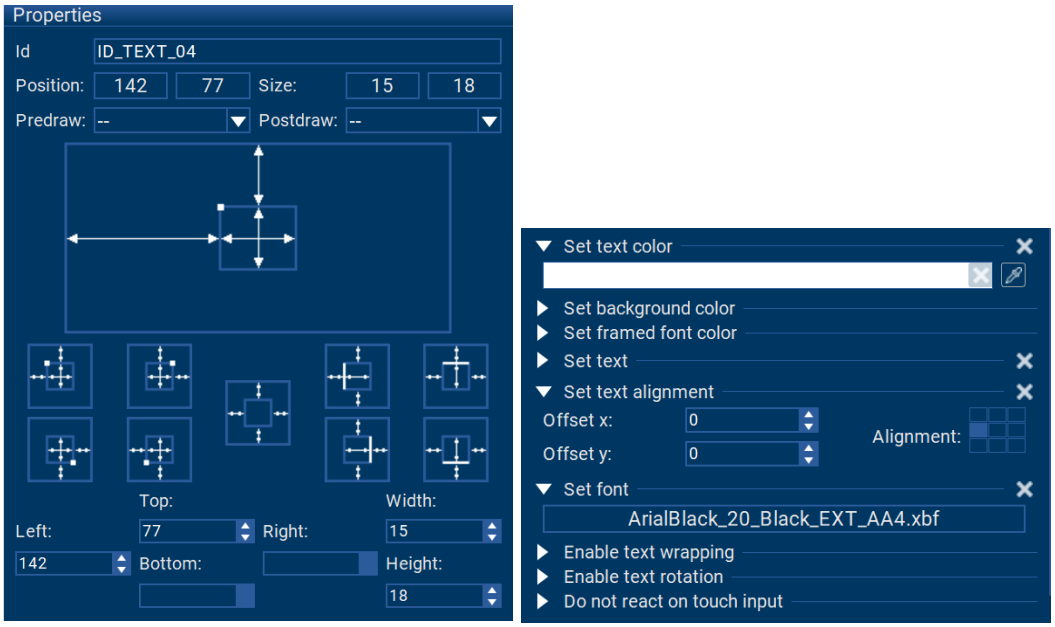


Figure 2-161 Set Properties of ID_TEXT_04

3. Add two Slider objects (SLIDER_T1, SLIDER_T2) to adjust the value. Turn on "Vertical mode" and Invert direction functions in properties (turn on by clicking the arrow), make the slider appear vertical and make the value increase from bottom to top, then use the images *slider_blue_down_16x30.png*, *DARK_Slider_Shaft_V_Top_Gray_16x30.png*, *DARK_Slider_Thumb_Up_22x22.png*, and *DARK_Slider_Thumb_Down_22x22.png* as Shaft bottom, Shaft top, Thumb unpressed and Thumb pressed of bitmap respectively.

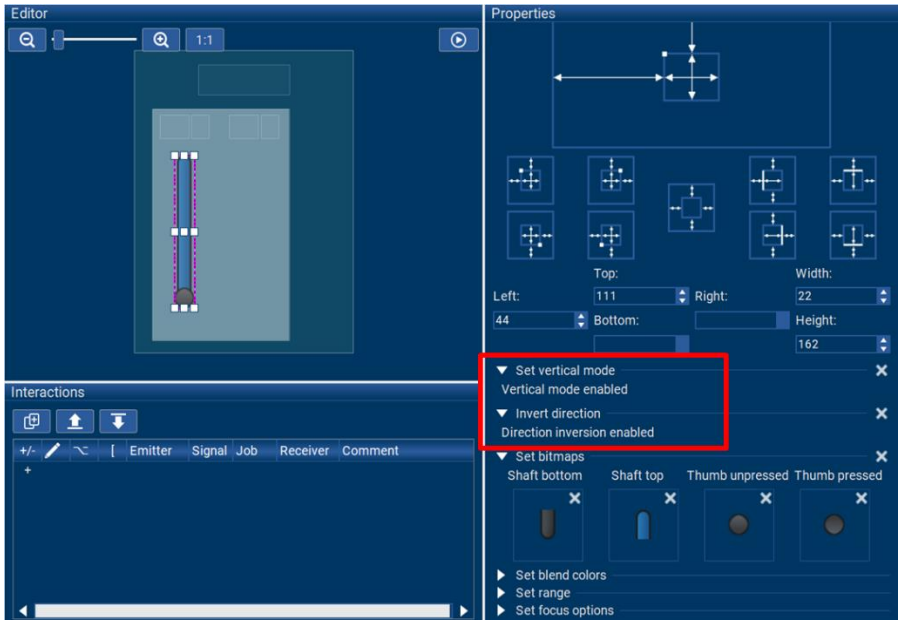


Figure 2-162 Set the Vertical Mode and Invert Direction

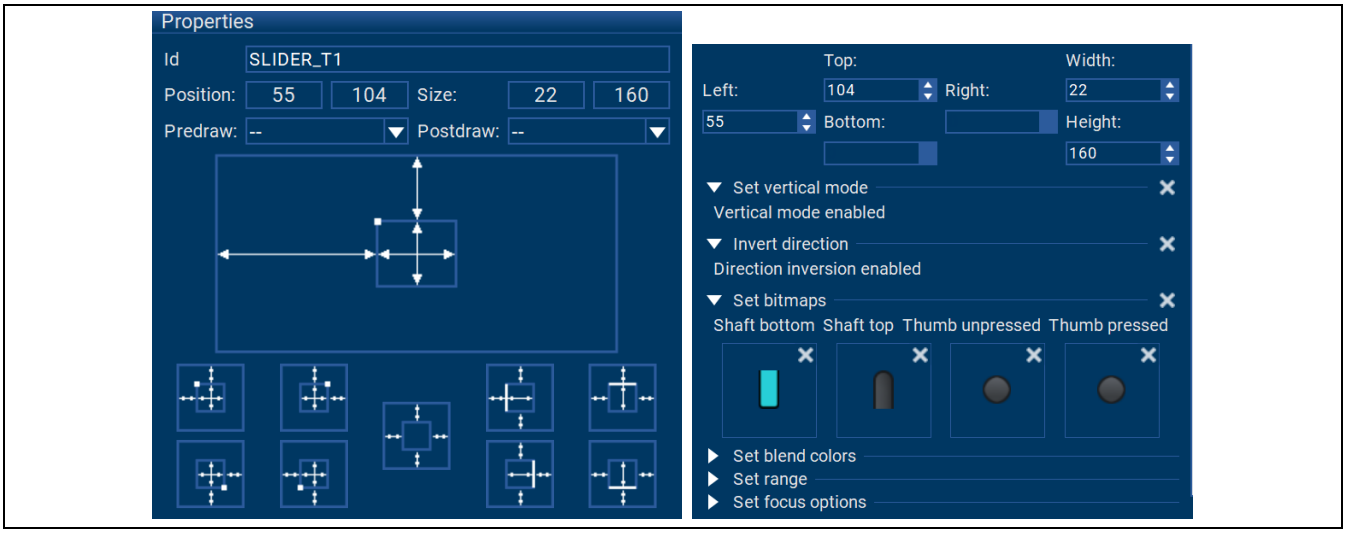


Figure 2-163 Set Properties of SLIDER_T1

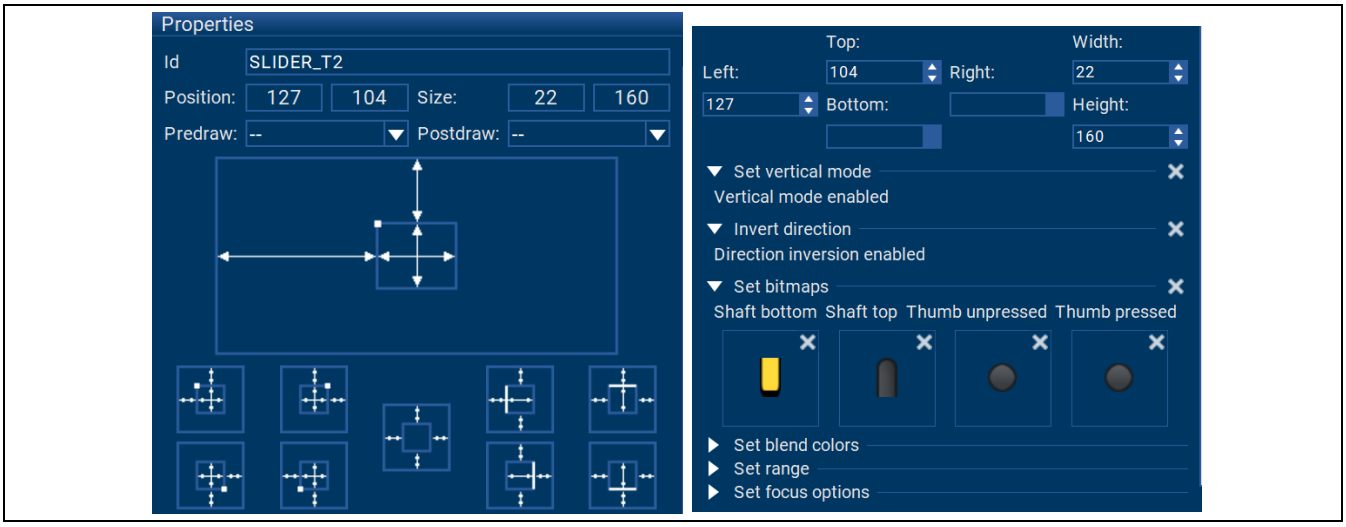


Figure 2-164 Set Properties of SLIDER_T2

4. To set the switch for Sliders, add a Switch object (SWITCH_locker) and choose images *Switch_Button_Pressed_68x22.png* and *switch_Button_Unpressed_68x22.png* as the bitmap, then set the properties according to Figure 2-165 and Figure 2-166.

