GUIGUIDERUG_1.7.1

GUI Guider v1.7.1 User Guide

Rev. 15 — 29 March 2024 **User guide**

Document information

Information	Content
Keywords	GUIGUIDERUG, IDE, GUI, MCU, MPU, LVGL, RTOS, Yocto Linux. QNX, FreeMASTER
Abstract	This document describes GUI Guider and targets embedded GUI application developers with a basic knowledge of C on NXP MCU and MPU devices.

1 Welcome

Welcome to the official documentation of GUI Guider!

Figure 1. GUI Guider IDE layout

If you are unfamiliar with GUI Guider, we recommend reading on to get an overview of what this document has to offer. If you encounter any issues or have questions, feel free to visit our forum to communicate. We greatly value your suggestions as they are crucial to our ongoing improvement.

1.1 Overview

The GUI Guider is built on the Light and Versatile Graphics Library (LVGL) library. GUI Guider provides an IDE to design embedded graphic application UI using drag-and-drop widgets and helps in the editing process. The software facilitates the UI design for graphic applications on embedded devices.

This document describes GUI Guider and targets embedded GUI application developers with a basic knowledge of C on NXP MCU devices.

1.2 Content

The major sections of this user guide are as follows:

- Getting started: General information and feature list of GUI Guider, and how to quick start.
- IDE function: Essential GUI Guider function usage.
- Widgets: Introduces the widgets and events.
- Development: How to develop a GUI Guider application, including debugging, performance, porting OS, and so on.
- Tutorials: The common use cases.
- Miscellaneous: Frequently asked questions and answers, and known issues.

2 Getting started

This section describes the key features and installation. Once completed, you can start your first GUI Guider project.

2.1 Introduction

This section describes the important features and target support of the GUI Guider.

2.1.1 Feature

- WYSIWYG HMI designer with drag and drop support.
- Multiple languages for IDE UI: English and Chinese.
- Multiple themes for IDE UI.
- Support two versions of the open source LVGL graphics library.
- Shortcut key and shortcut button for common functions.
- 40+ widgets with customizable attributes, events, actions, animations, and flags.
- Resource management for images, fonts, video, and Lottie.
- FreeMASTER debug function.
- GUI auto-scaling for using HMI design on different boards and panels.
- Autogenerate C and Micro-Python source files.
- Custom code interface for function extension.
- Code viewer and editor, real-time log viewer.
- Auto build and deploy on simulator and NXP devices.
- PXP and VGLite acceleration enable and disable.
- Project upgrader and backward compatibility.
- Seamlessly integrate with MCUX, IAR, Keil, Yocto, QNX.
- Cross host OSes: Windows, Linux, and macOS(X86, ARM).
- Support NXP general purpose MCU, crossover MCU, wireless MCU, and MPU.
- Multiple target OSes: FreeRTOS, Zephyr, RT-Thread, Linux, and QNX.
- Build-in HMI reference design and widgets demo.
- Cloud online templates and offline templates package.

2.1.2 Support widgets

The widgets of v7 are listed in Table 1.

Table 1. v7 widgets

V7 widget type	Widgets list
Button	button, image button, checkbox, button group, and switch
Form	label, drop-down list, text area, and calendar
Table	table, tab, message box, container, chart, canvas, list, window, and titleview
Shape	arc, line, roller, LED, spinbox, color picker, and spinner
Image	image, animation image, and 3D image
Progress	bar and slider
Gauge	gauge and line meter

The widgets of v8 are listed in Table 2.

Table 2. v8 widgets

V8 widget type	Widgets list
Button	button, image button, checkbox, button group, and switch

Table 2. v8 widgets...*continued*

V8 widget type	Widgets list
Form	label, spangroup, drop-down list, text area, calendar, and date text box
Table	table, tab, message box, container, chart, canvas, list, window, titleview, and menu
Gauge	meter
Shape	arc, line, roller, LED, spinbox, color picker, and spinner
Image	image, animation image, and 3D image
Progress	bar and slider
Advanced	analog clock, carouse, video, lottie, QR code, barcode, digital clock, radio button, and text progress bar

2.1.3 Built-in fonts

Table 3 lists the built-in fonts.

Table 3. Built-in fonts

ID	Name
1	SourceHanSerifSC-Regular (Support Chinese characters)
2	Arial
3	montserratMedium
4	Abel-regular
5	Acme-Regular
6	Adventpro-regular
7	AguafinaScript-Regular
8	Alatsi-Regular
9	AlexBrush-Regular
10	AmaticSC-Regular
11	Amiko-Regular
12	Antonio-Regular
13	ArchitectsDaughter
14	FontAwesome5

2.1.4 Target

GUI Guider supports in-designing HMI applications for NXP MCUs and MPUs, such as i.MX RT, LPC, MCX, KW, RW, and i.MX. Each platform includes build-in support for multiple LCD display.

Table 4. Supported boards

Type	Board name	Verified display part number	Note
i.MX RT	MIMXRT1010-EVK	adafruit-1947	
i.MX RT	MIMXRT1015-EVK	adafruit-1947	

Table 4. Supported boards*...continued*

Type	Board name	Verified display part number	Note
i.MX RT	MIMXRT1020-EVK	adafruit-1947	
i.MX RT	MIMXRT1024-EVK	adafruit-1947	
i.MX RT	MIMXRT1040-EVK	RK043FN02H-CT, RK043 FN66HS-CTG	
i.MX RT	MIMXRT1050-EVKB	RK043FN02H-CT, RK043 FN66HS-CTG	
i.MX RT	MIMXRT1060-EVK	RK043FN02H-CT, RK043 FN66HS-CTG	
i.MX RT	MIMXRT1060-EVKB	RK043FN02H-CT, RK043 FN66HS-CTG	
i.MX RT	MIMXRT1060-EVKC	RK043FN02H-CT, RK043 FN66HS-CTG	
i.MX RT	MIMXRT1064-EVK	RK043FN02H-CT, RK043 FN66HS-CTG	
i.MX RT	MIMXRT595-EVK	G1120B0MIPI, Mikroe TFT Proto 5", RK055AHD091, RK055MHD091	
i.MX RT	MIMXRT1160-EVK	RK055AHD091, RK055 MHD091	
i.MX RT	MIMXRT1170-EVK	RK055AHD091, RK055 MHD091	
i.MX RT	MIMXRT1170-EVKB (portrait mode, landscape mode)	RK055AHD091, RK055 MHD091	
LPC	LPCXpresso54628	RK043FN02H-CT, RK043 FN66HS-CTG	
LPC	LPCXpresso54S018	RK043FN02H-CT, RK043 FN66HS-CTG	
LPC	LPCXpresso54S018M	RK043FN02H-CT, RK043 FN66HS-CTG	
LPC	LPCXpresso55S06	adafruit-1947	
LPC	LPCXpresso55S16	adafruit-1947	
LPC	LPCXpresso55S28	adafruit-1947	
LPC	LPCXpresso55S36	adafruit-1947	
LPC	LPCXpresso55S69	adafruit-1947	
MCX	MCX-N5xx-EVK	adafruit-1947	IAR embedded Workbench 9.30.1, Keil MDK 5.37, MCUXpresso 11.7.0
MCX	FRDM-MCXN947	PAR-LCD-S035	IAR embedded Workbench 9.40.2, Keil MDK 5.38.1, MCUXpresso 11.9.0
KW	KW45B41Z-EVK	ePaper-Shield	IAR embedded Workbench 9.32.1, MCUXpresso 11.7.1

Table 4. Supported boards*...continued*

Type	Board name	Verified display part number	Note
RW	RD-RW612-BGA	adafruit-1947	IAR embedded Workbench 9.32.1, Keil MDK 5.37, MCUXpresso 11.7.0
MPU	MCIMX93EVK	**EV121WXM-N12-3GP0**	

GUI Guider provides device template for supported platforms. The HMI application can be built and deployed to target devices by GUI Guider.

Table 5. Status of advance functions

Feature list	LVGL version	Toolchain	Platform
widget:lottie	v8.3.10	MCUXpresso	MIMXRT1040-EVK, MIMXRT1050-EVKB, MIMXRT1060-EVK, MIMXRT1064-EVK, MIMXRT1160-EVK, MIMXRT1170-EVK,MIMXRT1060-EVKB,MIMXRT1060-EVKC
widget:video	v8.3.10	MCUXpresso, IAR, ARMGcc, MDK	MIMXRT1040-EVK, MIMXRT1050-EVKB, MIMXRT1060-EVK, MIMXRT1064-EVK, MIMXRT1160-EVK, MIMXRT1170-EVK,MIMXRT1060-EVKB,MIMXRT1060-EVKC
SD card storage	v8.3.10	MCUXpresso, IAR, ARMGcc, MDK	MIMXRT1040-EVK, MIMXRT1050-EVKB, MIMXRT1060-EVK, MIMXRT1064-EVK,LPCXpresso54S018M,MIMXRT1060-EVKB,MIMXRT1060-EVKC

2.2 Installation

This section describes the steps to install GUI Guider.

2.2.1 Hardware requirement for LVGL application

Every modern controller, which is able to drive a display is suitable to run LVGL. The minimal requirements are as follows:

- 16, 32, or 64-bit microcontroller or processor
- 16 MHz clock speed is recommended
- Flash/ROM: > 64 kB for the essential components (> 180 kB is recommended)
- RAM:
  - Static RAM usage: ~2 kB depending on the used features and object types
  - Stack: > 2 kB (> 8 kB is recommended)

- Dynamic data (heap): > 4 kB (> 32 kB is recommended if using several objects). Set by `LV_MEM_SIZE` in `lv_conf.h`
- Display buffer: > "Horizontal resolution" pixels (> 10 × "Horizontal resolution" is recommended)
- One frame buffer in the MCU or in an external display controller

- Basic C (or C++) knowledge: pointers, structures, and callbacks

Note: *Memory usage can vary depending on architecture, compiler, and build options.*

2.2.2 Windows 10

To install GUI Guider on Windows 10, download the installer from www.nxp.com/gui-guider, and run the `setup.exe` file.

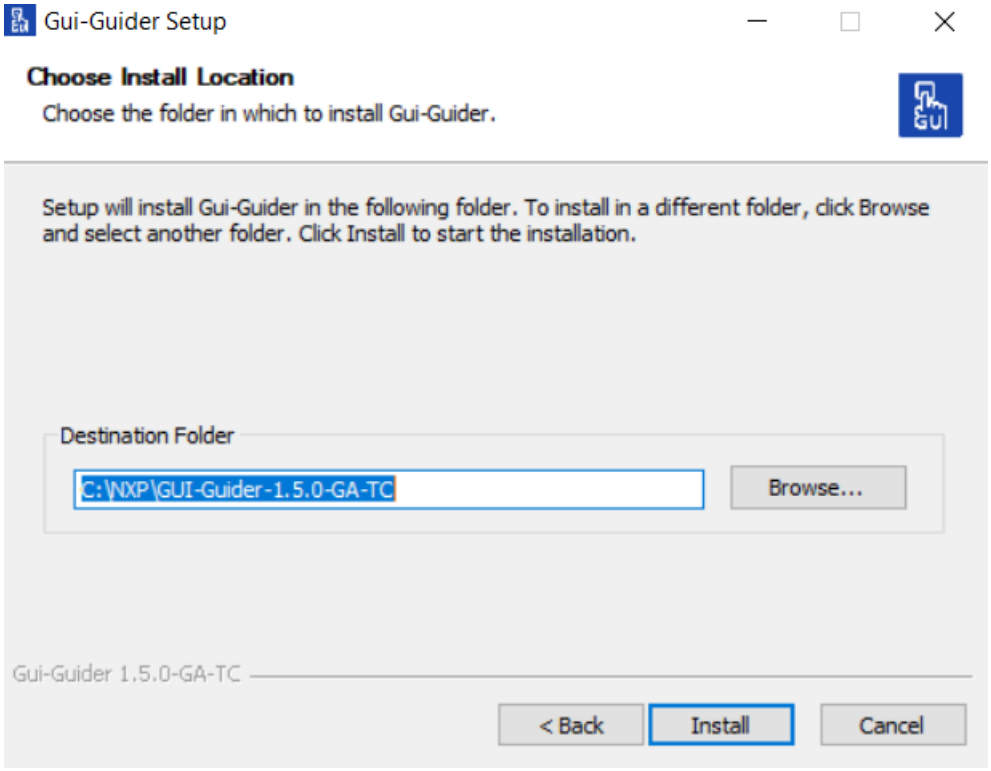

Figure 2. Choose install location

2.2.3 Ubuntu 22.04

To install the software, run the following command:

```
$ sudo apt install ./Gui-Guider-Setup-1.7.1-GA.deb
```

2.2.4 MacOS

To install GUI Guider on MacOS, perform the following steps.:

1. Install SDL2:
   - Download SDL source code:

```
wget https://github.com/libsdl-org/SDL/releases/download/release-2.26.5/
SDL2-2.26.5.tar.gz
```

```
tar -zvxf SDL2-2.26.5.tar.gz
cd SDL2-2.26.5
./configure --prefix=/usr/local
sudo make -j && sudo make install
```

2. Run `brew install cmake` command.
3. Install the MCUXpresso IDE.
   Note: *M core chip: ensure that the SDL2 is installed in* `/usr/local`.
4. Install GUI Guider on MacOS.
   a. Download the installer from www.nxp.com.
   b. Click the installer package and complete the installation based on the installation guide.

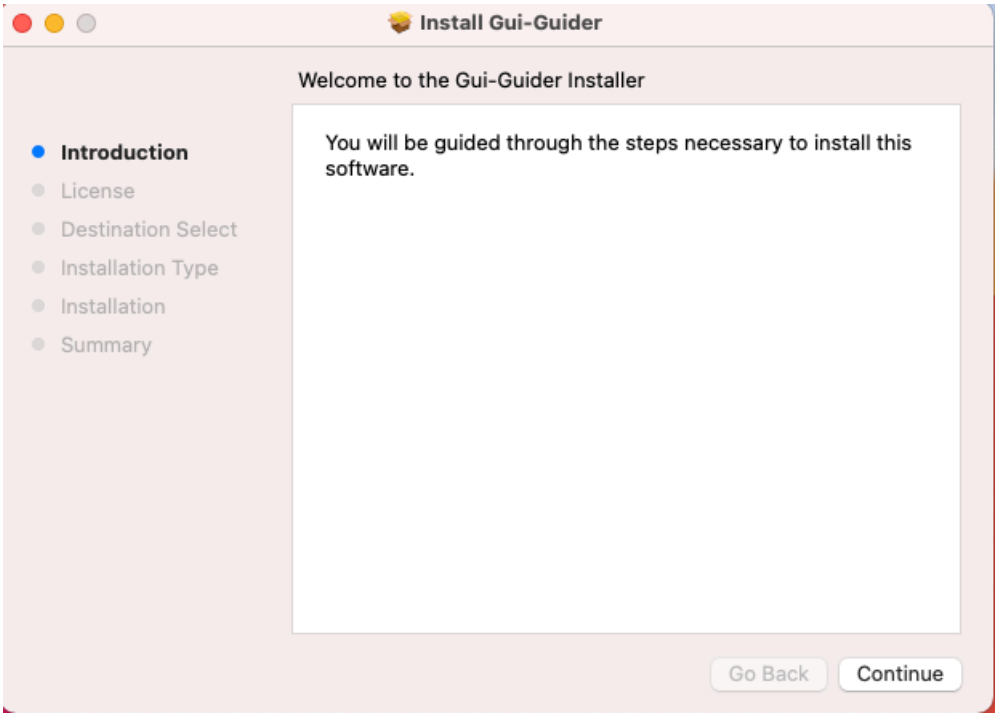

Figure 3. Install GUI Guider

2.2.5 Offline template

If you cannot connect to the network normally, install the offline templates as follows:

1. Download the corresponding version of the offline template package named `offline-template.zip` from the NXP official website.
2. Unzip the offline package to the GUI Guider installation path, for example: `C:\nxp\GUI-Guider-1.7.1-GA\environment`.
   Note: *The package must be placed under the environment directory, and ensure that the folder is named "offline-template".*

2.3 Quick start

You do not need anything else to start working but to install the GUI Guider. Here, you can choose to create a project with official examples or the local projects.

2.3.1 Create a project based on template

To create a project, perform the following steps:

1. Click the **Create a new project** button.

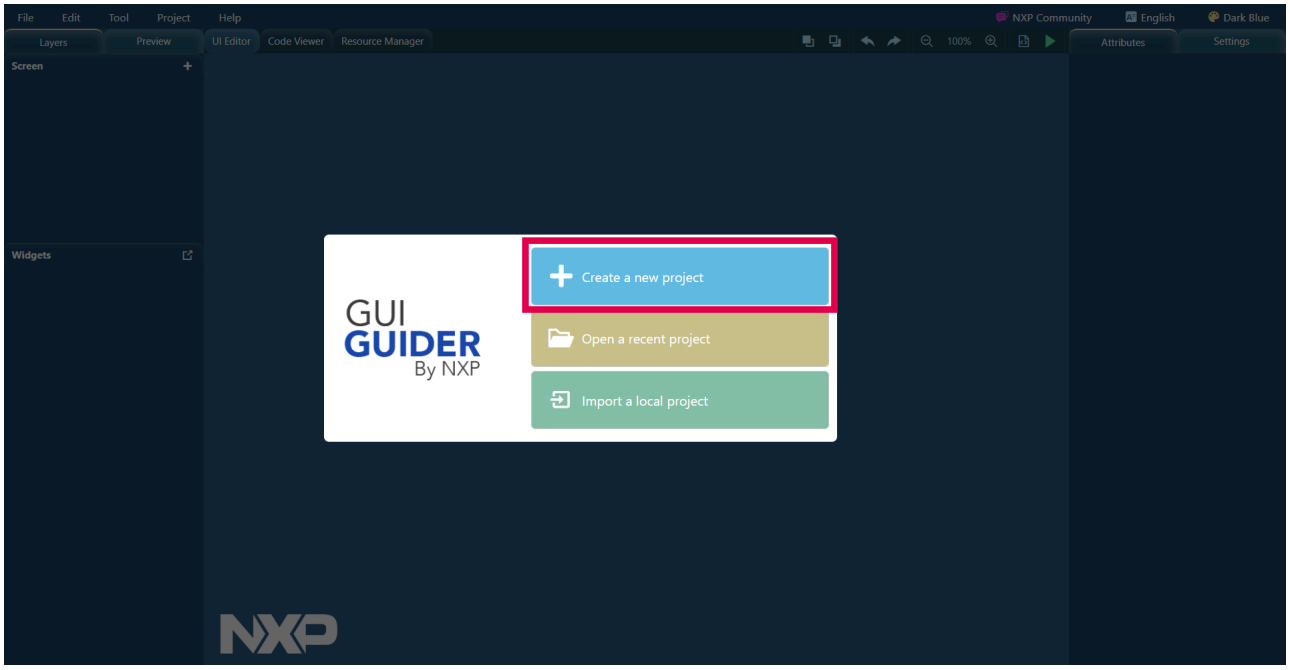

Figure 4. Create a project

Note: *Alternatively, you can select **File** > **New** in the GUI editor.*
*The **New Project** wizard dialog box appears.*

2. Select the LVGL version that you want to use. For example, v8.
3. Click **Next** or double click.
   The **Select a Board Template** page of the wizard appears.
4. Click the **Simulator**, **i.MXRT**, or **LPC** tab and select a board from the template list. For example, select **MIMXRT1050-EVKB**. Click **Next** or double click.
5. Click **Next** or double click.
   The **Select an Application Template** appears.
6. Select a GUI application template. For example, click the Widget tab and select **ScreenTransition**.