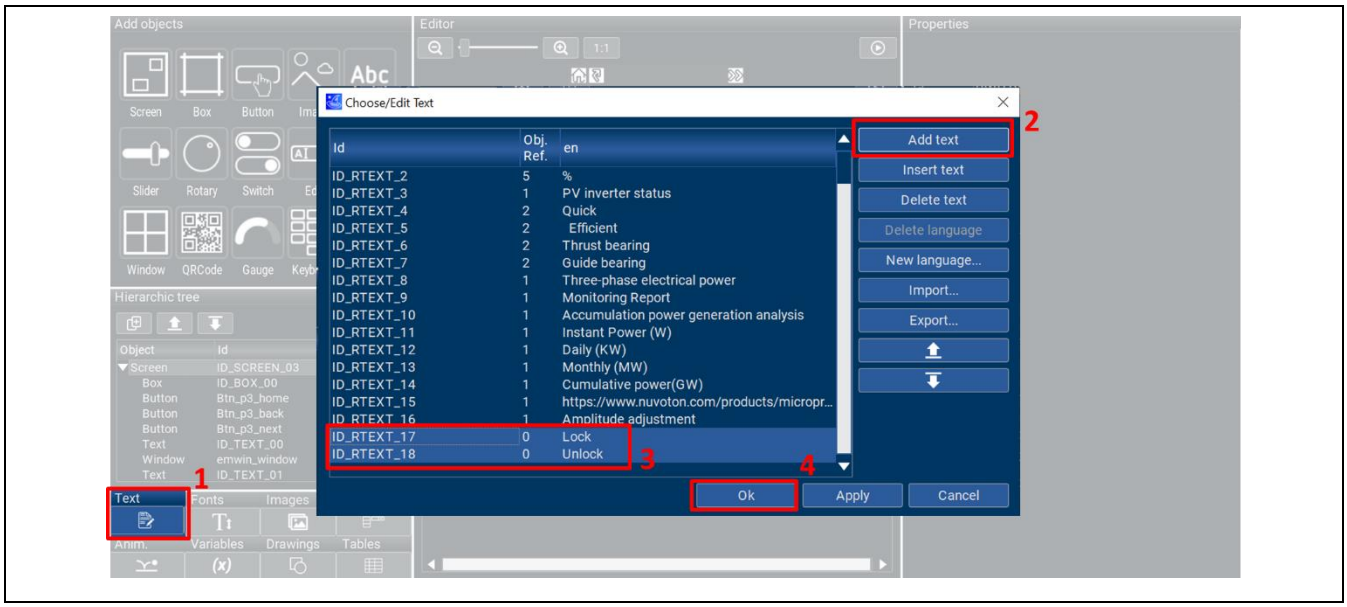


Figure 2-165 Add the Display Content of SWITCH_locker (Lock, Unlock)

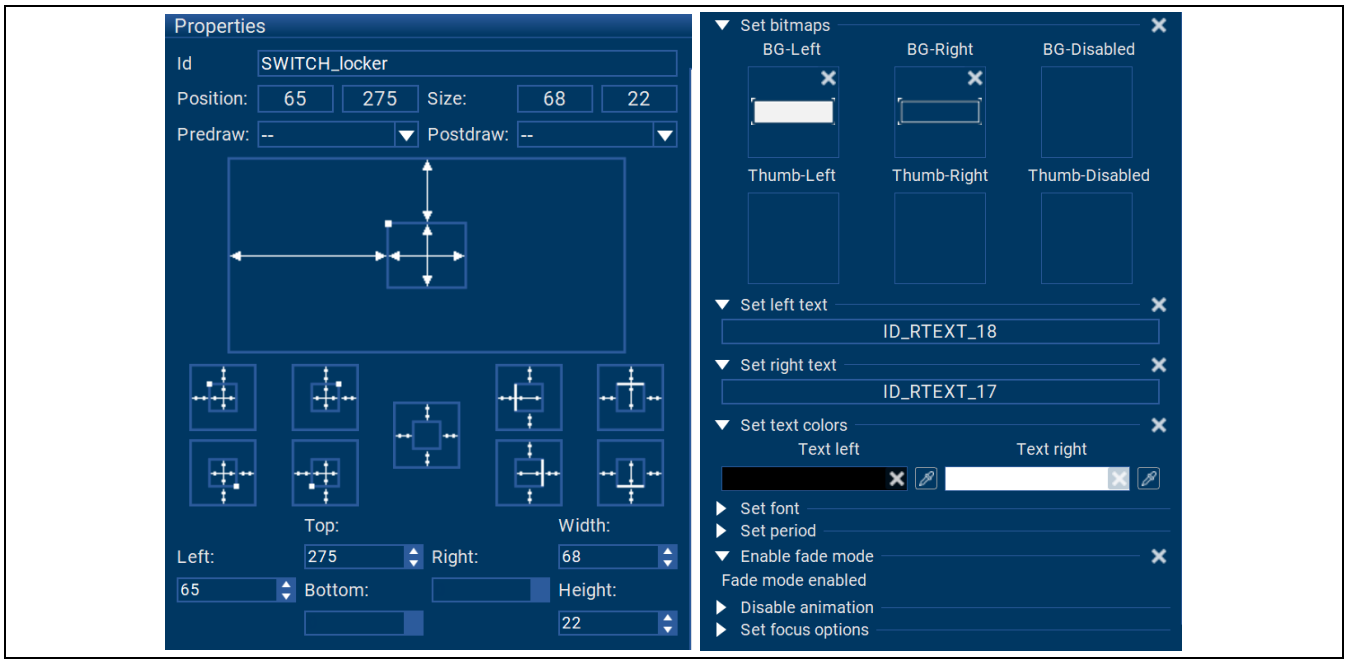


Figure 2-166 Set Properties of SWITCH_locker

5. Add three Button objects for the screen change and upload. Set the properties according to Figure 2-167 to Figure 2-170, then choose the images *home_20x20_new.png*, *arrow2_20x20_new.png*, *Squere_Button_Unpressed_40x40.png*, and *Squere_Button_Pressed_40x40.png* as bitmap.

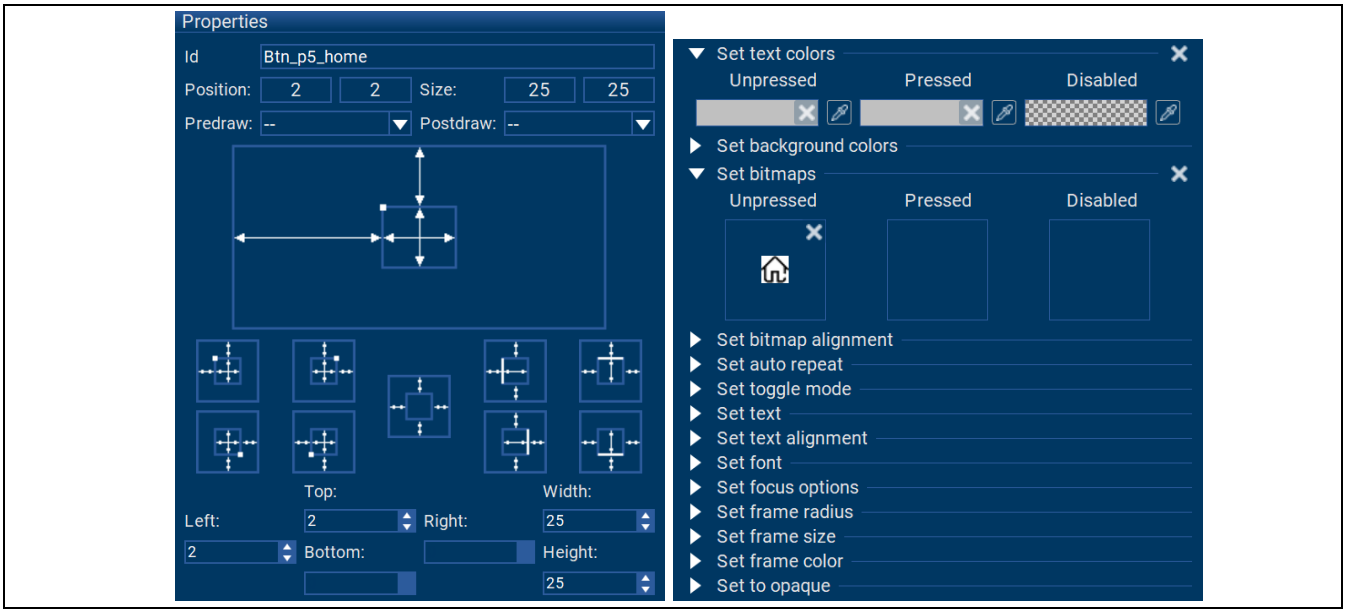


Figure 2-167 Set Properties of btn_p5_home

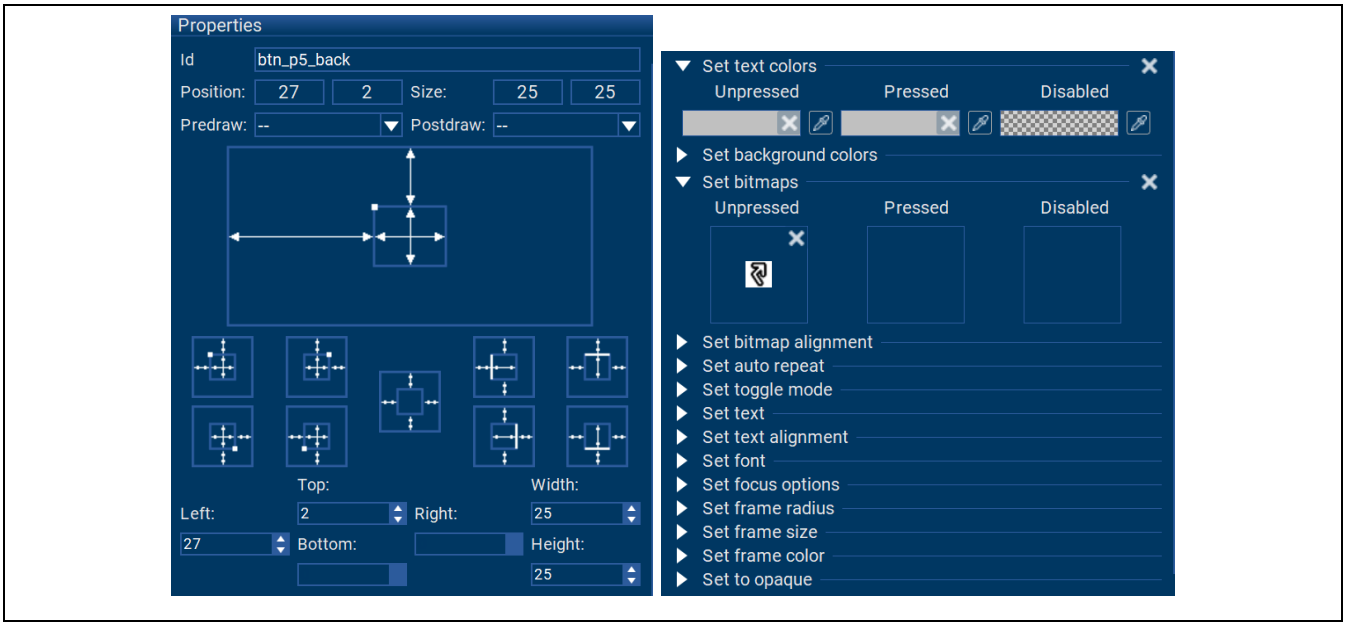


Figure 2-168 Set Properties of btn_p5_back

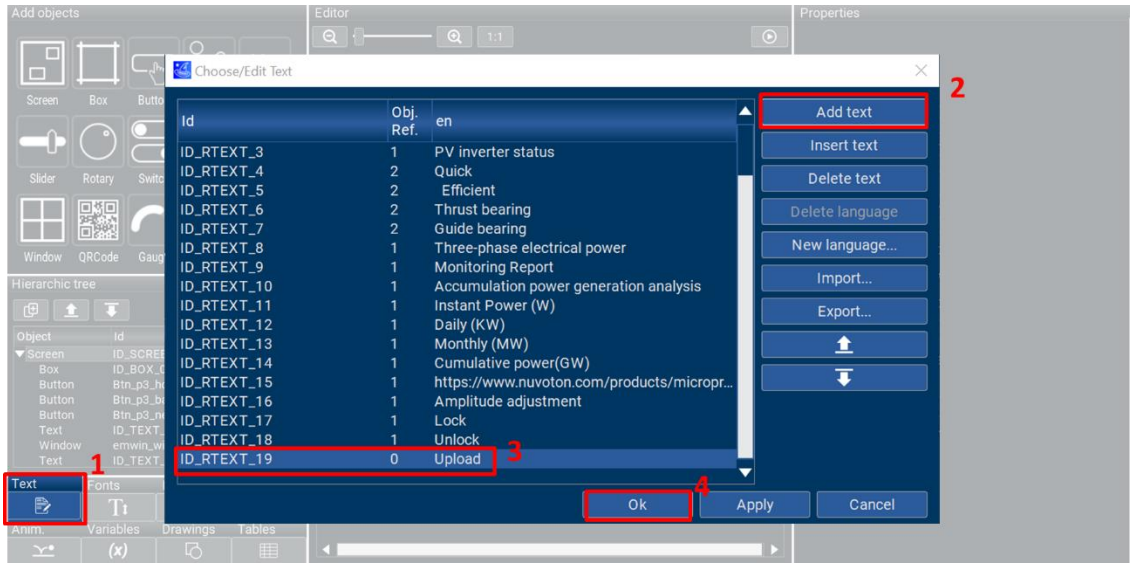


Figure 2-169 Add the Display Content of btn_upload (Upload)

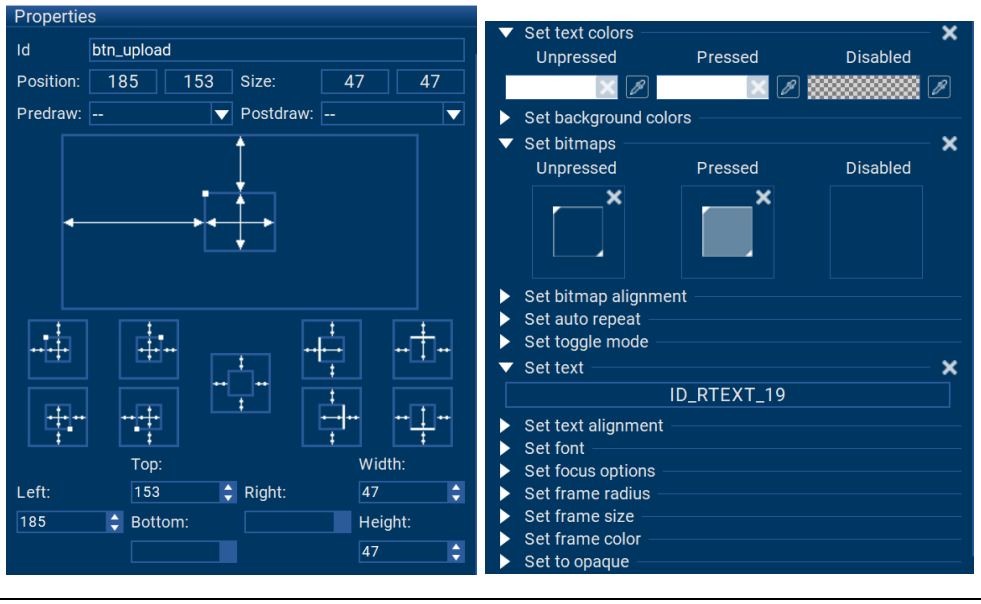


Figure 2-170 Set Properties of btn_upload

2.3.5.2 Interaction Settings

The completed design is shown in Figure 2-171. Next, users need to establish interactions between objects so that corresponding behaviors will be displayed when operating the objects.

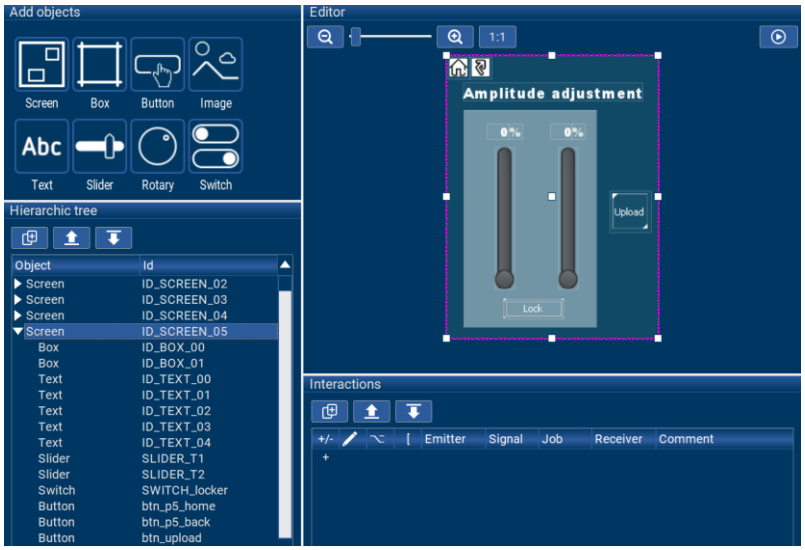


Figure 2-171 Completed Appearance of Screen 5

1. Set the interaction between Slider and Text so that the TEXT object displays the adjusted value of Slider, the settings are shown in Figure 2-172 and Figure 2-173; the settings of closing the Slider object after opening the Switch are shown in Figure 2-174 and Figure 2-176; the settings of re-enabling the Slider object after closing the Switch object are shown in Figure 2-175 and Figure 2-177.