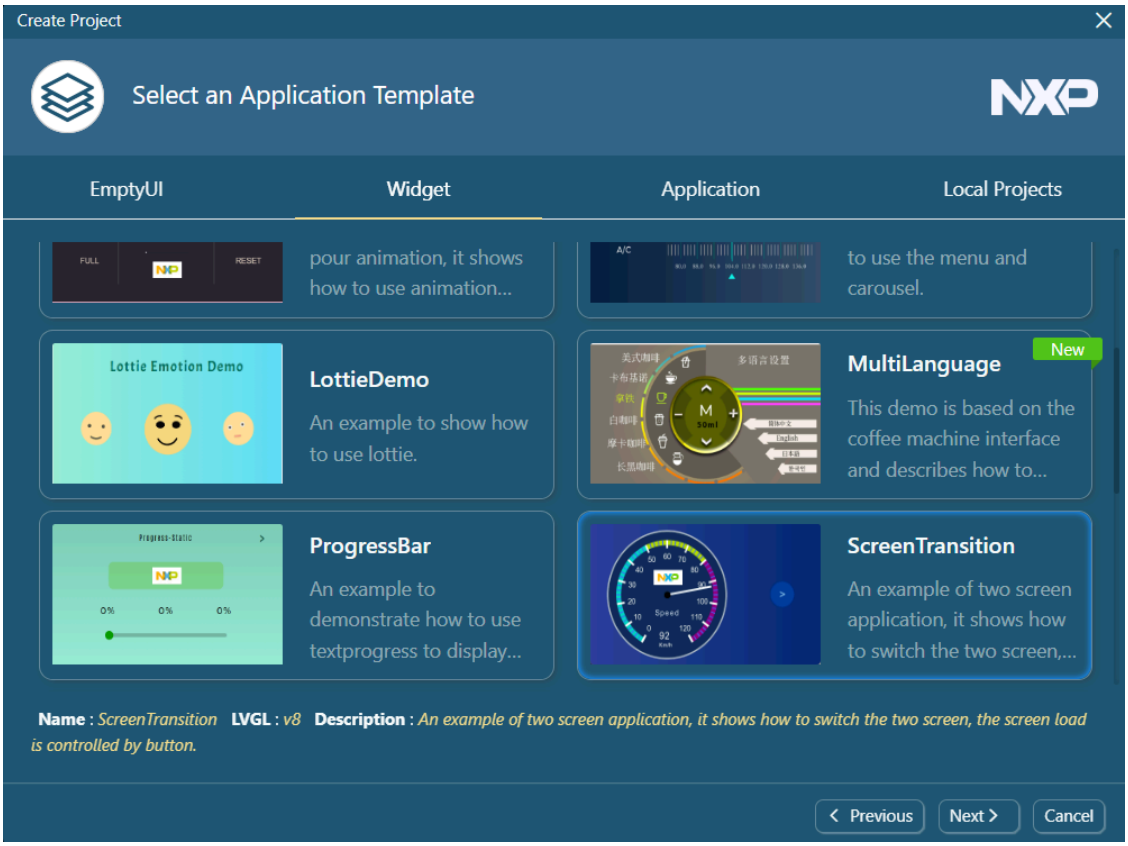

Figure 5. Select a GUI application template

7. Click **Next** or double click.
   The **Project Settings** page appears.
8. Configure the basic information of the project, including **Project Name**, **Project Directory**, **Panel Type**, **Color Depth**, and **GUI Resize**. Then click **Create**.
   Note: *To ensure that the GUI application is displayed normally on the board, select Auto Ratio. To customize the size of the application display, set the scaling ratio of width and height in the custom ratio.*

Figure 6. Set up project settings

Note: *GUI Guider supports multiple panel types for each board. The new panel is selected by default. Check the display type on your board and select the right panel type.*

2.3.2 Create a project based on local project

The function can support a new project based on a local project. The auto-scaling function is useful when you want to reuse an application design based on a particular display size. It is recommended that you create a project based on an existing local project as a template.

Note: *If there is a hard-coded size in the custom code, the position and the size-related code must be adjusted manually in the custom code.*

Figure 7. Customize panel size

2.3.3 Run simulator

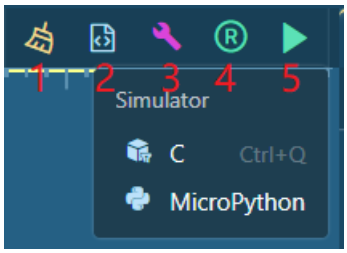

Figure 8. Run simulator

Table 6. Run simulator

label	Description
1	Clean the generated resource (image, font) files
2	Generate the application codes
3	Build the current project
4	Run the application
5	This button includes steps such as code generation, compilation, and running

Note: *If there are unused resource files generated in the design, you can click the clean button to clear them.*

3 IDE function

This section describes some key features of GUI Guider.

3.1 Project management

The chapter describes how to manage projects in GUI Guider, including creating a project, opening a recent project, importing a local project, and so on.

3.1.1 Create a new project

For detailed steps, see Section 2.3.

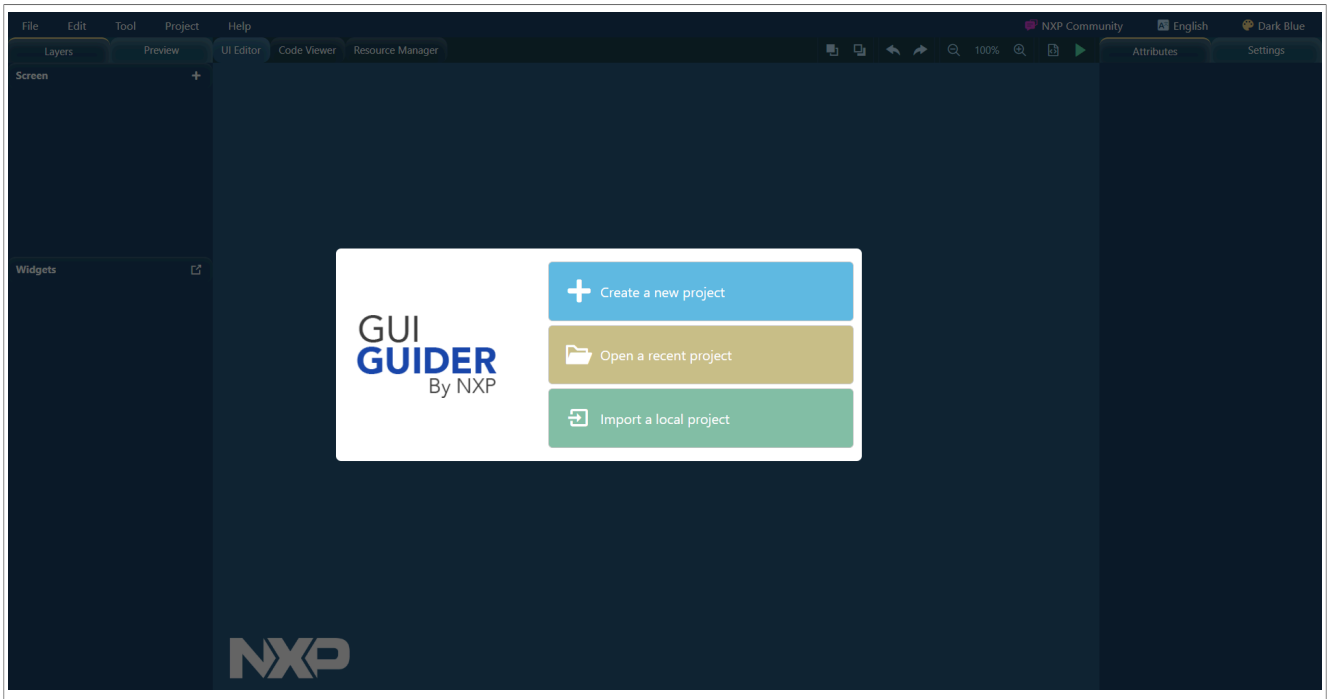

Figure 9. Create a project

3.1.2 Open a recent project

To open a recent project, perform the following steps:

1. Click the **Open a recent project** button.

Figure 10. Open a recent project

The **Recent Projects** dialog box appears with a list of existing projects.

Note: *Alternatively, you can select* ***File > Open*** *in the GUI editor.*

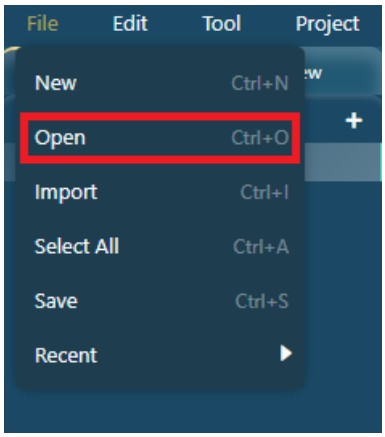

Figure 11. Alternative way to open a project

2. Select a project in the list.
   The selected project opens in the GUI editor.
   Note: *The projects of different LVGL version appear in the respective tabs.*

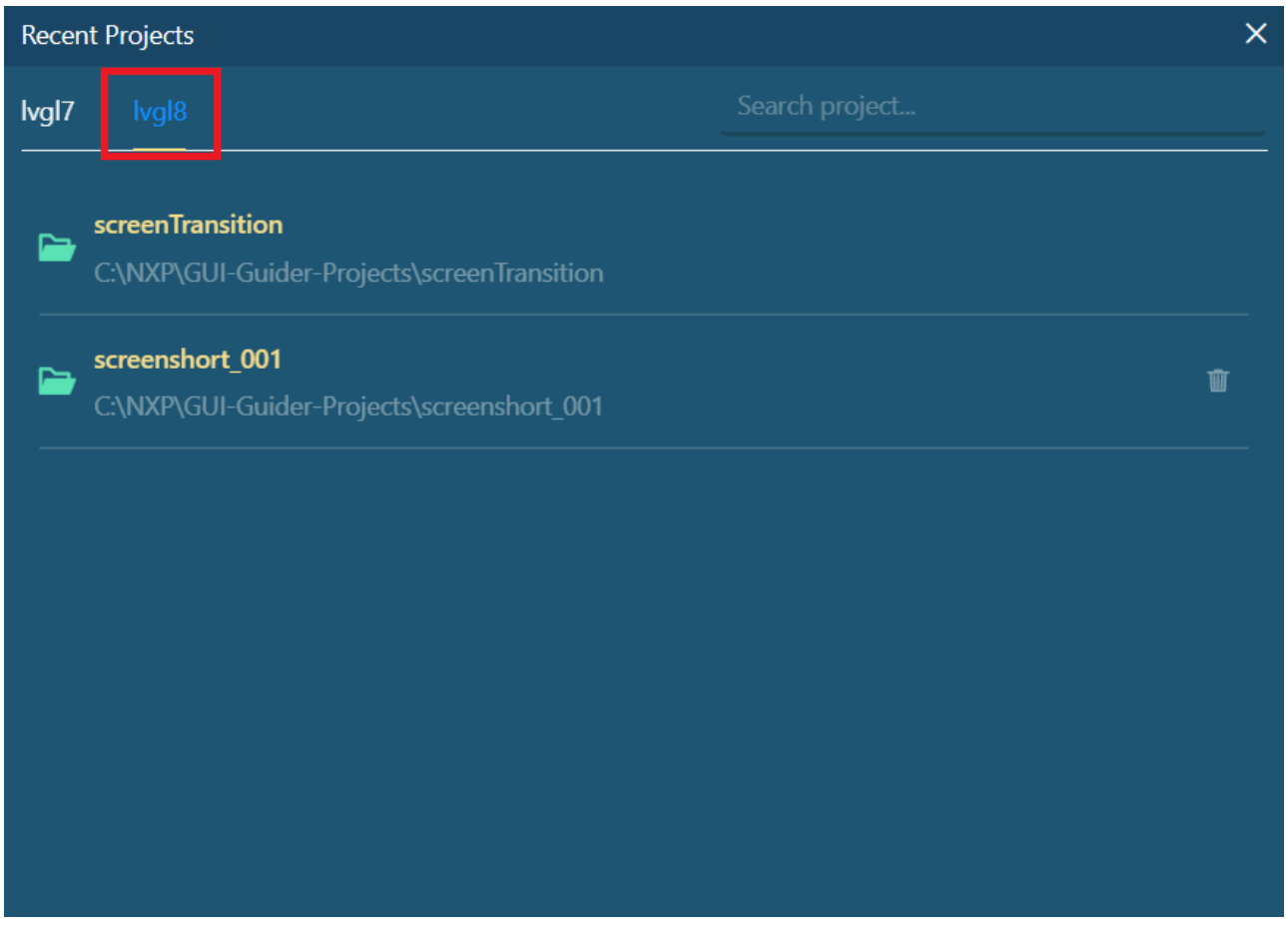

Figure 12. Select a project

3.1.3 Import a local project

To import an existing project, perform the following steps.

1. Click the **Import a local project** button.

Figure 13. Import a local project

Note: *Alternatively, you can select* ***File*** *>* ***Import*** *from the GUI editor.*

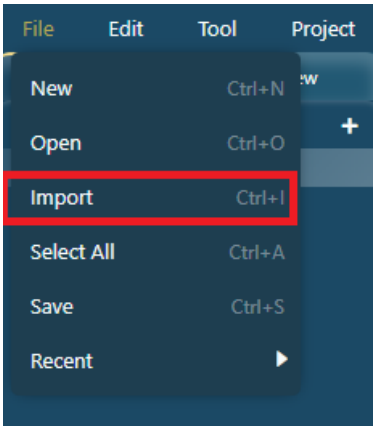

Figure 14. Alternative way to import a project

The **Choose Project** dialog box appears.

2. Navigate to the project that you want to import from your local directory.
3. Click **Open**.

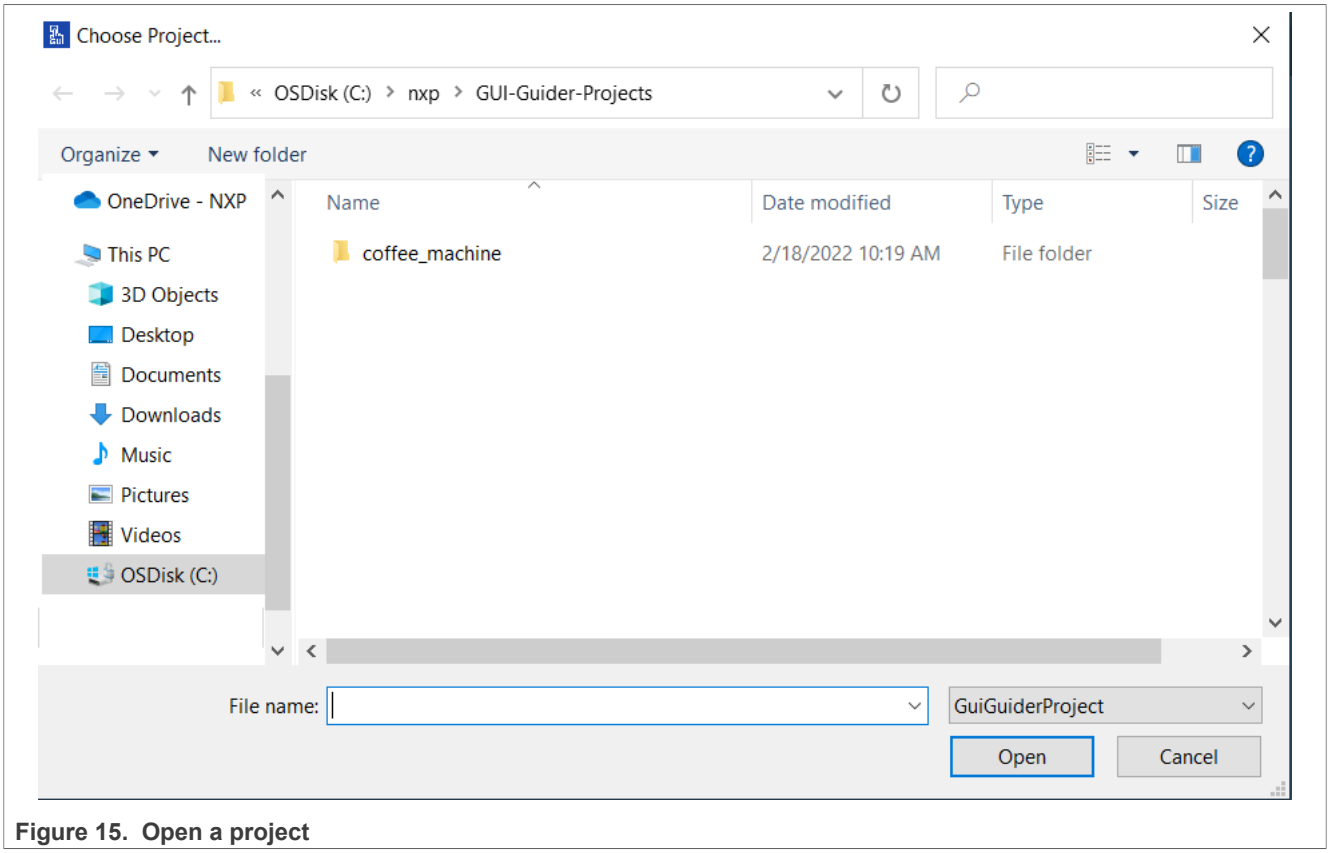

Figure 15. Open a project

The project is imported in the editor. However, if you try to import an older version of the project, a message prompts whether you want to update the project to match the current GUI Guider. Click **OK** to proceed. If you select "Backup", you can find the backup project zip file in the workspace.