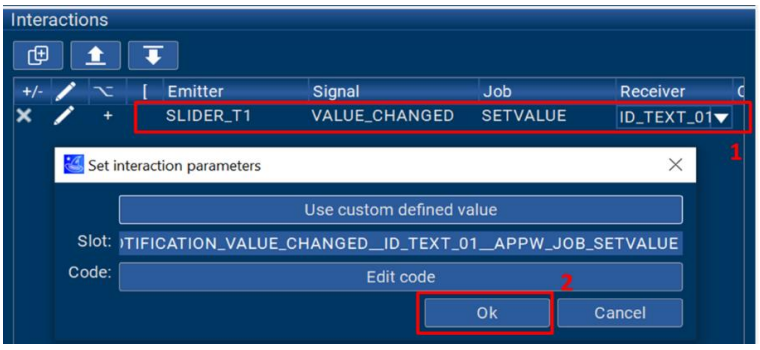


Figure 2-172 Set the Interaction between SLIDER_T1 and ID_TEXT_01

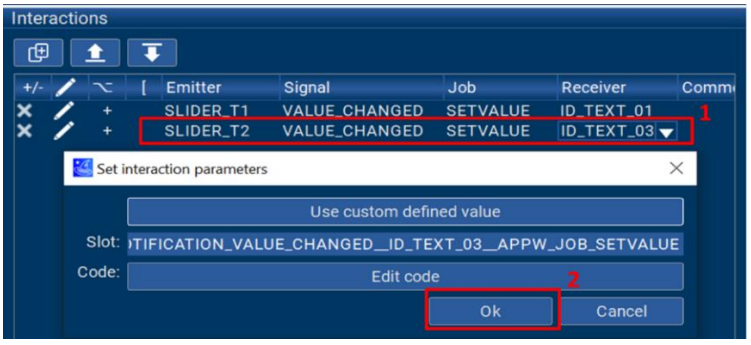


Figure 2-173 Set the Interaction between SLIDER_T2 and ID_TEXT_03

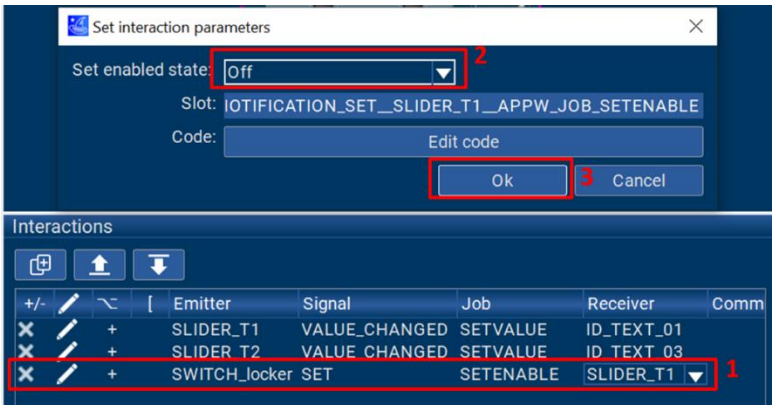


Figure 2-174 Set the Interaction between SLIDER_T1 and Switch_locker (1)

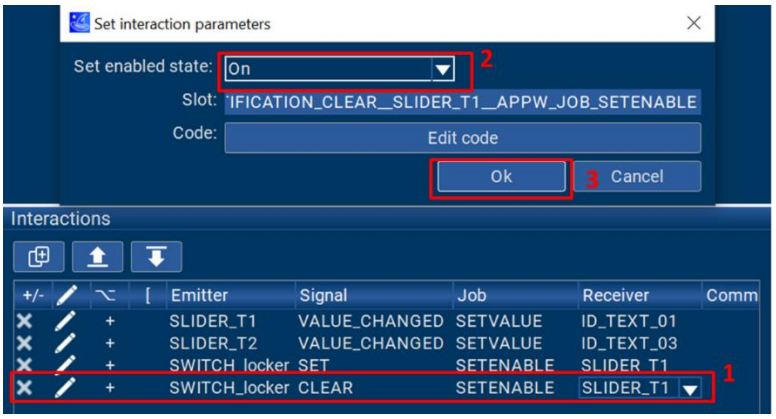


Figure 2-175 Set the Interaction between SLIDER_T1 and Switch_locker (2)

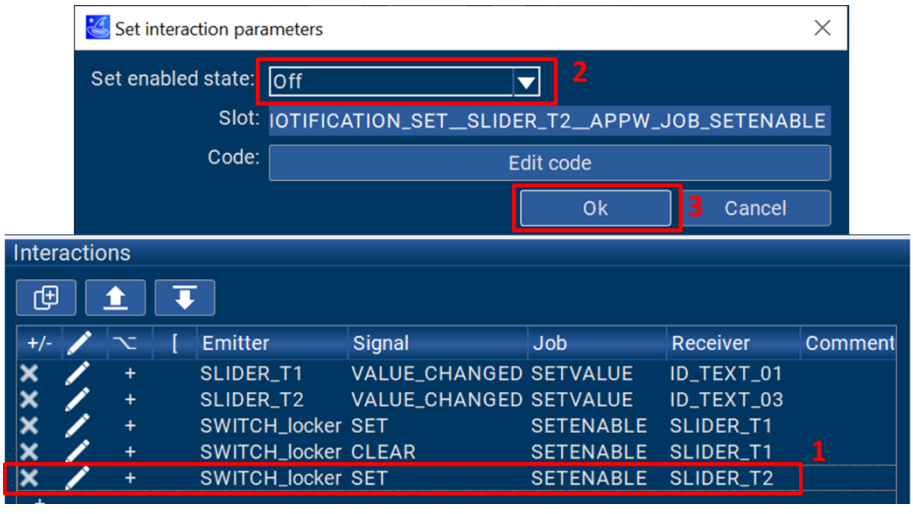


Figure 2-176 Set the Interaction between SLIDER_T2 and Switch_locker (1)

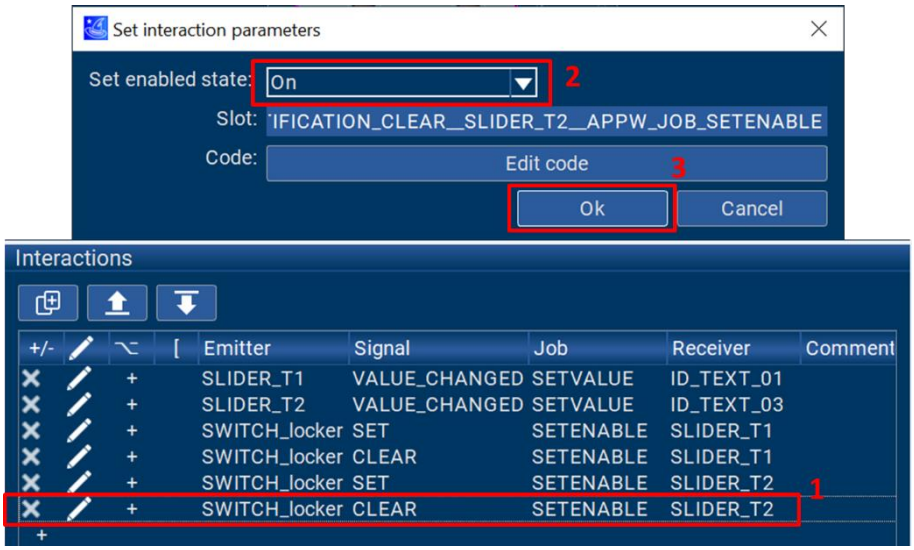


Figure 2-177 Set the Interaction between SLIDER_T2 and Switch_locker (2)

2. Set the screen switch by the Button objects (btn_p5_home, btn_p5_back), and switch the display page to the home page and the previous page through buttons.

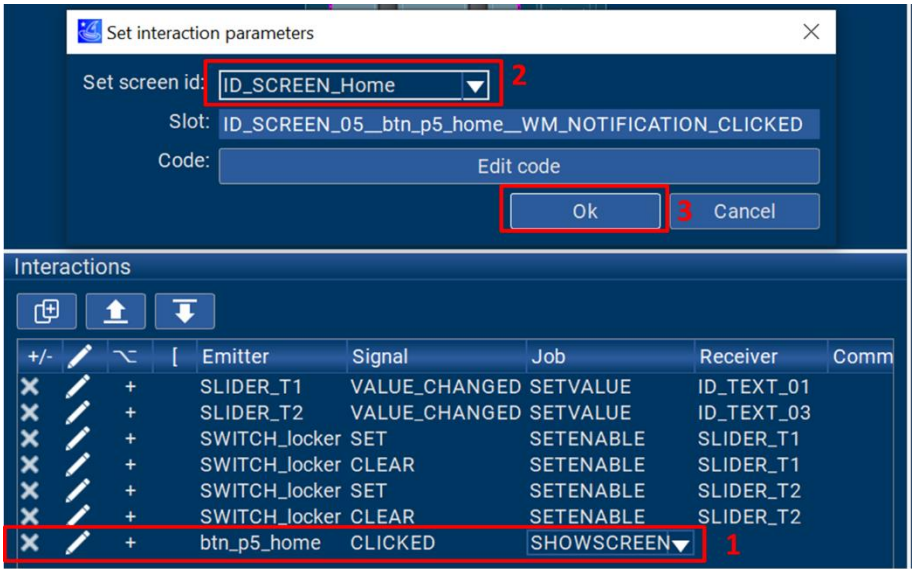


Figure 2-178 Interaction Settings of Screen Switching (btn_p5_home)

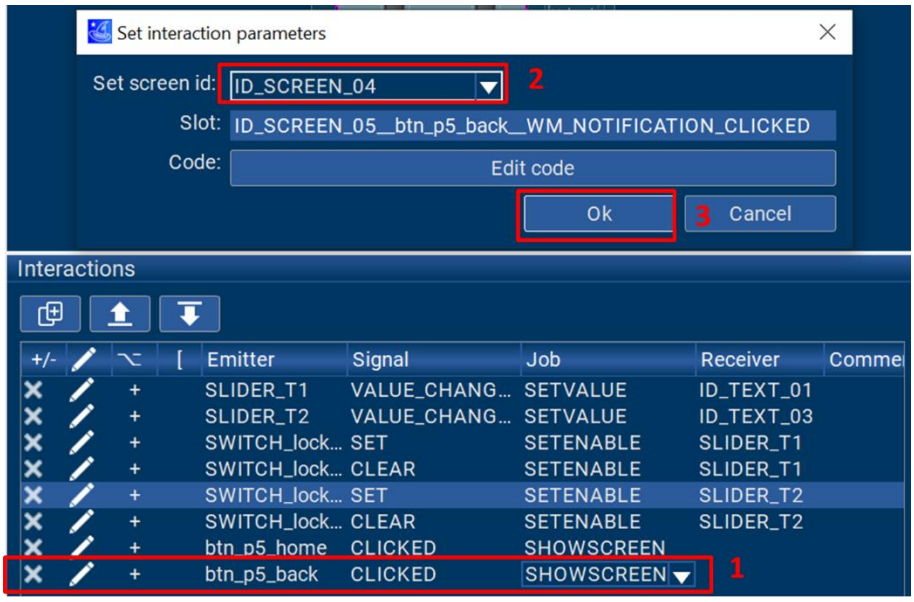


Figure 2-179 Interaction Settings of Screen Switching (btn_p5_back)

2.3.6 Export the AppWizard Project

Save the GUI design and export the project through the toolbar above the AppWizard editor (the method is the same as Figure 2-115), and the generated folder is shown in Figure 2-180.

Figure 2-180 Generated Folders by AppWizard Project

2.4 Deploying the Application to Evaluation Board

2.4.1 How to Build the Bin File by EmWin Package

The emWin package contains all the libraries and resources required by emWin AppWizard. Users only need to put the designed GUI project into the package and execute the **Makefile** to generate the program bin file. For detailed operation procedures, please refer to *README.txt* in the emWin package, and the method of placing the exported AppWizard project into the emWin package is as follows.

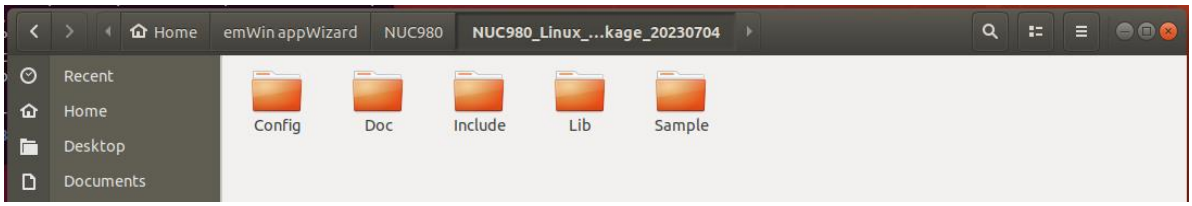


Figure 2-181 Content of EmWin Package (After unzip)

1. Copy the folders "**Resource**" and "**Source**" generated by the AppWizard project to the emWin package path in Linux environment: */Sample/SimpleDemoAppwizard/Application*

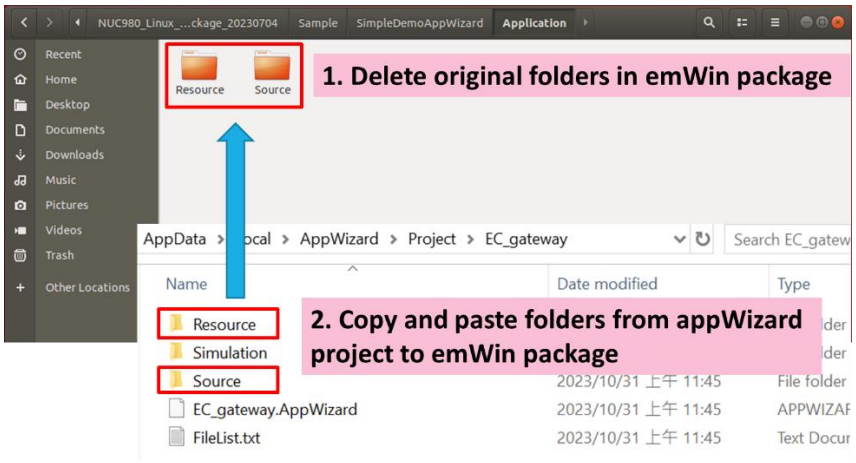


Figure 2-182 Put the AppWizard-generated File into the Package

2. Return to the previous path (*/Sample/SimpleDemoAppwizard*), right-click "Open in Terminal" to open the console window, and enter the command "make" to compile the file, as shown in Figure 2-183 to Figure 2-185.

```
$ make
```