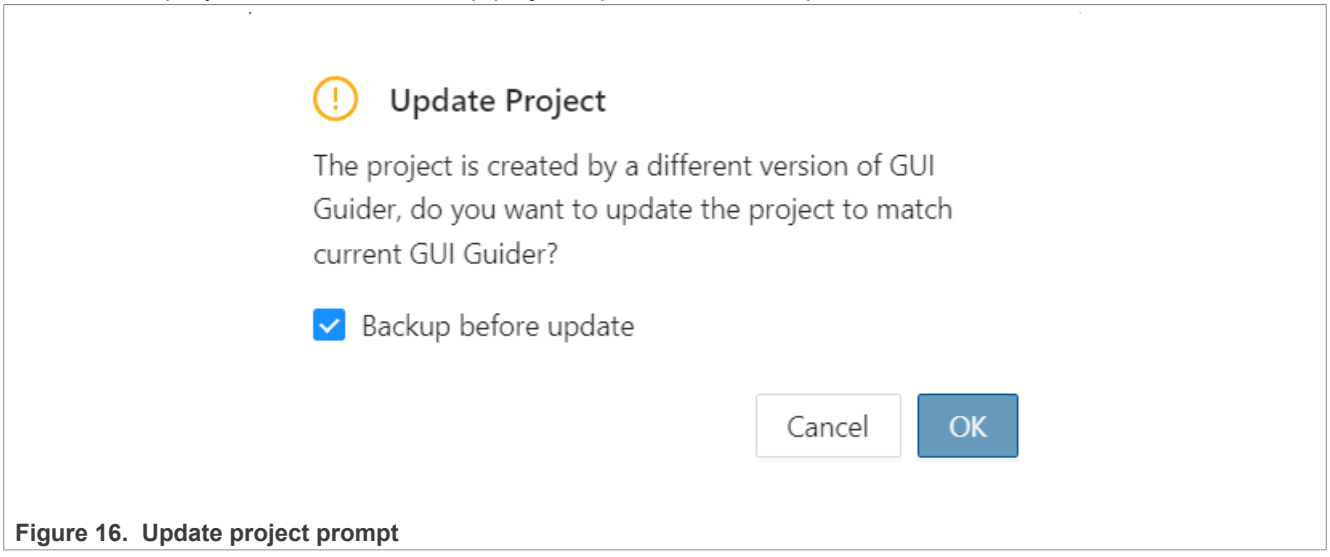

Figure 16. Update project prompt

3.1.4 Delete a project

To delete a project, perform the following steps:

1. Exit GUI Guider IDE.
2. Delete the project folder from local file system if the project is not needed.

3. Open the GUI Guider IDE.
4. Select the **Open a recent project** button.
5. Click the delete icon corresponding to the project you want to delete.

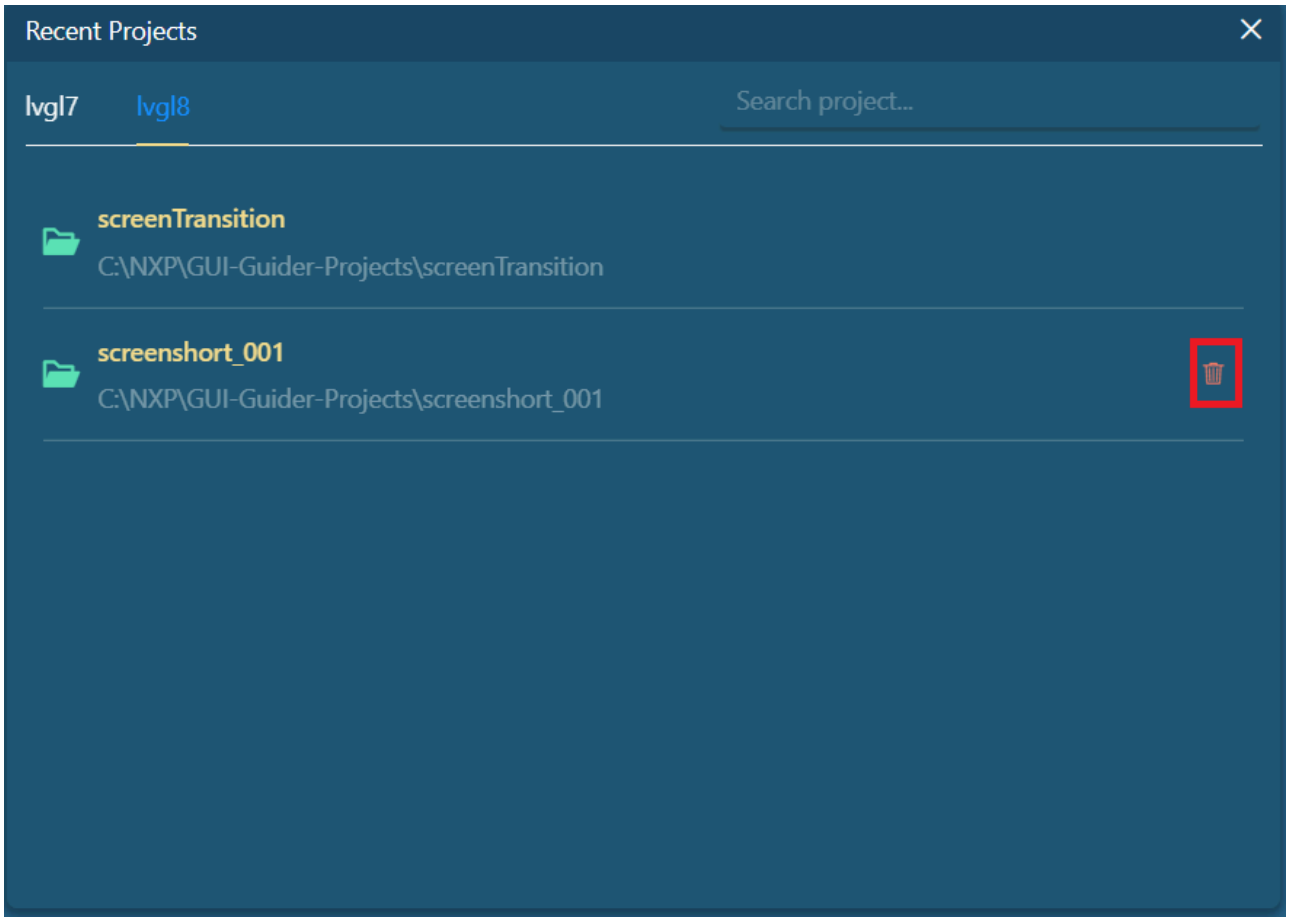

Figure 17. Click the delete icon

3.1.5 Upgrade project

From GUI Guider v1.6.0 release, we have added version control for the project upgrade.

- GUI Guider can only upgrade projects created by the last major version and related minor version.
  For example, GUI Guider v1.6.x can import project created by GUI Guider v1.5.x.
  A message box opens when GUI Guider v1.6.0 imports a project created by GUI Guider v1.4.1 or older version.

Figure 18. Update to latest version of GUI Guider

- GUI Guider cannot import projects created by newer GUI Guider version.
  For example, GUI Guider v1.5.1 cannot import project created by GUI Guider v1.6.0.
  A message box opens when GUI Guider imports a project created by a new version.

Figure 19. Update software version

3.1.6 Export project

To share the GUI Guider project more conveniently, we have added the export project function. The IDE remembers the export path which is a common path in the export code function.

To export the project, click **Project** > **Export project**.

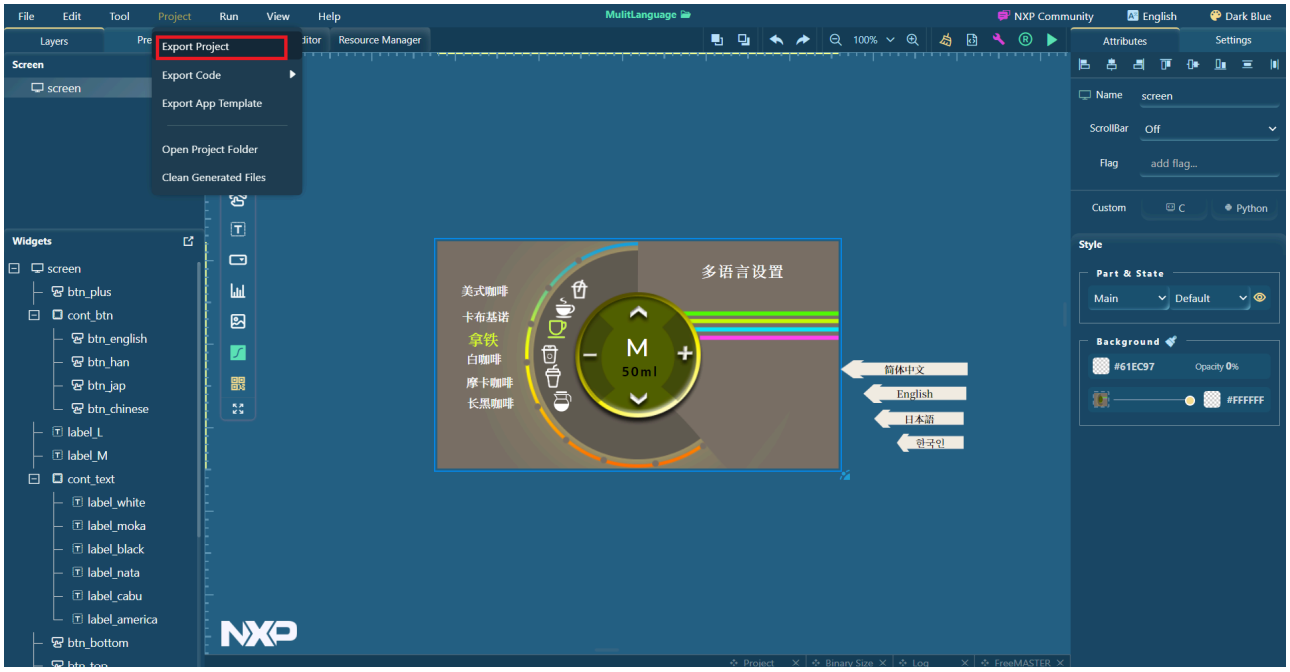

Figure 20. Exporting a project

The output of the export is the condensed project directory and a compressed file named <projectname>.zip. This file contains custom code, project resources, and UI configured files.

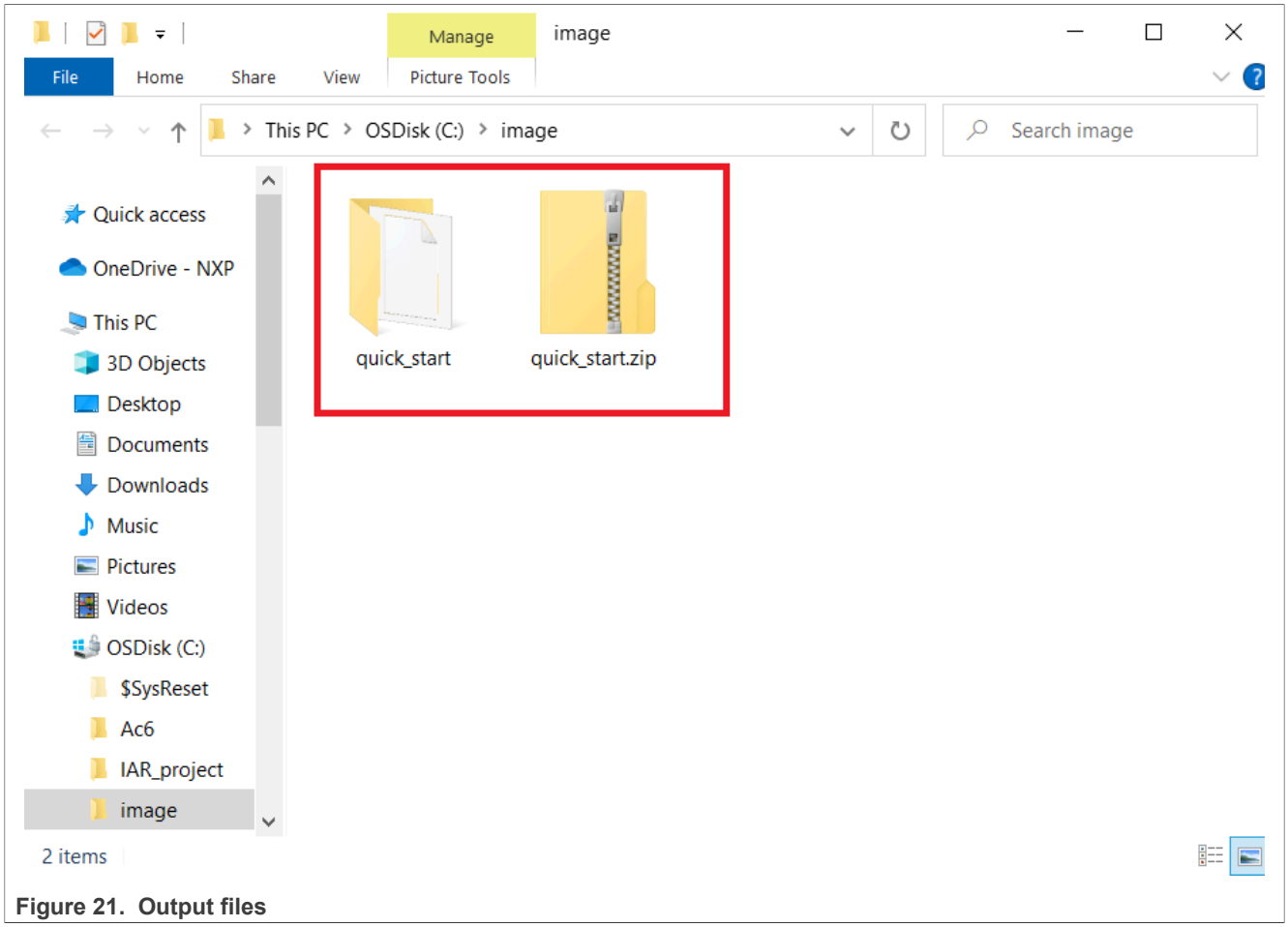

Figure 21. Output files

3.1.7 Export application template

We have added the export application function. The IDE remembers the export path, which is a common path in the export code function.

To export the project, click **Project** > **Export App Template**.

3.2 Project build and deploy

When an HMI application is designed, GUI Guider can generate the C and MicroPython source code. The application can be debugged in the simulator and target. GUI Guider can compile and deploy the HMI application in the simulator and target board.

3.2.1 Generated code

To generate the source code of a GUI project, click the  icon in the upper right of the edit window. Then, it can choose to generate C or MicroPython code.

Figure 22. Generating code

3.2.2 Run simulator

Both the C simulator and the MicroPython simulator are supported. To select a simulator and run it in the GUI application, click the  icon.

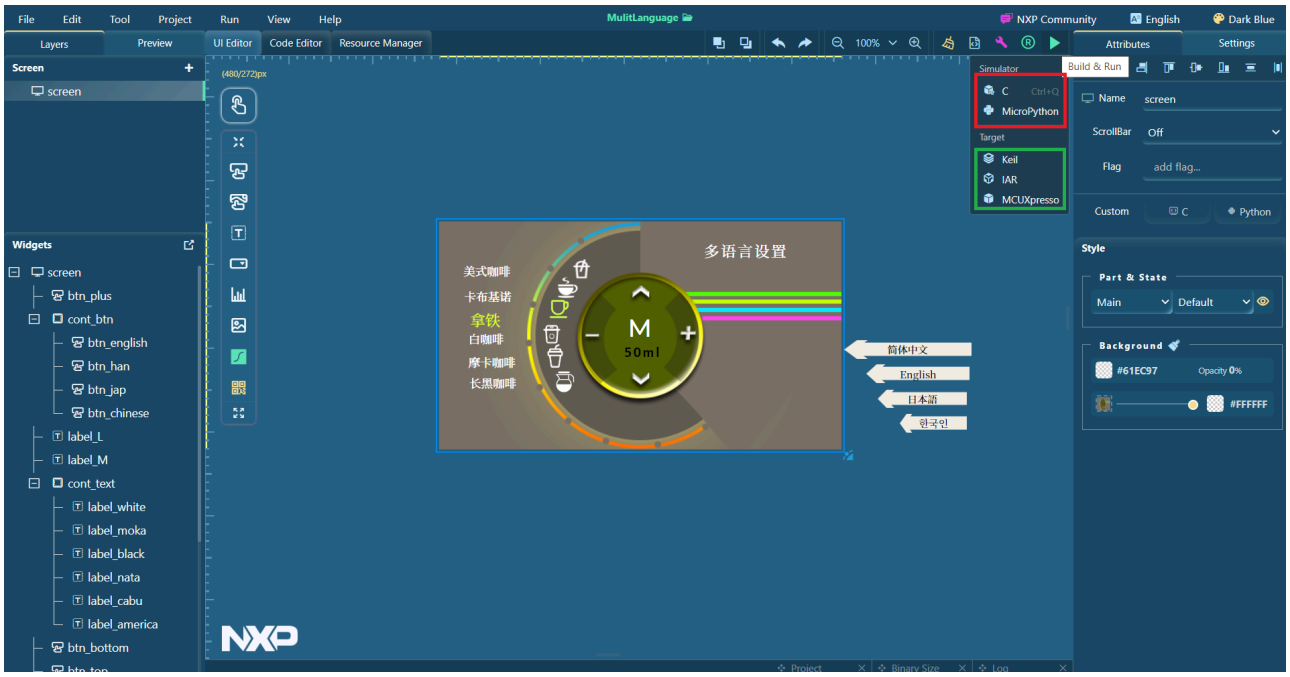

Figure 23. Running simulator

The simulator opens in a separate window.

Note: *When the simulator is launched, the* ***Generate Code****,* ***Run Simulator****, and* ***Run Target*** *options are disabled until the simulator window is closed. You can use the mouse or the keyboard to interact with the GUI elements in the simulator.*

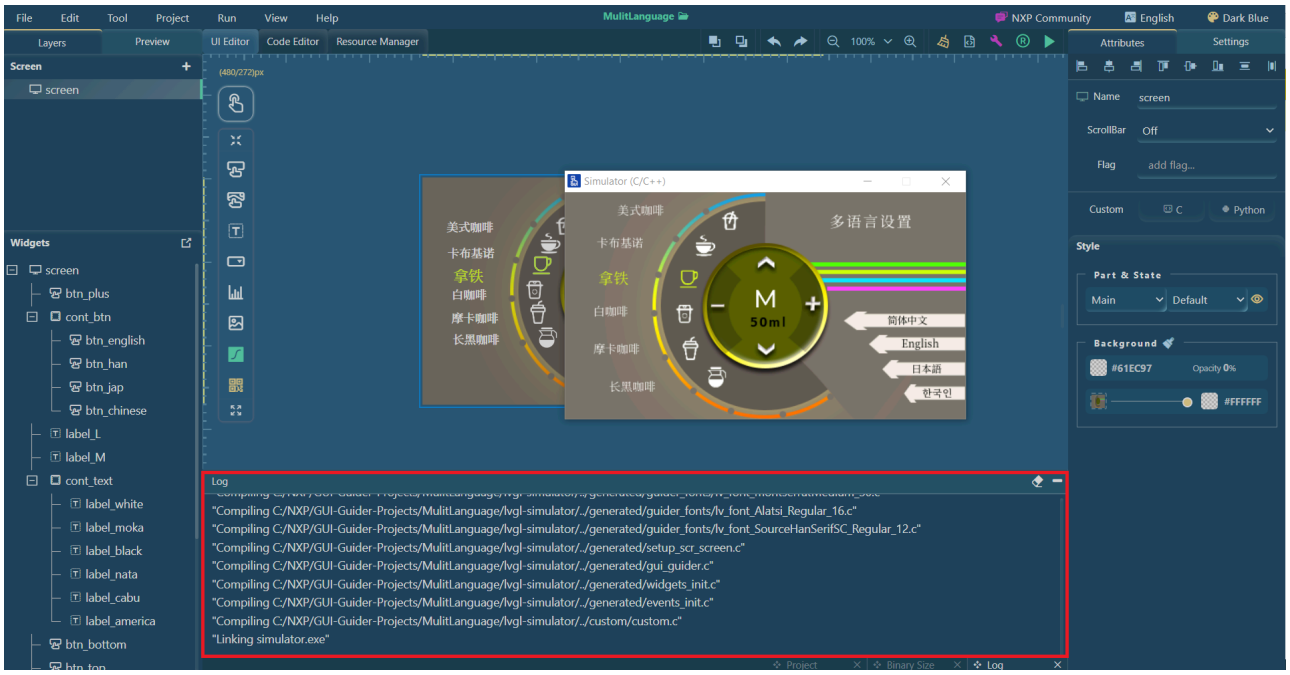

Figure 24. Interacting with GUI elements in simulator

Note: *The GUI Guider main window changes to modal state when* ***Run simulator*** *is clicked. MicroPython is not supported for LVGL v7.*

3.2.3 Run target

GUI Guider supports one-key build and deploy image on target board. GUI Guider also supports three toolchains: MCUXpresso, IAR, and Keil. Ensure that the corresponding IDE is installed on your host machine. Table 7 provides information on the supported toolchain.

Table 7. Supported toolchain

Toolchain	Version	Support OS	Connector
IAR	9.51	Win10	USB
MCUXpresso IDE	11.9.0	Win10, OSX11, and Ubuntu 22.04	USB
Keil MDK	5.38	Win10	USB

The following prerequisites must be met to run the target successfully:

- Boards with CMSIS-DAP/mbed/DAPLink interface.
- For LPCXpresso boards, install the DFU jumper for the debug probe.
- Connect the development platform to your PC via USB cable.
  Figure 25 shows the window of log, project information, and memory monitor.

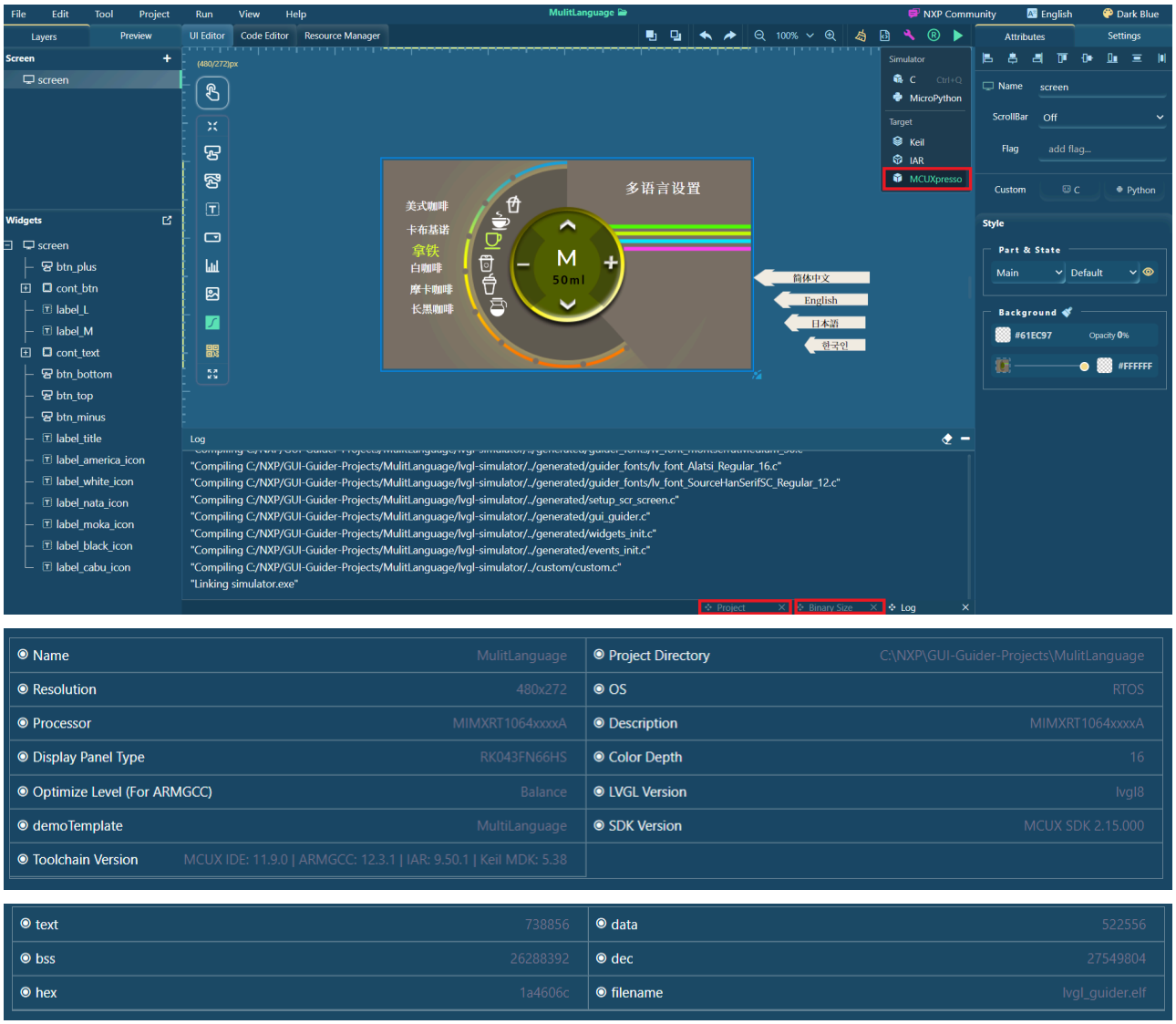

Figure 25. Project log, information, and memory monitor

Typically:

- The flash consumed by the GUI application is text + data.
- The RAM consumed by the application is data + bss.

Note:

- *Only MCUXpresso IDE supports memory display.*
- *The project does not support "Run Target" when simulator is selected as board template.*

3.2.4 Edit code

The source code generated by GUI Guider appears in the **Code Editor** tab. The navigator is on the left side of the code editor and switches to the source file that you want to edit.

Files in the custom directory can only be edited.

3.3 Resource management

Currently supported resource types are: font, image, video, and Lottie_Json. Before you import a resource, make sure that its name does not contain spaces and special characters. The recommended nomenclature is: "aaa_bbb.json".

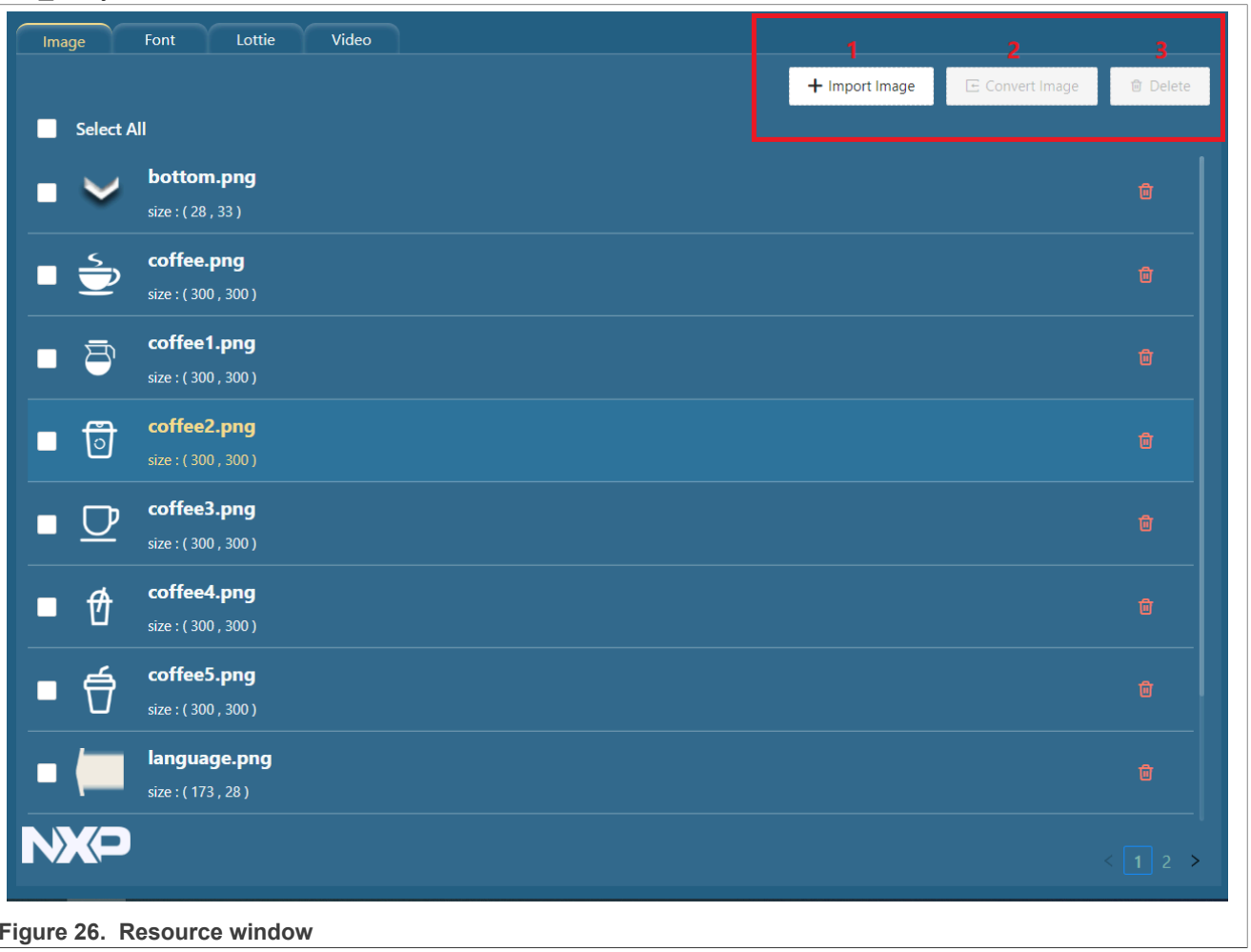

Figure 26. Resource window

Table 8. Resource management

Label	Description
1	Import image button
2	Convert images to bin file
3	Delete button

3.4 System setting

This section describes the settings of GUI Guider IDE, project, and `lv_conf.h`.

3.4.1 IDE setting

Table 9 lists the options in the **IDE setting** window.

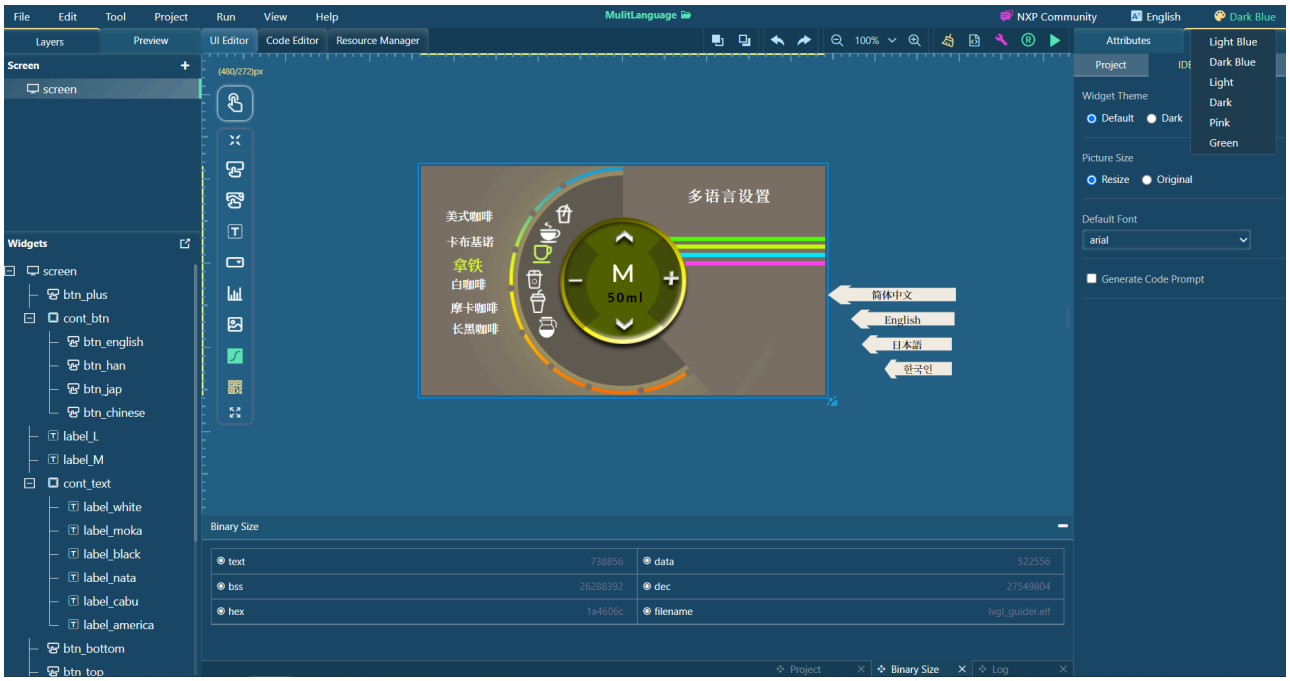

Figure 27. IDE setting

Table 9. IDE setting window

Item name	Option
Language	English and Chinese
Theme	Light Blue, Dark Blue, Light, Dark, Pink, and so on
Widget theme	Set the default widget theme, default or dark
Picture size	Select the image default size, resize by widget, or image original size
Default font	Configure the default font in the widgets attribute setting
Generate code prompt	Enable or disable the code override prompt when generating code

3.4.2 Project setting

Table 10 lists the options in the **Project setting** window.