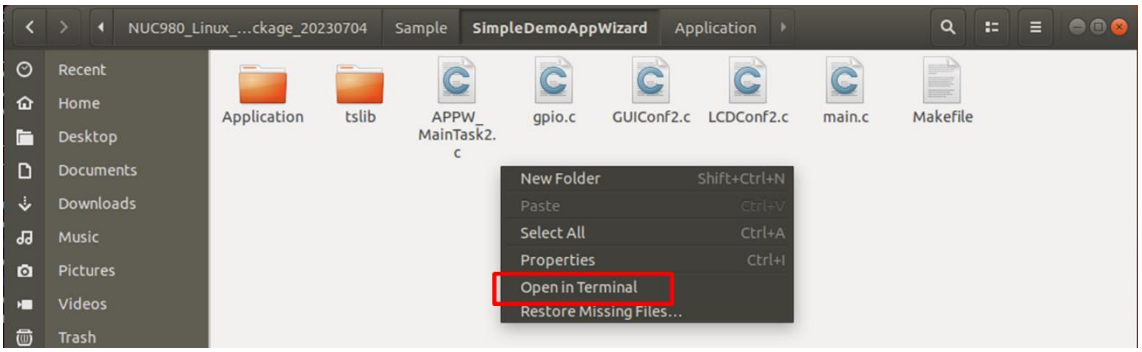


Figure 2-183 Open Terminal Console in the Folder

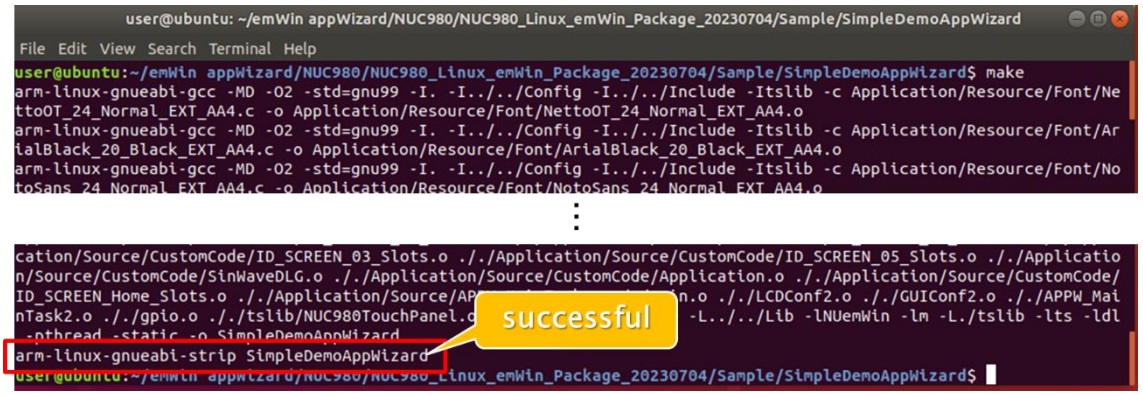


Figure 2-184 File Compilation Success

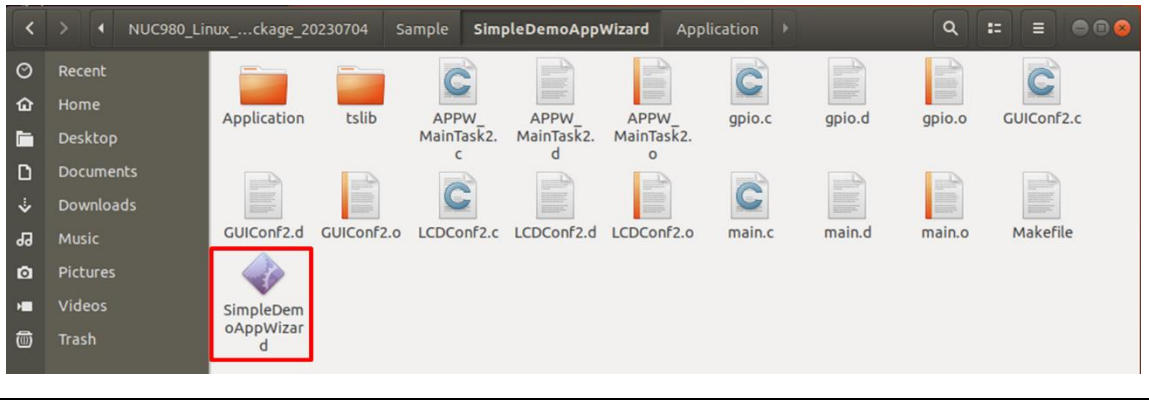


Figure 2-185 Generated Bin File by Compiler

3. Copy the generated bin file to the SD card for the spare. (**Note:** SD card must be the FAT format.)

2.4.2 Execute GUI Program on NuMaker-IIoT-NUC980G2D

Power on the VCOM interface of NuMaker-IIoT-NUC980G2D via Type-C USB. After logging in as "root" in the console then insert the SD card, as shown in Figure 2-186 to Figure 2-188.

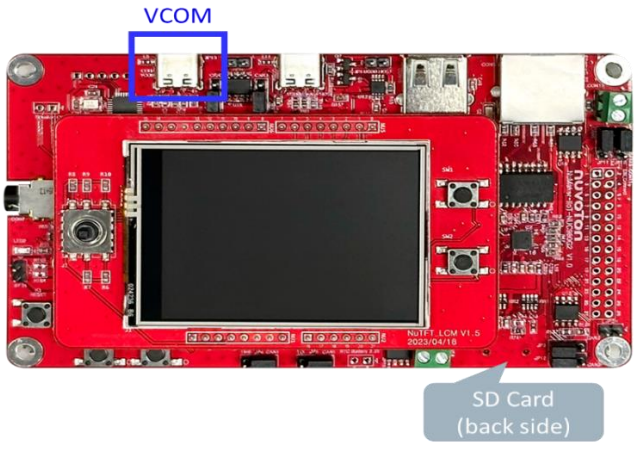


Figure 2-186 VCOM and SD Card Insertion Location of NuMaker-IIoT-NUC980G2D

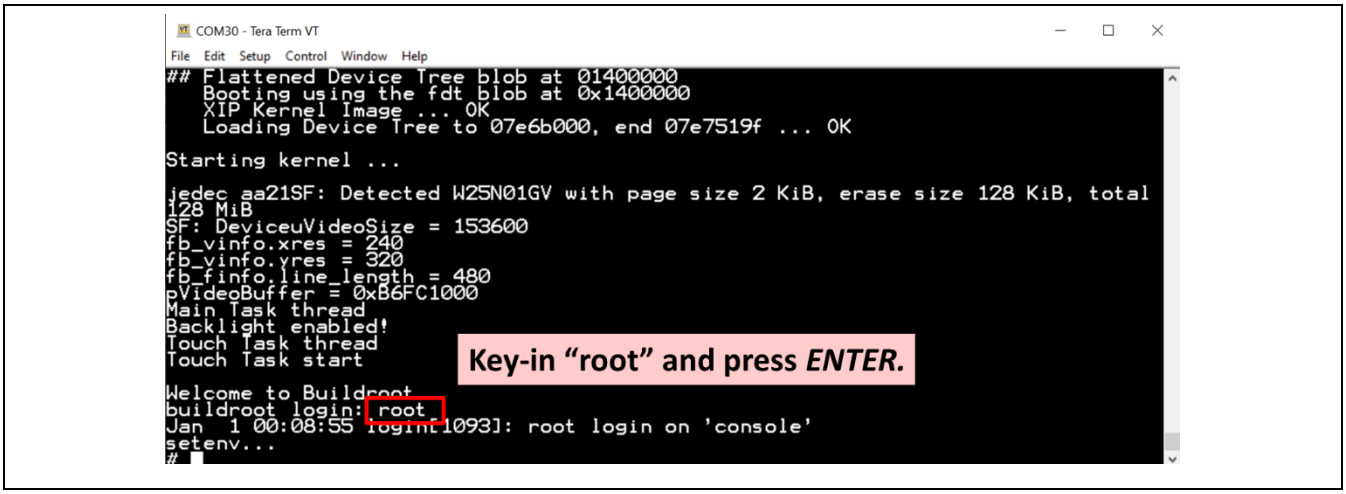


Figure 2-187 Log in the System by "root"

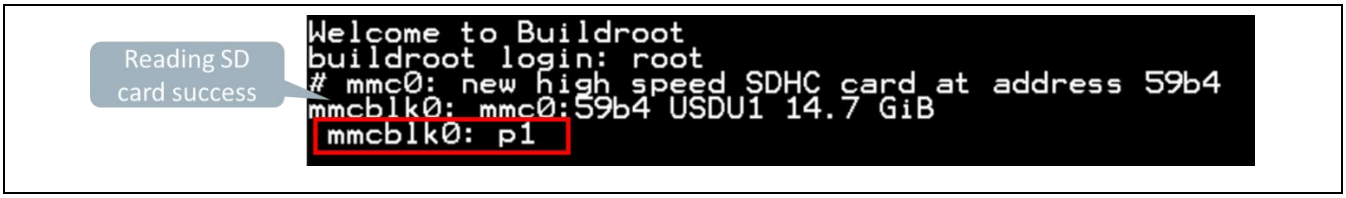


Figure 2-188 Reading Success with SD Card

After inserting the SD card, type the command as follows in the console to mount the SD card contents to NuMaker-IIoT-NUC980G2D:

```
$ mount -t vfat -o noatime,shortname=mixed,utf8 /dev/mmcblk0p1 /mnt/
$ cd /mnt
```

Then set the environment variables required by the GUI:

```
$ cp /mnt/ts.conf /etc
$ export TSLIB_CONFFILE=/etc/ts.conf
$ export TSLIB_TSDEVICE=/dev/input/event0
$ /usr/bin/ts_calibrate  //ONLY ONCE
```

Execute file:

```
$ ./SimpleDemoAppWizard //project name
```

2.4.3 Demo Result

The results of executing this sample code are shown in Figure 2-189.