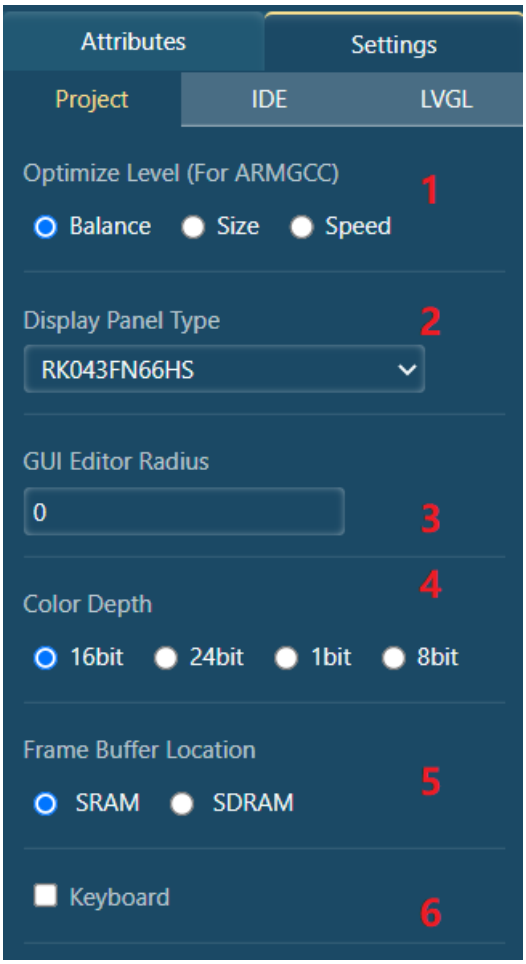

Figure 28. Project setting

Table 10. Project setting window

Label	Description
1	Select optimization level (Only support NXP target)
2	Screen selection of the same resolution
3	Change the GUI editor radius
4	Select color depth for display
5	Select frame buffer region (MIMXRT1040-EVK, MIMXRT1050-EVKB, MIMXRT1060-EVK, MIMXRT1060-EVKB, MIMXRT1060-EVKC, and MIMXRT595-EVK, support)
6	Enable the keyboard and set the style

3.4.3 LVGL

Table 11 lists the options in the LVGL setting window.

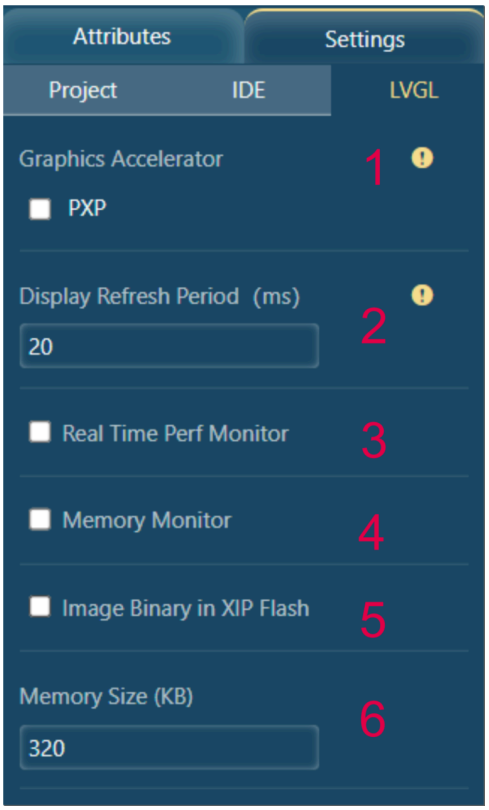

Figure 29. LVGL setting window

Table 11. LVGL setting window

Label	Description
1	Graphics accelerator
2	Set the default display refresh period
3	Enable the real-time performance monitoring
4	Enable the real-time memory monitor
5	Set the image binary download base address
6	Set the heap size allocated for LVGL application usage

3.5 Key menu function

This section describes the key menu function.

3.5.1 Generate custom fonts

The generated font file is stored in the `<project_name>\generated\guider_customer_fonts` folder. The purpose is to add new characters, which otherwise are supported by the selected font type and size. The function is used for non-English languages. For example, Chinese.

To generate fonts, perform the following steps:

1. Select **Tool** > **Generate Fonts**.
   The **Generate Font** dialog box appears.
2. Select the font family and size. Ensure that the font family is for the English language.

Figure 30 is an example of fonts generated for Chinese language.

Figure 30. Select font and size

3. Click **OK**.
   The newly generated font appears normal in the GUI application.

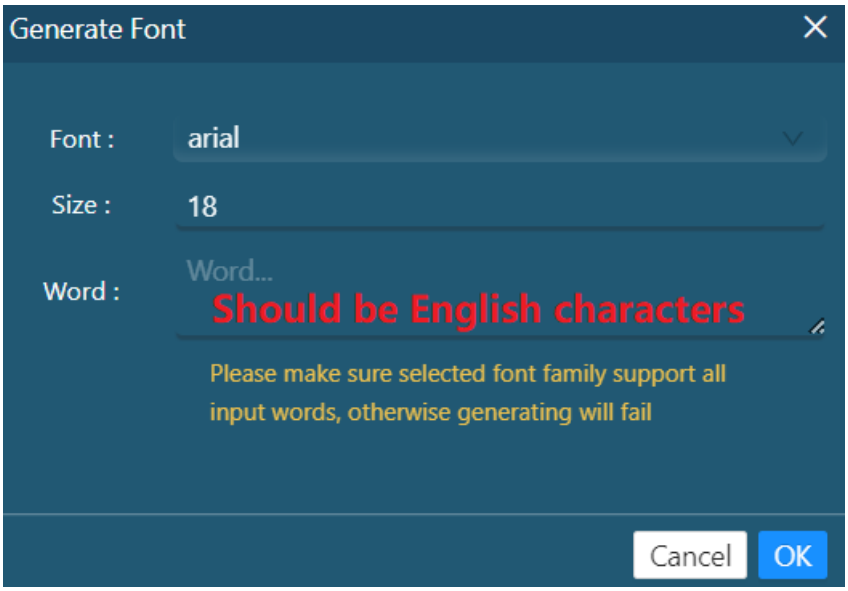

Figure 31. Generate your fonts

The function provides an API to convert fonts to a C array. The C array file is generated in the `generated\guider_customer_fonts` folder.

The following is the example code of using the generated font:

```
#include "lv_font.h"
LV_FONT_DECLARE(lv_customer_font_SourceHanSerif_VF_18)
lv_style_set_text_font(&style_screen_ddlist1_selected,
```

```
LV_STATE_DEFAULT, &lv_customer_font_SourceHanSerif_VF_18);
```

3.5.2 Convert images

In the resource manager window, you can use the Convert button to convert images to bin or C array files.

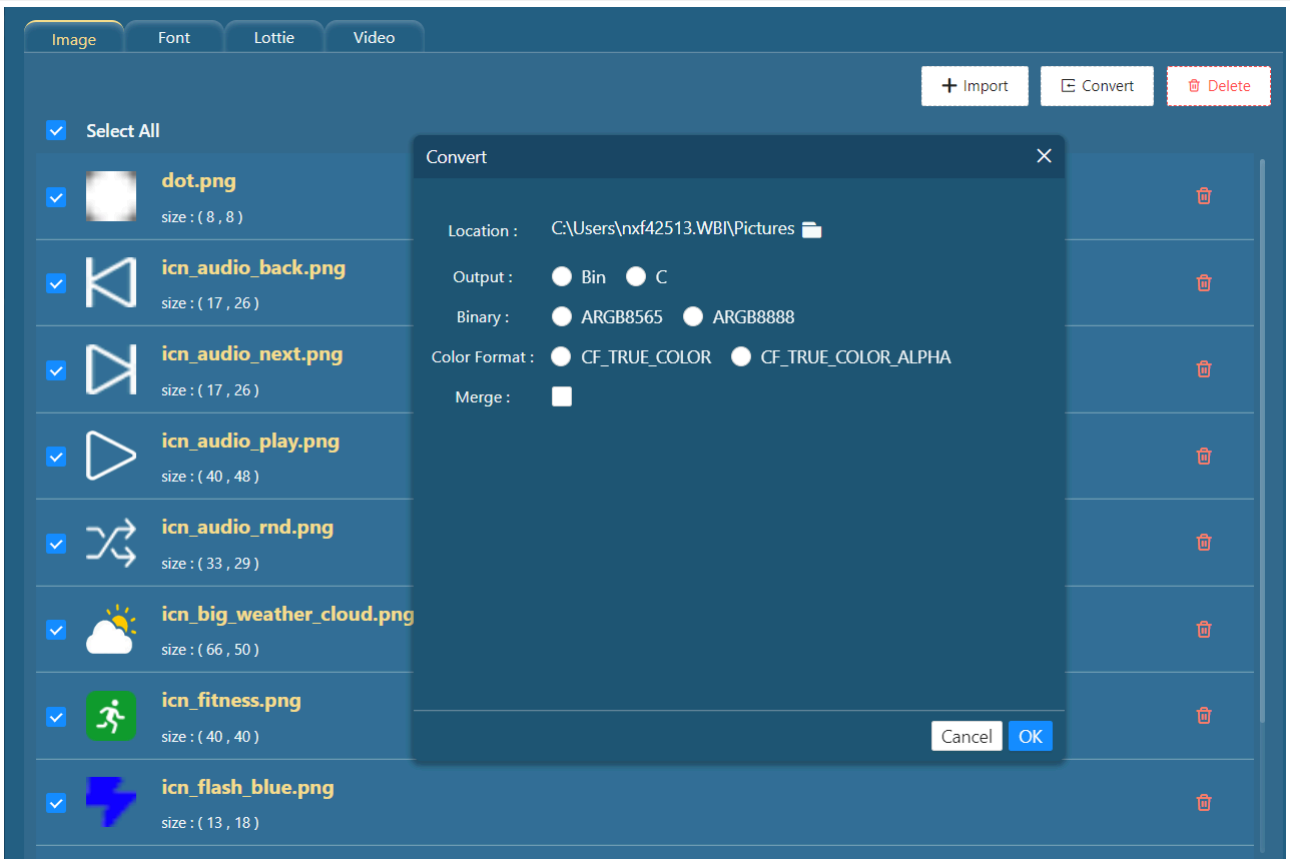

Figure 32. Resource manager window

Table 12. Convert images window

Item name	Description
Location	Select the export file path
Output	Select the output file type
Binary	Select the binary format type
Color format	Choose the color format
Merge	If the output is a bin, you can merge it into one file

If the output is bin, you get two files: `mergeBinFile.bin`, `mergeBinFile.c`. The C file defines the size and address of the bin file corresponding to each image.

```
#include "lvgl.h"

#if LV_USE_FS_RAWFS

const rawfs_size_t rawfs_file_count = 10;
rawfs_file_t rawfs_files[10] = {
    0x0, 0, 132, "/dot.bin",
    0x84, 0, 888, "/icn_audio_back.bin",
    0x3fc, 0, 888, "/icn_audio_next.bin",
    0x774, 0, 3844, "/icn_audio_play.bin",
    0x1678, 0, 1918, "/icn_audio_rnd.bin",
    0x1df6, 0, 6604, "/icn_big_weather_cloud.bin",
    0x37c2, 0, 3204, "/icn_fitness.bin",
    0x4446, 0, 472, "/icn_flash_blue.bin",
    0x461e, 0, 1348, "/icn_heart.bin",
    0x4b62, 0, 1200, "/icn_home.bin",

};

#endif  /*LV_USE_FS_RAWFS*/
```

Figure 33. mergeBin file

3.5.3 Convert video

Sometimes the user want to play a short video without storing the video source file in the target. For this purpose, we provide a way to use an animation widget to play the picture of each frame. To convert the video file into the desired image collection, perform the following steps:

1. Import the video file. Currently, it only supports MP4 files.

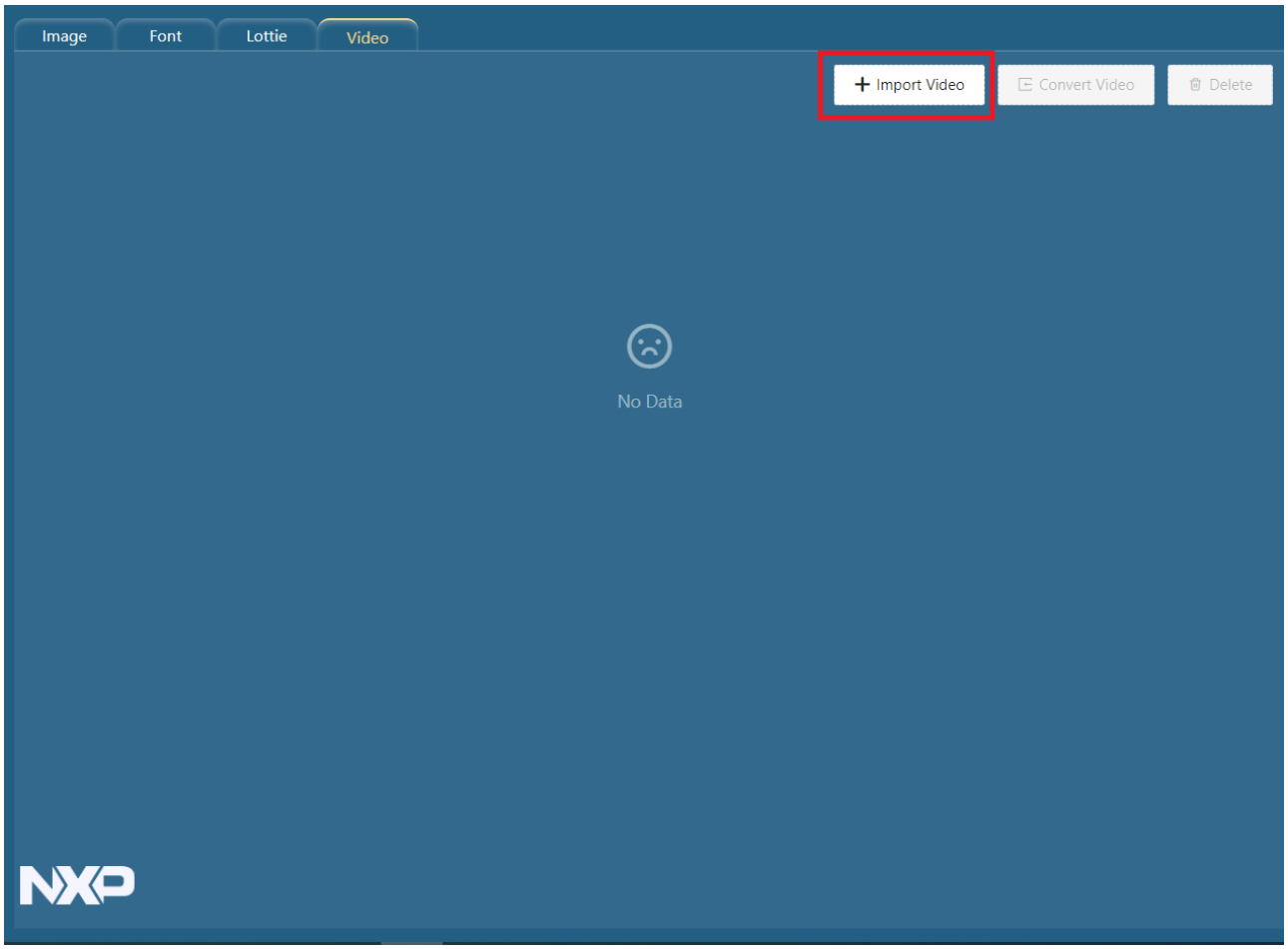

Figure 34. Import video file

2. Select the video and click the "Convert Video" button.

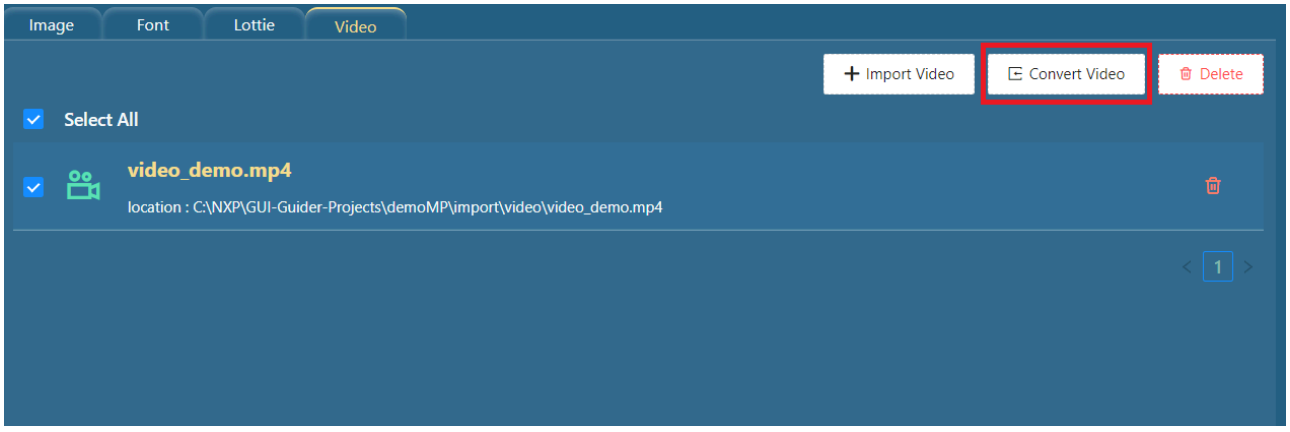

Figure 35. Save video

3. Set conversion configuration information as needed.

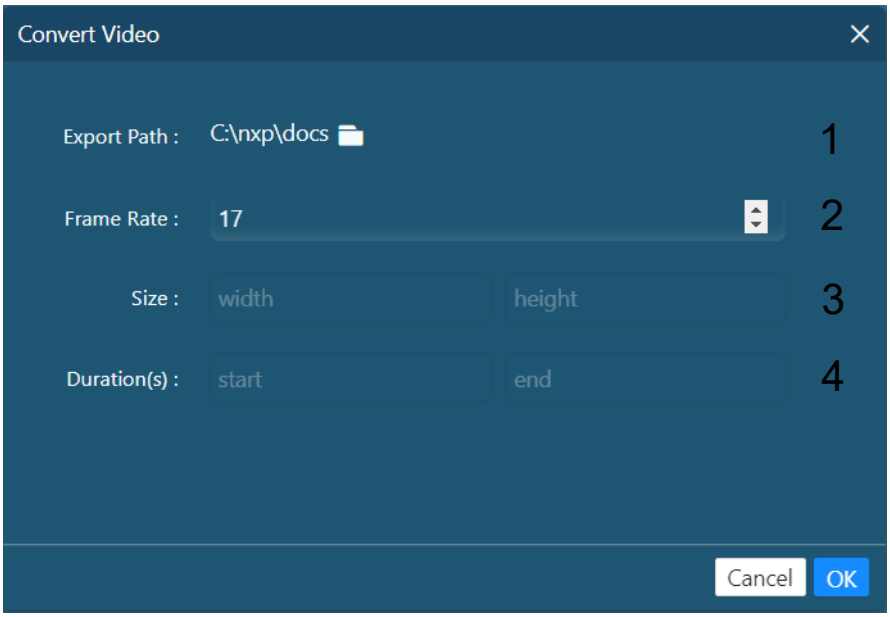

Figure 36. Set conversion configuration

Table 13. Convert video

Item name	Description
Export path	Select the converted image exported path
Frame Rate	Set the frame rate refresh per second during conversion
Size	Set converted image size
Duration	Select the start and end time of the video to be converted

Note: *The conversion process takes time proportional to the time and frame rate of setting the conversion.*

3.5.4 Widget distribution

Figure 37 shows the widget distribution.

Figure 37. Widget distribution

The uniform distribution is calculated according to the distance between the first widget and the last one.

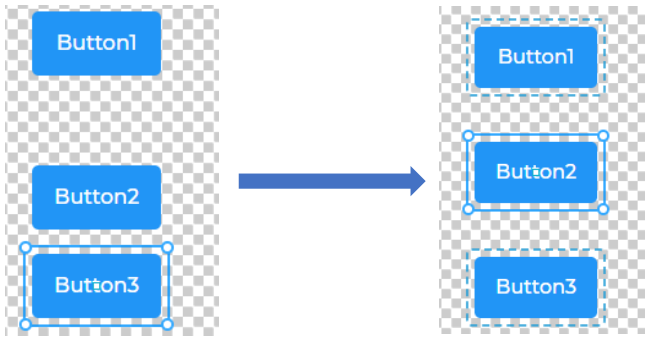

Figure 38. Vertical distribution

Note: *If no widget is selected for distribution, the default behavior is to distribute the entire editor. If you want to adjust the editor position, you can use this function.*

3.6 Shortcut function

Table 14 lists the keyboard shortcuts supported by GUI Guider.

Table 14. Shortcut function

Function name	Shortcut
New project	Ctrl + N
Open project	Ctrl + O
Import project	Ctrl + I
Select All	Ctrl + A
Save	Ctrl + S
Copy	Ctrl + C
Paste	Ctrl + V
Delete	Del
Undo	Ctrl + Z
Redo	Ctrl + Y
Generate C code	Ctrl + G
Build & Run C simulator	Ctrl + Q
Add event	Ctrl + E
Only build C	Ctrl + B
Only Build Target	Shift + B
Only Run C simulator	Ctrl + U
Build & download target	Shift + U

4 Widget details

This chapter introduces the details of supported widgets, including attributes, styles, and events.

4.1 Attribute

GUI Guider supports configuring widget attributes in the IDE directly. The following sub-chapters describe attributes for all supported widgets.

4.1.1 Screen

The screen is the base object that defines the application of the workspace, and widgets can be added to it.

The attributes specific to the screen are as follows:

- Custom code: You can add the custom style or logic code for the current screen.

Note: *You can right-click the icon*  *in the lower right corner of the screen and drag the mouse to change the GUI editor radius. The effect is only displayed for the editor, and the simulator does not have this realistic effect.*

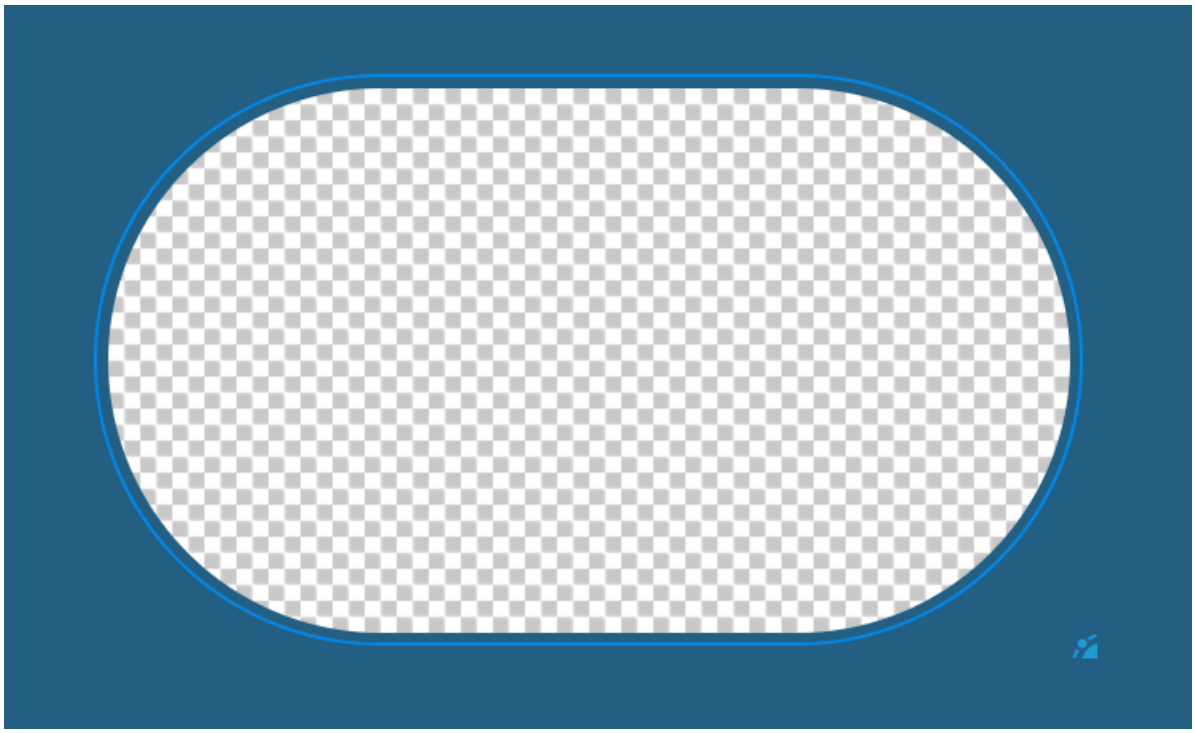

Figure 39. Screen

4.1.2 Button

The buttons are simple rectangle-like objects. They can be enabled to automatically transition to the checked state on a click.

The attributes specific to the button are as follows:

Text：Add the button or the lvgl symbol.

Toggle: Enabling the toggle means that the "checked" state remains when the button is clicked once. The button must be clicked again to get back to the initial state. You can select the "checked" state to update the checkable style.

Figure 40. Button

4.1.3 Image button

The Image button is similar to the simple "Button" object. The only difference is that it displays user-defined images in each state instead of drawing a rectangle.

The attributes specific to the image button are as follows:

- Format: The color format setting.
  You can select "true color" or "true color alpha". The "true color" is for RGB image and "true color alpha" is for ARGB image.
- Image selected: You can set the following state in which you want to show the image.
  Released, Pressed, Checked Released, Checked Pressed

Figure 41. Image button

4.1.4 Checkbox

Checkbox objects are built from a button background, which contains a button bullet and a label to realize a classical checkbox.

The attributes specific to the checkbox are as follows:

- Text: The title of the checkbox.

Figure 42. Checkbox

4.1.5 Button matrix

The button matrix objects can display multiple buttons in rows and columns. The main reasons of using a button matrix instead of a container and individual button objects are as follows:

- The button matrix is simpler to use for grid-based button layouts.
- The button matrix consumes less memory per button.

Figure 43. Button matrix

4.1.6 Switch

The switch can be used to turn on/off something. It looks like a little slider.

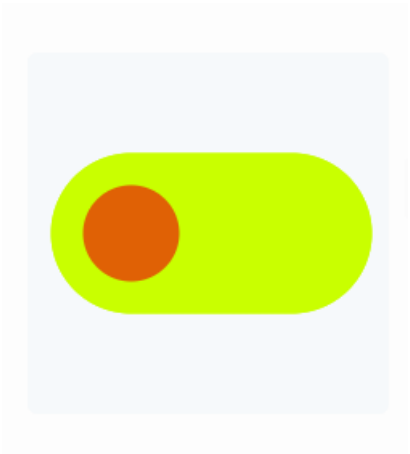

Figure 44. Switch

4.1.7 Label

A label is the basic object type that is used to display text on the screen.

The attributes specific to the label are as follows:

- Text: Add the text or the lvgl symbol to be displayed.
- Label mode: You can choose different solutions for long text.
  - Circular: If the text is wider than the label, scroll it horizontally. If it is higher, scroll vertically.
    Note: *Only one direction is scrolled and horizontal scrolling has a higher precedence.*
  - Clip: Clip the end of the text if the text is wider than the label.
  - Dot: Replaces the last 3 characters from the bottom-right corner of the label with dots.
  - Scroll: If the text is wider than the label, scroll it horizontally back and forth. If it is higher, scroll vertically.
    Note: *Only one direction is scrolled and horizontal scrolling has a higher precedence.*
  - Wrap: Wrap long text.

Figure 45. Label

4.1.8 Spangroup

A spangroup is the object that is used to display rich text. Different from the label object, it can render text style with different fonts, colors, and sizes into the spangroup object.

The attributes specific to the spangroup are as follows:

- Mode: The spangroup can be set to one of the following modes:
  - Fixed: Fixes the object size.
  - Expand: Expands the object size to the text size but stays on a single line.
  - Break: Keeps width, breaks the too long lines, and auto expands height.
- Item: Click each item to set the item parameters, including font color, font size, underline, strikethrough, and so on.

Figure 46. Spangroup

4.1.9 Drop-down

Drop-down is a list that allows the user to select one value from a list. The drop-down list is closed by default and displays a single value or a predefined text. When activated (by clicking the drop-down list), a list is created from which the user can select one option. When the user selects a new value, the list is deleted again and displays only the selected value. The drop-down list is added to the default group (if it is set). Besides, the drop-down list is an editable object to allow selecting an option with encoder navigation too.

- Arrow – If it is disabled, the expand arrow icon is removed.

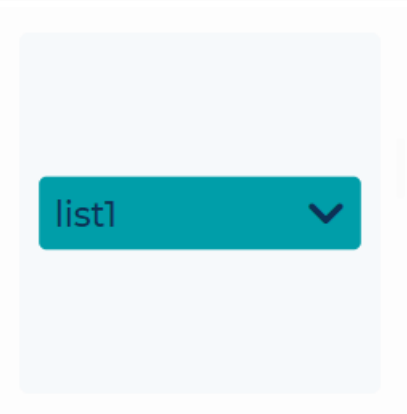

Figure 47. Drop-down

4.1.10 Text area

A text area is a basic object with a label and a cursor on it. Texts or characters can be added to it. Long lines are wrapped and when the text becomes long enough, the text area can be scrolled.

The attributes specific to the text area are as follows:

- Text: The default text display in the text area.
- Password mode: Text area works like a password area; characters are replaced by asterisks.
- One line mode: The text area can be configured to be a single line. In this mode, the height is set to show automatically only one line. Line break characters are ignored and word wrapping is disabled.

Note: *You can enable and set one keyboard on the current screen setting. The text area triggers the keyboard when the current keyboard is enabled.*

Figure 48. Text area

4.1.11 Calender

A calendar widget shows the days of any month, name of days, and highlights today and any user-defined dates in a 7x7 matrix. The default date is the real date.

4.1.12 Table

Tables, as usual, are built from rows, columns, and cells containing texts. The table object is lightweight because only the texts are stored. No real objects are created for cells but they are drawn on the fly. The table is added to the default group (if it is set). Besides, the table is an editable object to allow selecting a cell with encoder navigation.

Note: *The height is calculated automatically from the cell styles (font, padding, and so on) and the number of rows.*

Type	Board
LPC	LPC54S018
KW	KW45B41Z
MCX	MCXN947

Figure 49. Table

4.1.13 Tab view

The tab view object can be used to organize the content in tabs. The tab buttons can be positioned on the top, bottom, left, and right sides of the tab view. A new tab can be selected either by clicking a tab button or by sliding horizontally on the content.

The attributes specific to the tab view are as follows:

- Size: It is the value of the tab button height.
- Position: Selects on which side to place the tab buttons.
- Page: You can click "+" to add new pages. Each page support setting the name and text.

Figure 50. Tab view

4.1.14 Message box

A message box acts as a pop-up. They are built from a background container, a title, an optional close button, a text, and optional buttons. The text is broken into multiple lines automatically and the height is set automatically to include the text and the buttons.

The attributes specific to the message box are as follows:

- Title: The message box title name; It can be empty.
- Text: The message box content text; It cannot be empty, otherwise, an exception occurs.
- Btn size (width, height): The buttons are built in the message box widget. So, the button size must be set on the attribute tab.
- Close button: Selects whether to show the close button.
- Button: You can click "+" to add new buttons.

Figure 51. Message box

4.1.15 Container

A container widget is a basic object. By using it, you can create a rectangle, which can be transformed freely with styles. You can add background color, border, padding, and shadows.

4.1.16 Chart

A chart is a basic object to visualize data points. Currently line charts (connect points with lines and/or draw points on them) and bar charts are supported.

The attributes specific to the chart are as follows:

- Line: Sets the grid lines horizontal and vertical number. If you do not want to set the grid lines, set them to 0.
- Zoom: The chart can be zoomed in independently in X and Y directions. The factor, 256, means that there is no zoom and 512 means double zoom. Fractional values are also possible but < 26 values are not allowed.
- Type: The data display types are as follows:
  - Line: Draws lines between the data points and/or points (rectangles and circles) on the data points. You can hide the data points by enabling the hide switch.
  - Bar: Draws bars.
  - None: Do not display any data. It can be used to hide the series.
- Left Y/Right Y/Bottom X/Top X: Ticks and labels can be added to the axis in each direction.
  - Range: Sets the ticks range; Min and max value.
  - Enable tick: Enables the tick to see the tick settings.
    - Enable label: Adds the label to the axis.
      - Draw size: Extra size required to draw the tick and labels (start with 20 px and increase if the ticks/labels are clipped).
  - Length: You can set the length of the major and minor ticks.
  - Count: Sets the number of the major ticks on the axis, and the number of the minor ticks between the two major ticks.
- Data: You can add any number of series to the charts. To add the data, click the data item.
  - Color: It is the color of the data.
  - Left Y/Right Y: Choose to display data based on the left or right y-axis.

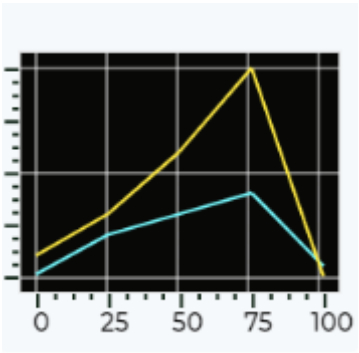

Figure 52. Chart

4.1.17 Canvas

A canvas inherits from an image where the user can draw anything. Rectangles, texts, images, lines, arcs can be drawn here using the drawing engine of LVGL. Also, "effects" can be applied, such as rotation, zoom, and blur.

The attributes specific to the canvas are as follows:

- Color: Sets the canvas background color.
- Opacity: Sets the canvas background opacity value.
- Canvas: Adds the canvas item.
- Type: To draw something to the canvas, use the following types:
  - Rect: Draws the rectangle.
    - Position: Sets the rectangle position in the canvas.
    - Background color and opacity: Sets the rectangle background color and opacity.
    - Size: The rectangle size (width and height).
    - Border color, width, and radius: The rectangle border settings.
- Text: Draws the text content.
  - Position: Sets the position in the canvas.
  - Max width: Sets the maximum width of the text.
  - Text font color, font family, and text content.
- Image: Draws the image.
  - Position: Sets the image position in the canvas.
  - Size: Sets the image size (width, height).
  - Image: Adds the image resource.
- Arc: Draws the arc
  - Position: Sets the arc position in the canvas.
  - Arc color, width, and radius.
  - Angle: Sets the start/end angle in degrees.
- Line: Draws the line between a set of points.
  - Line color and width: Sets the line style.
  - Line round: Selects whether to enable rounded corners.
  - Points: Adds the set of the points.
- Polygon: Draws the polygon by a set of points.
  - Polygon background color and opacity: Sets the polygon main style.
  - Polygon border color, border width, and Radius: Sets the polygon main style.

– Points: Adds the set of the points.

4.1.18 List

The list is basically a rectangle with a vertical layout to which buttons and texts can be added.

- Type: To select the added list item type, the following options are available:
  - Symbol: Adds the symbol item.
    - Text: Adds the description text for the symbol.
    - Symbol: The LVGL default symbol.
  - Image: Adds the image item.
    - Text: Adds the description text for the image icon.
    - Image: Adds the image resource and set the image size.
  - Text: Only shows the text in this item.

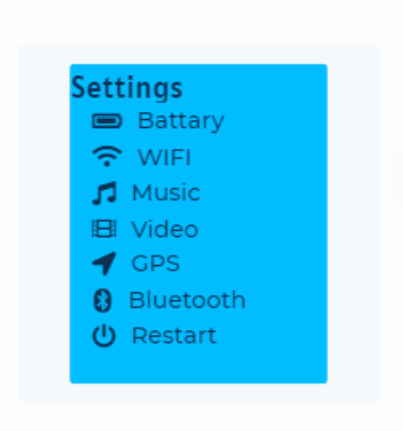

Figure 53. List

4.1.19 Window

The window is a container-like object built from a header with title, buttons, and a content area.

- Title: Sets the window title name.
- Height: Sets the window title height.
- Text: Add the text content; support the new line.
- Button: Refer to the list of item settings.

Figure 54. Window

4.1.20 Tile view

The tile view is a container object whose elements (called tiles) can be arranged in a grid form. A user can navigate between the tiles by swiping. Any direction of swiping can be disabled on the tiles individually to not allow moving from one tile to another.

- Direction: Sets the tile view direction; Support horizon and vertical.
- Page: Adds the new title for tile view.
  - Name: Sets the tile name.

Note: *To learn more, refer to Section 6.5.*

4.1.21 Menu

The menu widget can be used to easily create multi-level menus. It handles the traversal between pages automatically.