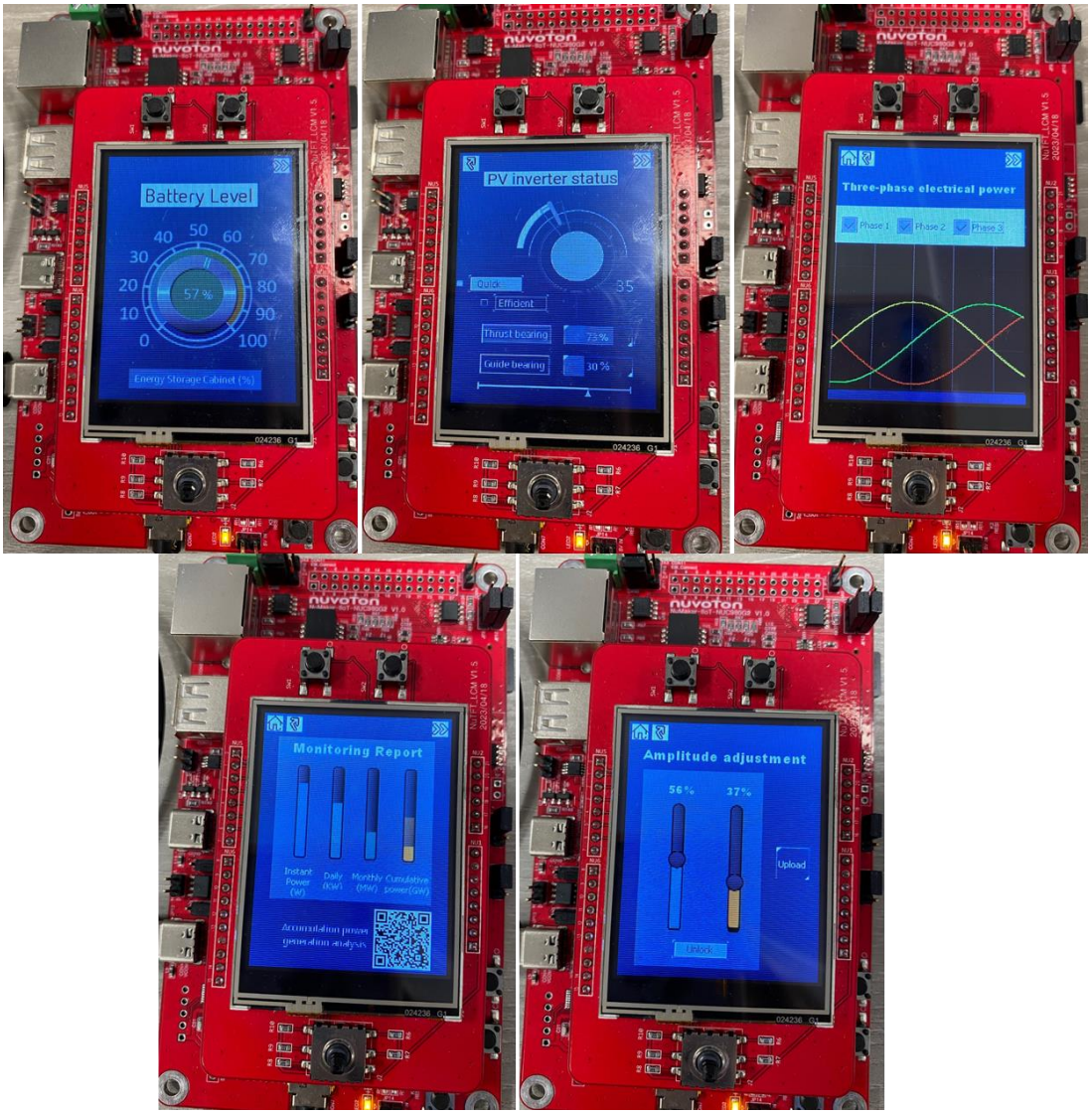


Figure 2-189 Execution Results

3. Code Description

In the emWin Linux package, *main.c* will initialize the GUI program, set up the touch screen, frame buffer, multiple buffering, etc., and then the application main program *MainTask2()* will start executing the project. The following program is located in *main.c*.

```
……
// Declare two frame buffers
unsigned char g_au8FB[240*320*2*2];
unsigned char *g_pu8FB;

void *MainTask_ISR(void *arg)
{
    unsigned int gpio = 0xC7;

    printf("Main Task thread\n");
    //
    // Enable Backligh
    //
    gpio_export(gpio);
    gpio_set_dir(gpio, 1);
    gpio_set_value(gpio, 1);
    printf("Backlight enabled!\n");
    //
    // emWin Entry Point
    //
    GUI_Init();
    //
    // Create a dirty device.
    //
    GUI_DIRTYDEVICE_Create();
    WM_MULTIBUF_Enable(1);
    MainTask2();
}

void *TouchTask_ISR(void *arg)
{
    printf("Touch Task thread\n");
    TouchTask();
}

int main()
{
    int ret;
    int i, t = 0;
    FILE *fpVideoImg;
    int uVideoSize;
    pthread_t tid1, tid2;

    vpost_fd = open("/dev/fb0", O_RDWR);
    if (vpost_fd == -1)
    {
        printf("Cannot open fb0!\n");
        return -1;
    }
```

```
    if (ioctl(vpost_fd, FBIOGET_VSCREENINFO, &fb_vinfo) < 0)
    {
        perror("ioctl FBIOGET_VSCREENINFO");
        close(vpost_fd);
        return -1;
    }
    //
    // emWin Multiple Buffering
    //
    g_pu8FB = (unsigned char *)g_au8FB;

    fb_vinfo.yres_virtual = fb_vinfo.yres * 1;
    fb_vinfo.yoffset = 0;
    if (ioctl(vpost_fd, FBIOPUT_VSCREENINFO, &fb_vinfo))
    {
        perror("ioctl FBIOPUT_VSCREENINFO");
        close(vpost_fd);
        return -1;
    }

    if (ioctl(vpost_fd, FBIOGET_FSCREENINFO, &fb_finfo))
    {
        perror("ioctl FBIOGET_FSCREENINFO");
        close(vpost_fd);
        return -1;
    }

    g_line_length = fb_finfo.line_length;

    uVideoSize = fb_vinfo.xres * fb_vinfo.yres * fb_vinfo.bits_per_pixel / 8 * 1;

    printf("uVideoSize = %d\n", uVideoSize);
    printf("fb_vinfo.xres = %d\n", fb_vinfo.xres);
    printf("fb_vinfo.yres = %d\n", fb_vinfo.yres);
    printf("fb_finfo.line_length = %d\n", fb_finfo.line_length);

    pVideoBuffer = mmap(NULL, uVideoSize, PROT_READ|PROT_WRITE, MAP_SHARED, vpost_fd, 0);

    printf("pVideoBuffer = 0x%08X\n", (unsigned int)pVideoBuffer);
    if(pVideoBuffer == MAP_FAILED)
    {
        printf("LCD Video Map Failed!\n");
        exit(0);
    }

    g_xres = fb_vinfo.xres;
    g_yres = fb_vinfo.yres;
    g_bits_per_pixel = fb_vinfo.bits_per_pixel;

    pthread_create(&tid1, NULL, MainTask_ISR, (void *)"MainTask2");
    pthread_create(&tid2, NULL, TouchTask_ISR, (void *)"TouchTask");

    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);

    /* Close LCD */
```

```
    munmap(pVideoBuffer, uVideoSize);  //return memory
    close(vpost_fd);
    return 0;
}
```

If users need to update the number of available bytes of the GUI and settings of the Multiple Buffering function switch, the related parameters can be adjusted in *LCDConf2.c* and *GUIConf2.c* respectively.

LCDConf2.c:

```
//
// Buffers / VScreens
//
#define NUM_BUFFERS  2 // Number of multiple buffers to be used
#define NUM_VSCREENS 1 // Number of virtual screens to be used
```

GUIConf2.c:

```
#include "GUI.h"
…
//
// Define the available number of bytes available for the GUI
//
#define GUI_NUMBYTES  (4 * 1024 * 1024)
```

After entering the main program MainTask2(), it will execute the initialization of emWin AppWizard and display GUI objects. The following program is located in *APPW_MainTask2.c*.

```
void MainTask2(void) {
  // Setup configuration dependent pointers
  APPW_X_Setup();

  // Initialize AppWizard
  APPW_Init(APPW_PROJECT_PATH);

  // Create all persistent screens except initial screen
  APPW_CreatePersistentScreens();

  // Create initial screen...
  APPW_CreateRoot(APPW_INITIAL_SCREEN, WM_HBKWIN);

  // ...and keep it alive

  while (1) {
#if 0
    while (GUI_Exec1()) {
      APPW_Exec();
    }
    APPW_Exec();
    GUI_X_Delay(5);
#else
    GUI_Exec1();
    APPW_Exec();
    GUI_X_Delay(1);
#endif
  }
}
```

After exporting the AppWizard project, all GUI designs will be coded and stored in c files with corresponding names. For example, the *Source\Generated* file contains the object and its property settings, and the *Source\CustomCode* contains the interaction settings for each screen. If the user wants to perform some additional actions when the objects are triggered, they can add codes in the corresponding programs of *ID_SCREEN_Home_Slots.c* and *ID_SCREEN_02_Slots.c* to *ID_SCREEN_05_Slots.c*.