- Title: Sets the title name of the menu. It supports empty (no text).
- Page: Adds the new item for the menu.
  - label: Sets the menu item name.

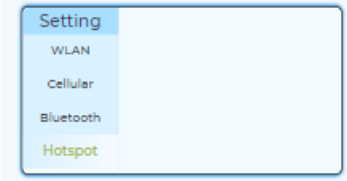

Figure 55. Menu

4.1.22 Arc

The arc consists of a background and a foreground arc. The foreground (indicator) can be touch-adjusted.

- Mode: The arc can be one of the following modes:
  - Normal: The indicator arc is drawn from the minimum value to the current.
  - Reverse: The indicator arc is drawn counter-clockwise from the maximum value to the current.
  - Symmetrical: The indicator arc is drawn from the middle point to the current value.
- Value: Starter value.
- Angle: Sets the start/end angle in degrees.
- Background: Sets the start/end angle of the background in degrees.
- Rotate: Supports the offset from 0 to 360 degree position.
- Arc rounded: Selects the arcs rounded or perpendicular line ending.

Figure 56. Arc

4.1.23 Line

The line object is capable of drawing straight lines between a set of points.

- Points: Adds the set of the points.

Figure 57. Line

4.1.24 Roller

Roller allows you to simply select one option from a list by scrolling.

- Direction: Switchs between normal and infinite modes:
  - Normal: You can roll from start to end.
  - Infinite: You can reroll the list.
- Row: The options number and the roller height changes with the row values.
- Option: Options you can select from the Roller list. You can create enlisted elements by adding new lines to the list.

4.1.25 LED

The LEDs are rectangle-like (or circle) objects whose brightness can be adjusted. With lower brightness, the colors of the LED become darker.

- Color: Sets the LED background color.
- LED bright: Sets the LED brightness. The value must be between 0 (darkest) and 25 (lightest).

4.1.26 Color

The color wheel allows the user to select a color. The hue, saturation, and value of the color can be selected separately. Long pressing the object, the color wheel changes to the next parameter of the color (hue, saturation, or value). A double click resets the current parameter.

4.1.27 Spinner

The spinner object is a spinning arc over a ring.

- Length: Sets the length of the spinning arc in degrees.
- Time: Sets the spin time in milliseconds.

Figure 58. Spinner

4.1.28 Spin box

The spinbox contains numbers as a text, which can be increased or decreased by keys or API functions. Under the hood, the spinbox is a modified text area.

- Digit – Sets the number of digits before and after the decimal point.

Figure 59. Spinbox

4.1.29 Meter

The meter widget can visualize data in flexible ways. It can show arcs, needles, tick lines, and labels.

- Dial: Click the "+" to add the new dial item for the meter.
  - Gap: Sets the distance of the tick value label from the tick line.
  - Range: Sets the value and angle range of the scale.
    - Value: The start and end value.
    - Angle: The angle range of the scale, from 0 to 360 degrees.
  - Tick: Sets the minor tick.
    - Color: Color of the minor tick line.
    - Count: The total minor tick count.
    - Size: Width and height of the minor tick lines.
  - Major tick: Sets the major tick.
    - Major tick enable: Selects whether to display the major tick.
    - Color: Color of the major tick lines.
    - Index: Sets the number of intervals between two major ticks.
    - Size: Width and height of the major tick lines.
  - Needle: Adds a needle line to a scale.
    - Color: The needle line color.
    - Value: The needle line initial value.
    - Size: The needle line size. By default, the length of the line is the same as the radius of the scale. The radius changes the length.
  - Image needles: Adds an image that is used as a needle.
    - Image: Adds the image resource.
    - Size: Sets the image width and height.
    - X/Y: Sets the pivot point of the rotation relative to the top-left corner of the image.
    - Value: The needle initial value.
  - Arcs: Adds the arc indicator.
    - Color and width: Sets the arc color and width.
    - Start/end: Sets the arc length.
    - Radius: Sets the arc radius. The radius of the arc is the same as the radius of the scale.
  - Scale lines: Adds an indicator that modifies the tick lines.
    - Start value/color: Sets the initial value and color of the scale line.
    - End value/color: Sets the end value and color of the scale line.

– Gradient: If the value is true, the ticks' color is faded from color_start to color_end in the indicator's start and end value range. If the value is false, the color_start and color_end is mapped to the start and end value of the scale and only a "slice" of that color gradient is visible in the indicator's start and end value range.

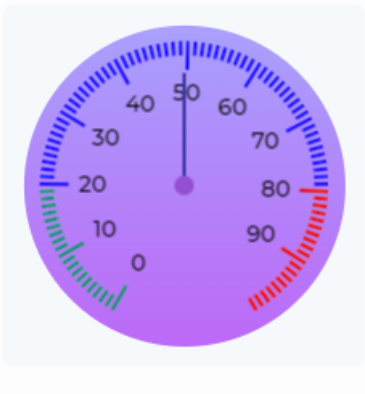

Figure 60. Meter

4.1.30 Image

Images are the basic objects to display images from flash (as arrays) or from files.

- External storage
  - Flash: Refer to Section 5.4.2.
  - SD card: Refer to Section 5.4.1.
- Image: Adds the image resource.
- Center: Sets the rotation and zoom center of the image.
- Rotation: Sets the rotation angle in degrees.
- Format: Selects the format parameter when generating the picture *.c file.

4.1.31 Animation image

The animation image is similar to the normal "Image" object. The only difference is that instead of one source image, you set an array of multiple source images.

- External storage
  - Flash: Refer to Section 5.4.2.
  - SD card: Refer to Section 5.4.1.
- Interval: Sets the interval time between pictures.
- Repeat: Sets the repeat count. If the value is -1, the animation plays infinitely.
- Play back: Sets the play back time and delay time.
- Start/ready callback function: Sets a callback function to indicate when the animation is started or ready.
- Auto play: Enables the animation play status.
- Reverse: Turns on the reverse playback.
- Image: Selects the set of images. Or, you can import the GIF image. It automatically gets divided into several pictures.

4.1.32 3D image

The 3D image is similar to the normal "Image" object. It can rotate a given image along with x-axis, y-axis, z-axis, or combined.

- External Storage
  - Flash: Refer to Section 5.4.2.
  - SD card: Refer to Section 5.4.1.
- Repeat: Sets the repeat count. If the value is -1, the animation plays infinitely.
- Frame: Sets the number of frames for 3D animation.
- Interval: Sets the interval time between pictures.
- Play back: Sets the play back time and delay time.
- Start/ready callback function: Sets a callback function to indicate when the animation is started or ready.
- Auto play: Enables the animation play status.
- Reverse: Turns on the reverse playback.
- Image: Selects the set of images. Or, you can import the GIF image. It automatically gets divided into several pictures.
- X/Y/X-axis: Sets the rotation angle of the three axes of the 3D animation.
- Image: Selects the image.

4.1.33 Bar

The bar widget has a background and an indicator on it. The width of the indicator is set according to the current value of the bar. Vertical bars can be created if the width of the object is smaller than its height. Not only the end, but also the start value of the bar can be set, which changes the start position of the indicator.

- Value: Defines the current value.
- Anim time: Defines the animation time.
- Mode: Sets the bar modes:
  - Normal: A normal bar as described above.
  - Symmetrical: Draws the indicator from zero value to the current value. It requires a negative minimum range and positive maximum range.

Figure 61. Bar

4.1.34 Slider

The slider widget looks like a bar supplemented with a knob. The knob can be dragged to set a value. Just like the bar, a slider can be vertical or horizontal.

- Value: Defines the slider range; Maximum and minimum values.
- Init value: Sets the initial left value and right value.
- Mode: Sets the bar modes:
  - Normal: A normal bar as described above.
  - Symmetrical: Draws the indicator from zero value to the current value. It requires a negative minimum range and positive maximum range.
  - Range: Allows setting the start value. The start value has to be always smaller than the end value.

Figure 62. Slider

4.2 Advance widget

These widgets are maintained by the GUI Guider team and are different from the widgets provided by the open source LVGL. Therefore, if you use these widgets in your project, make sure that LVGL is of GUI Guider.

4.2.1 Lottie

The lottie widget allows you to use the lottie animations in LVGL. LVGL provides the interface to the C API of Samsung/rlottie library.

Note: *Due to limitations, it is recommended to import the project into MCUXpresso IDE, build, and deploy.*

4.2.2 QR code

It generates a QR code based on the input text, typically used for storing URLs or information.

Note: *QR codes with less data are smaller, but they are scaled by an integer number to best fit to the given size.*

Figure 63. QR code

4.2.3 Barcode

It generates a Barcode based on the input text, typically used for storing URLs or information.

Note:

- *It is best not to manually set the width of the barcode, because when the width of the object is lower than the width of the barcode, the display is incomplete due to truncation.*
- *The scale adjustment can only be an integer multiple. For example,* `lv_barcode_set_scale(barcode, 2)` *means 2x scaling.*

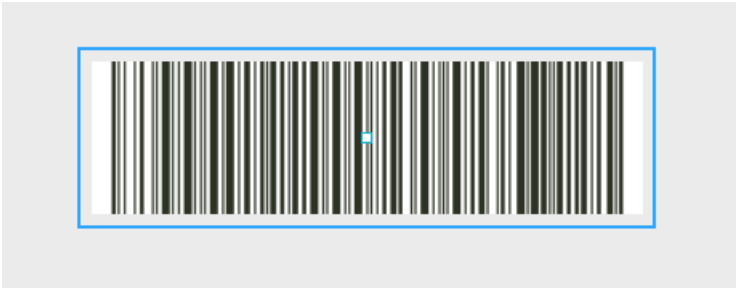

Figure 64. Barcode

4.2.4 Analog clock

Analog clock is a basic object with a meter. It can show the time dynamic.

- Gap: Sets the distance of the tick value label from the tick line.
- Hide digits: Disables the time digits.
- Hide point: Selects whether to display the center point.
- Minute tick: Sets the minute tick.
  - Color: The color of the minute tick lines.
  - Size: The width and height of the minute tick line.
- Hour tick: Sets the hour tick.
  - Color: The color of the hour tick lines.
  - Size: The width and height of the hour tick lines.
- Hour/Minute/Second: Adds the needle line to a scale.
  - Need type: Selects the need type.
    - Line needle
      - Color: Sets the line color.

  - Value: The initial value of the needle line.
  - Size: The needle line size. By default, the length of the line is the same as the scale's radius. The radius changes the length.
- Image needle
  - Value: The needle line initial value.
  - Image: Adds the image resource.
  - Size: Sets the image width and height.
  - X/Y: Sets the pivot point of the rotation relative to the top-left corner of the image.

Figure 65. Analog clock

4.2.5 Carousel

A carousel is a basic object. The carousel widget can display two or more pieces of content in a carousel format.

- Width: Sets the width of each item.
- Page: Clicks the "+" to add a new item.
  - Name: Sets the name of each item.
    Note: *Select an item in the widget tree or attribute setting tab, and then you can add widgets to this item. "DashBoardMenu" application template in a new project wizard demostrates how to use carousel widget.*

4.2.6 Video

Video is a basic object with image. It can play video files of H264 format, which can be converted from other formats by FFmpeg. Video widget is decoded and displayed in real time, using PXP on the board.

- Auto Play: Whether to auto play when the screen loading.
- SD path: The video *.h264 file location in SD path.
- Local resource list: Adds the *.h264 file.
  Note: *For more detail, refer to Section 5.4.1.2.*

4.2.7 Digital clock

A digital clock is a basic object with a label. It can show the dynamic time and supports the 12-hour and 24-hour mode.

- Time: Set the initial time; You can set the current time by clicking "Now".
- Second: Select whether to display seconds.
- AM/PM: Select the mode in which the time is displayed.

Figure 66. Digital clock

4.2.8 Text progress bar

A text progress bar object is built from a label. A text progress displays progress as a number with a given number of decimals.

- Range: Sets the progress range.
- Step: Sets the jumping steps whose value is changed from start value to end value. The "steps min" defines the default start step.
- Initial value: Sets the initial value of the text progress bar.
- Decimals: Selects the number of the digits after the decimal point. It supports 0, 1, and 2.

Figure 67. Text progress bar

4.2.9 Radio button

A radio button is a basic object with a button. The radio button allows the user to choose only one of a predefined set of mutually exclusive options.

- Item: Adds the items.

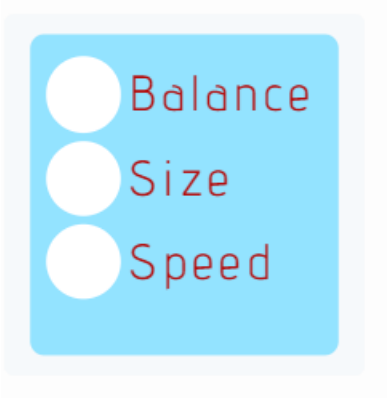

Figure 68. Radio button

4.2.10 Chinese input keyboard

A Chinese input keyboard is a basic object with a keyboard. It includes two sizes of fonts. You can enable the Chinese input keyboard on the project setting. If you enable the keyboard on the current project, the text area widget is able to trigger it.

- Keyboard of the current project: Enables the keyboard; the default is keyboard by LVGL.
- Font: Sets the font family. It is important to note that the font you choose must support Chinese. SourceHanSerifSC-Regular is supported on the built-in font family.
- Size: Sets the keyboard font size.
- Chinese input: Enables the Chinese input mode.
  - Library: Selects the Chinese characters.
    - Mini: It includes 1647 Chinese characters.
    - Full: It includes 7455 Chinese characters.

4.2.11 Date text

A date text widget is built from a label and a calendar. A date text box is a widget that allows users to select a date in the calendar and display the date in the label.

- Date text – Selects the date in the calendar. You can set the current date by clicking "today".

Figure 69. Date text

4.3 Style

Styles are used to set the appearance of objects. Styles in LVGL are heavily inspired by CSS.

4.3.1 Preset style usage

Click the **More Preset** button. It lists the default style of this widget. Select one to apply.

Note: *Use the default preset style.*

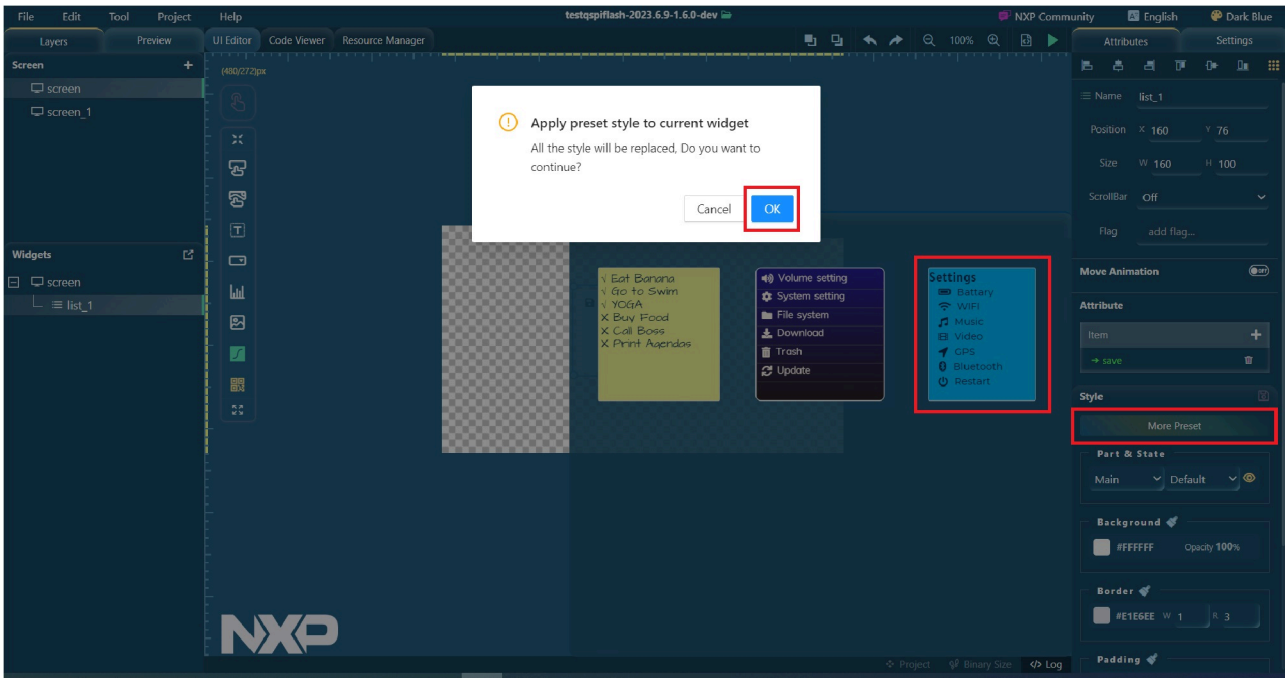

Figure 70. Default preset style

4.3.2 Custom style

When you want to save your own preset style for easy design, click the **OK** button.