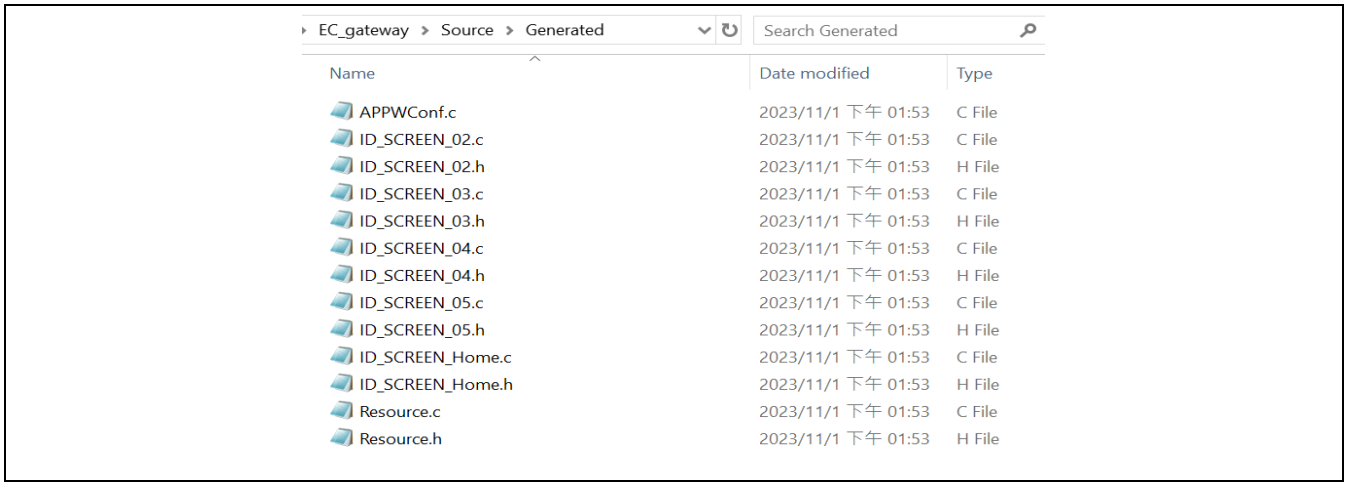


Figure 3-1Exported Files of Object and the Properties

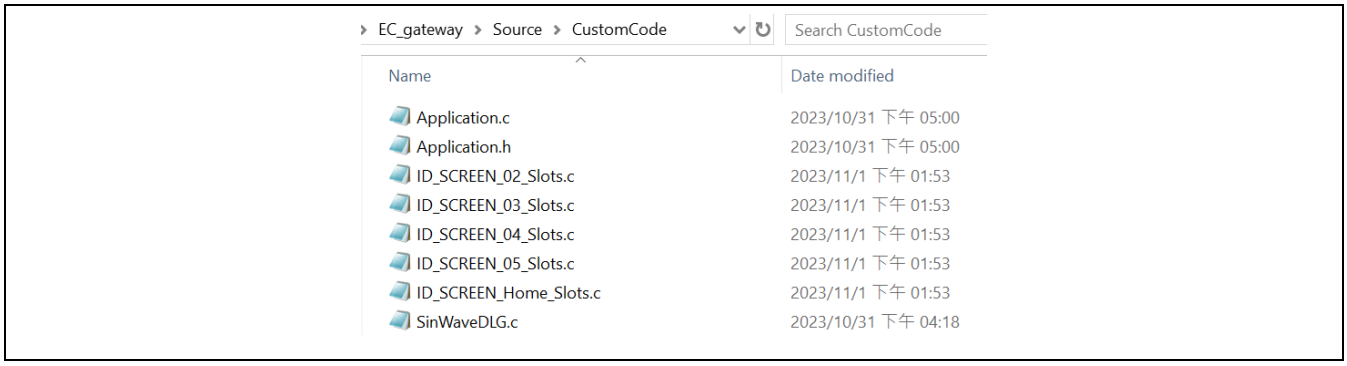


Figure 3-2 Exported Files of Interactions of Object

For example, add codes when interacting with objects on the main screen. Users can find the corresponding interaction through notes and add user codes in functions. The following codes are located in *ID_SCREEN_Home.c*.

```
// interaction target : ID_SCREEN_Home__ID_ROTARY_val__WM_NOTIFICATION_VALUE_CHANGED
void ID_SCREEN_Home__ID_ROTARY_val__WM_NOTIFICATION_VALUE_CHANGED(APPW_ACTION_ITEM *
pAction, WM_HWIN hScreen, WM_MESSAGE * pMsg, int * pResult) {
  GUI_USE_PARA(pAction);
  GUI_USE_PARA(hScreen);
  GUI_USE_PARA(pMsg);
  GUI_USE_PARA(pResult);

  /*
    Add the customer's code in here
                                    */
}
……
```

4. Software and Hardware Requirements

4.1 Software Requirements

- Version of IDE
  - Ubuntu 18.04
- Version of emWin AppWizard IDE
  - AppWizard_V134a_630a

4.2 Hardware Requirements

- Circuit components
  - NuMaker-IIoT-NUC980G2D V1.0
- Connect NuMaker-IIoT-NUC980G2D and PC
  - Connect the VCOM of NuMaker-IIoT-NUC980G2D to the PC via Type-C USB, as shown in Figure 4-1.
  - Turn the NuMaker-IIoT-NUC980G2D evaluation board to the back and insert the microSD into the card slot, as shown in Figure 4-2.

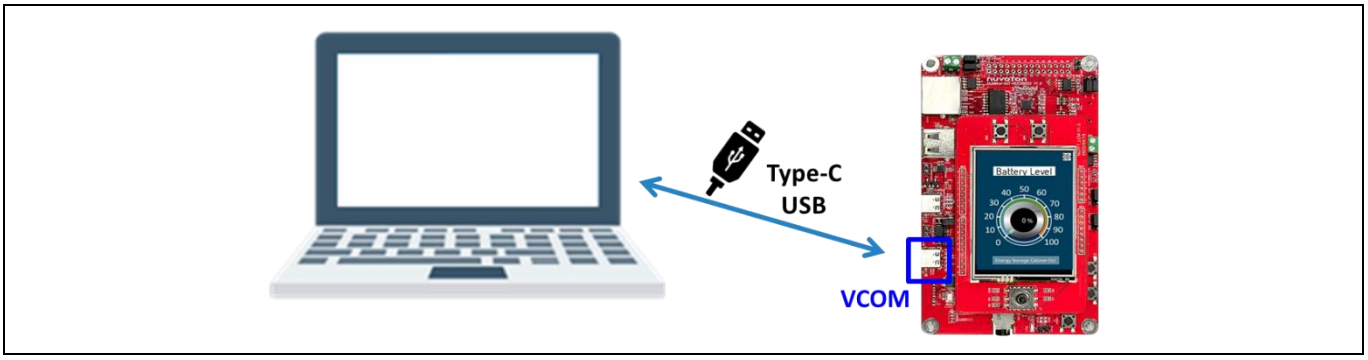


Figure 4-1 Connect NuMaker-IIoT-NUC980G2D to PC via Type-C USB Cable

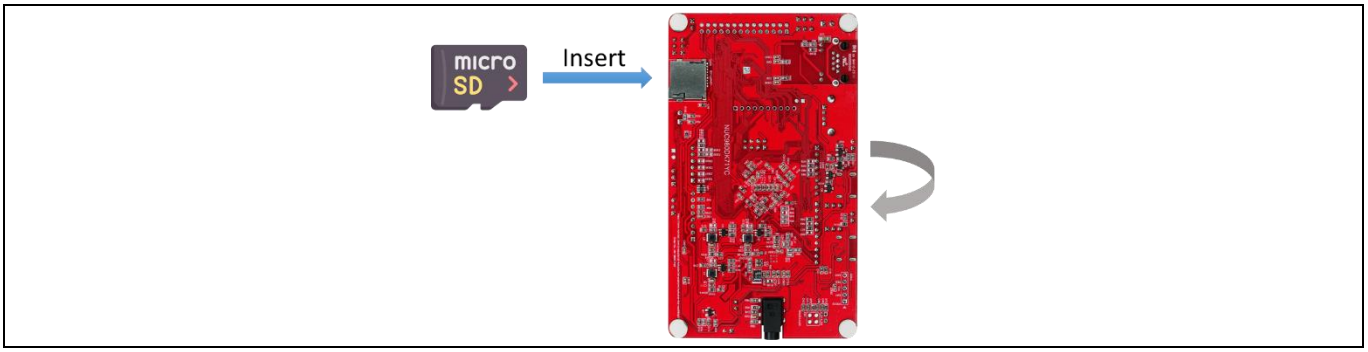


Figure 4-2 Insert the MicroSD into the Card Slot

5. Directory Information

The directory structure is shown below.

EC_NUC980_AppWizard_New_Energy_Gateway_V1.00	
Software Tools	Necessary software tools for this example.
EC_gateway	New energy gateway sample project of AppWizard. Please open it by *AppWizard_V134a_630a.exe*.
new_energy_gateway	The execute file of this example code. Please execute in the evaluation board.
ts.conf	A configuration file used to adjust the calibration, analysis method, compensation and other data of the touch screen.

Figure 5-1 Directory Structure

6. Example Code Execution

1. Download the emWin package on the Nuvoton Website, copy and decompress the file to the Linux environment, then follow the steps of *README.txt*. (Details are in Section 2.4.1)
2. Copy folders "*Resource*" and "*Source*" generated by AppWizard (As the folder “EC_gateway” in the directory) to the Linux emWin package path: */Sample/SimpleDemoAppwizard/Application* (**Note:** Please delete the original folders directly, avoiding replacing them by copy and paste).
   - Execute file ***Makefile*** in Linux environment to generate bin file of AppWizard project.
   - Place the bin file in the SD card and insert it into the SD card slot of the evaluation board.
3. In the Windows environment, log in to the system with root privileges through the console.
   - Mount the data of SD card.
   - Execute bin file.

Note: If it is found that some widgets are invalid when executing the program, it may be because the versions of emWin Package and AppWizard project are not matched. If this happens, please follow the steps below:

- Install the latest version of AppWizard IDE provided in emWin package.
- After the installation, open the AppWizard project (*../EC_NUC980_AppWizard_New_Energy_Gateway_V1.00/EC_gateway*) by the latest version of AppWizard IDE.
- Open the project, and a reminder window will pop up that the project needs to be upgraded. Click ***Yes*** to upgrade.
- After the project is upgraded, regenerate the AppWizard project through the toolbar: ***File->Export & Save***, and then go back to step 2 to continue the example code execution process.

7. Revision History

Date	Revision	Description
2024.04.02	1.00	Initial version.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.