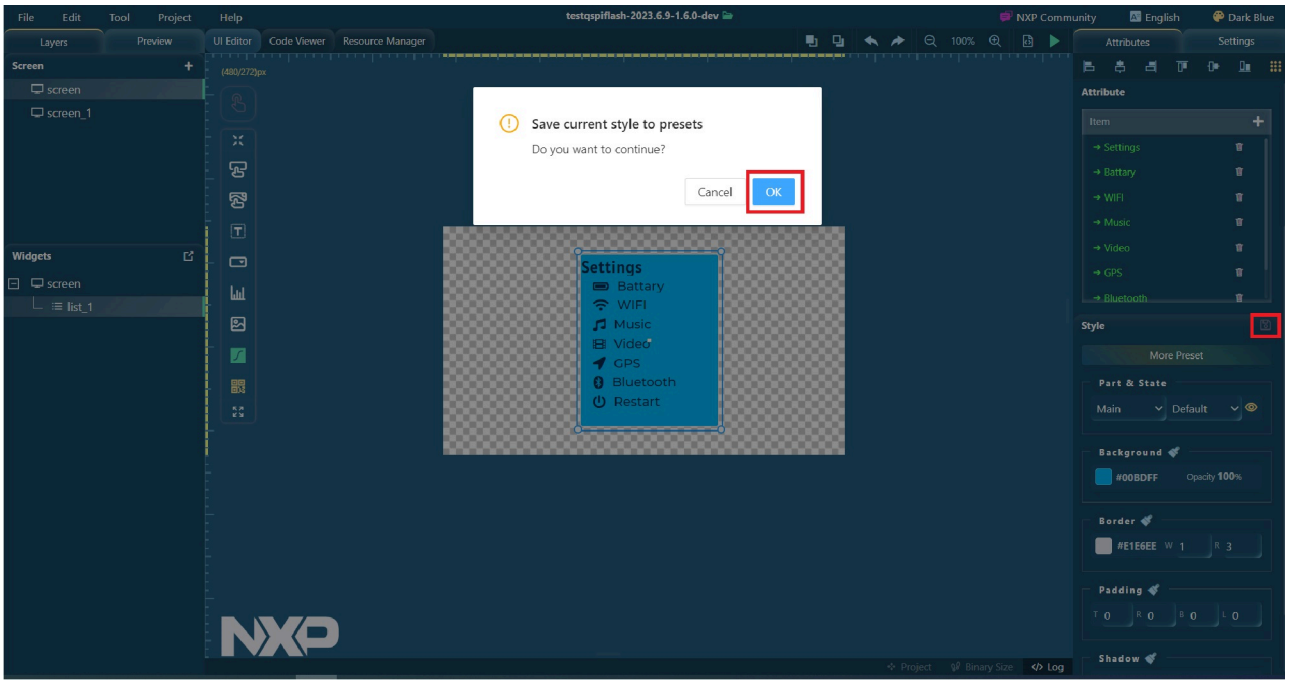

Figure 71. Saving preset style

You can find the custom style file at the following locations:

Windows:

```
%appdata%\\gui-guider\\{version}\\preset_record.setting
```

Ubuntu:

```
~/.config/gui-guider/{version}/preset_record.setting
```

MacOS:

```
~/Library/Application Support/gui-guider/{version}/preset_record.setting
```

4.3.3 Initialize styles

Styles are stored in `lv_style_t` variables. Style variables must be static, global, or dynamically allocated. In other words, they cannot be local variables in functions, which are destroyed when the function exits. Before using a style, it must be initialized with `lv_style_init(&gg_style)`. After initializing a style, properties can be added or changed. GUI Guider uses this method in widgets with multiple sub-items.

4.3.4 Local styles

In addition to "normal" styles, objects can also store local styles. This concept is similar to inline styles in CSS (For example,) with some modification.

Local styles are like normal styles, but they cannot be shared among other objects. If used, local styles are allocated automatically and freed when the object is deleted. They are useful to add local customization to an object.

4.3.5 Parts

Objects can be composed of parts, which can have their own styles.

The following predefined parts exist in LVGL:

- LV_PART_MAIN: A background like a rectangle.
- LV_PART_SCROLLBAR: The scrollbar(s).
- LV_PART_INDICATOR: Indicator. For example, for slider, bar, switch, or the tick box of the checkbox.
- LV_PART_KNOB: Like a handle to grab or to adjust a value.
- LV_PART_SELECTED: Indicate the currently selected option or section.
- LV_PART_ITEMS: Used if the widget has multiple similar elements (for example, table cells).
- LV_PART_TICKS: Ticks on scales. For example, for a chart or meter.
- LV_PART_CURSOR: Mark a specific place. For example, the cursor of the text area or chart.
- LV_PART_CUSTOM_FIRST: Custom part identifiers can be added starting from here.

4.3.6 States

The objects can be in the combination of the following states:

- LV_STATE_DEFAULT: Normal, released state.
- LV_STATE_CHECKED: Toggled or checked state.
- LV_STATE_FOCUSED: Focused via keypad or encoder, or clicked via touchpad/mouse.
- LV_STATE_PRESSED: Being pressed.
- LV_STATE_SCROLLED: Being scrolled.
- LV_STATE_DISABLED: Disabled state.

4.3.7 Properties

Tip: *You can click the*  *to copy the current style and paste it, like the Format Painter.*

4.3.7.1 Background

Background style is the background of the widgets. You can create gradients or make the corners of the background rounded.

- Color: Sets the background color of the object.
- Gradient color: Sets the gradient color of the object.
- Gradient direction: The direction of the gradient. It can be horizontal or vertical.
- Opacity: Sets the background opacity of the object; 0 – 255.
- Background image: Sets an image as the background.
- Background image opacity: Sets the background image opacity of the object.
- Background image fill color: Sets the background fill color of the object.

4.3.7.2 Font

Text style defines the parameters of the text that can be found on the widget.

- Font color: The color of the text.
- Font size: The text size.
- Font family: Select the font type.
- Align: The direction of text alignment.

- Letter spacing: The space between the letters.
- Line spacing: The space between the lines.
- Text décor: You can overline or underline the text.
  - None: Normal text.
  - Understand: Underlined text.
  - Strikethrough: Overlined text.

4.3.7.3 Border

Using Border, you can draw a border around the selected object onto the inner lines.

- Border color: The color of the border.
- Border width: The width of the border.
- Border radius: The radius of the border.
- Border side: You can set the direction of the border.

4.3.7.4 Line

A Line style can be used in those widgets, which have the line component.

- Color: The color of the line.
- Width: The width of the line.
- Line rounded: To set the ends of the line as rounded.

4.3.7.5 Padding

Properties to describe spacing between the parent's sides and the children and among the children.

- Padding_top: Sets the padding on the top.
- Padding_right: Sets the padding on the right.
- Padding_bottom: Sets the padding on the bottom.
- Padding_left: Sets the padding on the left.

4.3.7.6 Shadow

Using a shadow style, you can draw a shadow or a glow to the selected widget part.

- Shadow color: The color of the shadow.
- Shadow Opacity: The opacity of the shadow.
- Shadow X/Y: The position of the shadow. It shifts the shadow on the X/Y axis.
- Shadow width: The width of the shadow.
- Shadow spread: The depth of the shadow.

4.4 Event

Add a variety of events to have a better interactive experience. For example, change the screen by clicking a button.

4.4.1 Add event

To add events, select a widget that you want to add. Right-click the mouse button or press Ctrl + E and select the add event option. Then choose the required trigger to add it.

Table 15. Triggers

Trigger name	Description
Clicked	An object pressed for a short period, then released. Not called if scrolled.
Short Clicked	A short click is detected.
Pressed	An object has been pressed.
Pressing	An object is being pressed (called continuously while pressing).
Press Lost	An object is still being pressed but slid cursor/finger off the object.
Long Pressed	An object has been pressed for at least the `long_press_time` specified in the input device driver. Not called if scrolled.
Long Pressed Repeat	Called after `long_press_time` in every `long_press_repeat_time` ms. Not called if scrolled.
Released	Called in every case when an object has been released.
Value changed	The value of the object has been changed.
Scroll	An object was scrolled.
Scroll Begin	Scrolling begins.
Scroll End	Scrolling ends.
Focused	An object is focused.
Defocused	An object is unfocused.
Leave	An object is unfocused but still selected.
Hit Test	Perform advance hit-testing.
Key	A key is sent to an object.
Loaded	A screen has been loaded, called when all animations are finished.
Unloaded	A screen has been unloaded, called when all animations are finished.
Load start	A screen load started, fired when the screen change delay is expired.
Unload Start	A screen unload started, fired immediately when `lv_scr_load`/`lv_scr_load_anim` is called.
Gesture Left, Right, Bottom, Top	A gesture is detected on a widget with the selected direction.

4.4.2 Set action

After you confirm the trigger, you must select widgets and actions. Only the widgets in the current screen can be selected. The types of actions vary depending on the type of widget selected. Table 16 describes all currently supported actions.

Table 16. Actions

Action name	Description
Visibility	Set the hide or show for the widget.

Table 16. Actions...*continued*

Action name	Description
Add Flag	Add the widget flags.
Clear Flag	Remove the widget flags.
Add state	Add the state for the widget.
Set text	Modify the show text.
Width	Update the widget width size.
Height	Update the widget height size.
Position	Move the widget position.
Background	Change the background color.
Gradient	Change the color gradient.
Opacity	Set the widget opacity.
Move	Support play time, repeat, and type.
Scale	Dynamically changing widget size.
Rotate	Supports only image widget
Image zoom	Set the zoom size: the max value is 300 %.
Bg Opa	Update the background opacity
Boarder Width	Change the boarder width
Radius	Change the widget radius
Boarder Color	Update the boarder color
Boarder Opa	Update the boarder opacity
Boarder Side	Set the boarder side
bg Img Opa	Set the background image opacity
bg img recolor opa	Set the background image recolor opacity
bg img recolor	Modify the widget background image recolor

Note: *If the repeat value is -1, the animation runs infinite times.*

4.4.3 Custom code action

If you want to define the action yourself, use the custom code option.

First, implement the function logic code in the `custom.c` under the custom folder. Then, add the call in the event window. For example, Figure 72 demonstrates the example for this use case or case.

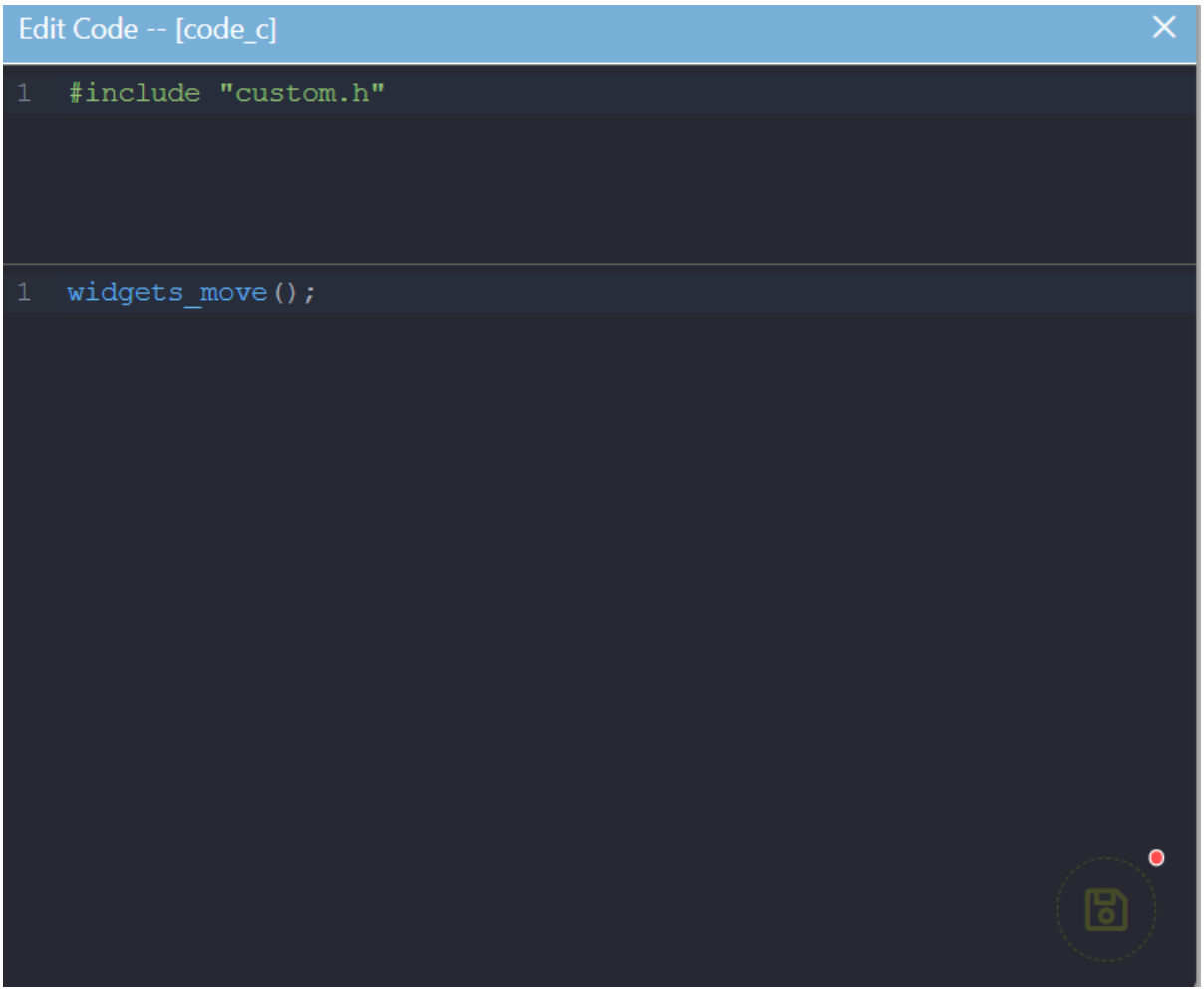

Figure 72. Custom code

4.4.4 Load screen

Projects generated by GUI Guider load on one screen. Other screens in the project is not instantiated by default.

So, the load screen only supports one target screen. You can make some settings for loading screen animation. The delay and duration cannot be set to 0 at the same time, otherwise memory leaks occur.

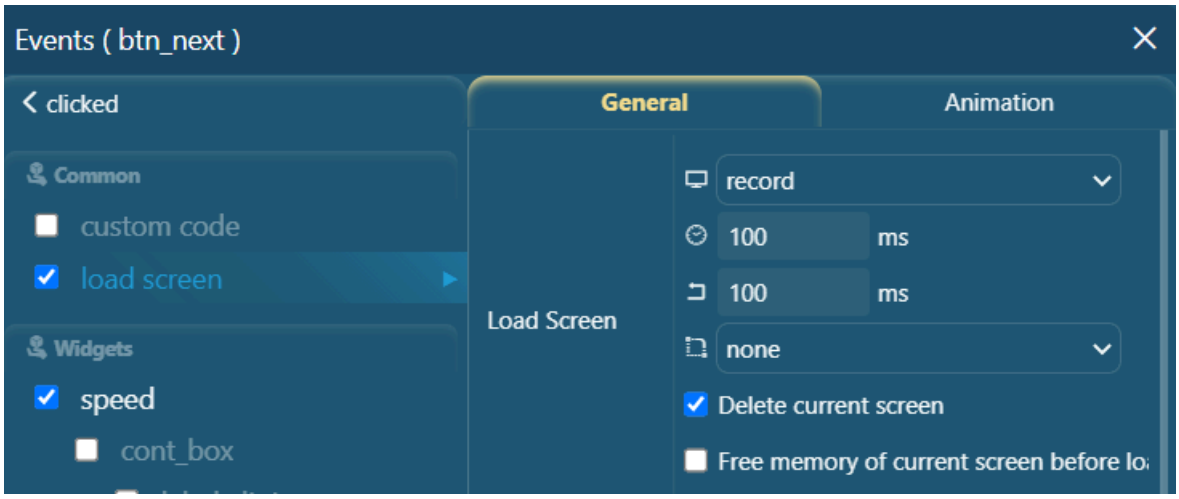

Figure 73. Load screen

5 Development

This chapter describes how to develop an HMI application in GUI Guider.

5.1 Debug project

This section contains information on how to debug the GUI Guider project.

5.1.1 Target

A GUI Guider project integrates the support of different IDE toolchains, including MCUXpresso IDE, IAR, Keil MDK. This chapter describes how to debug an HMI application by an IDE.

5.1.1.1 MCUXpresso

To debug the GUI Guider project on MCUXpresso, perform the following steps:

1. Open the link https://mcuxpresso.nxp.com/en/select.
2. Select the development board. For example, EVK-MIMXRT1064.
3. Click the **Build MCUXpresso SDK** button.
4. Select the two middleware LVGL and FreeRTOS from the **Build SDK for <target>** page.
5. Make sure to select the MCUXpresso (toolchain).
6. Click the **Download SDK** button.

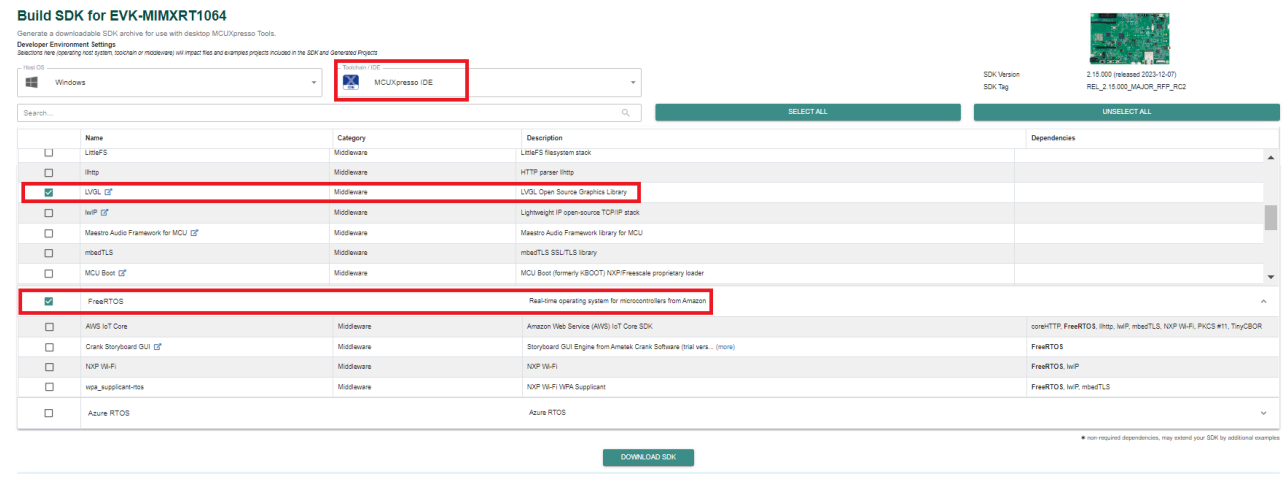

Figure 74. Downloading SDK

7. Import the downloaded SDK into the IDE.
8. Click **File** > **Import** > **General**.

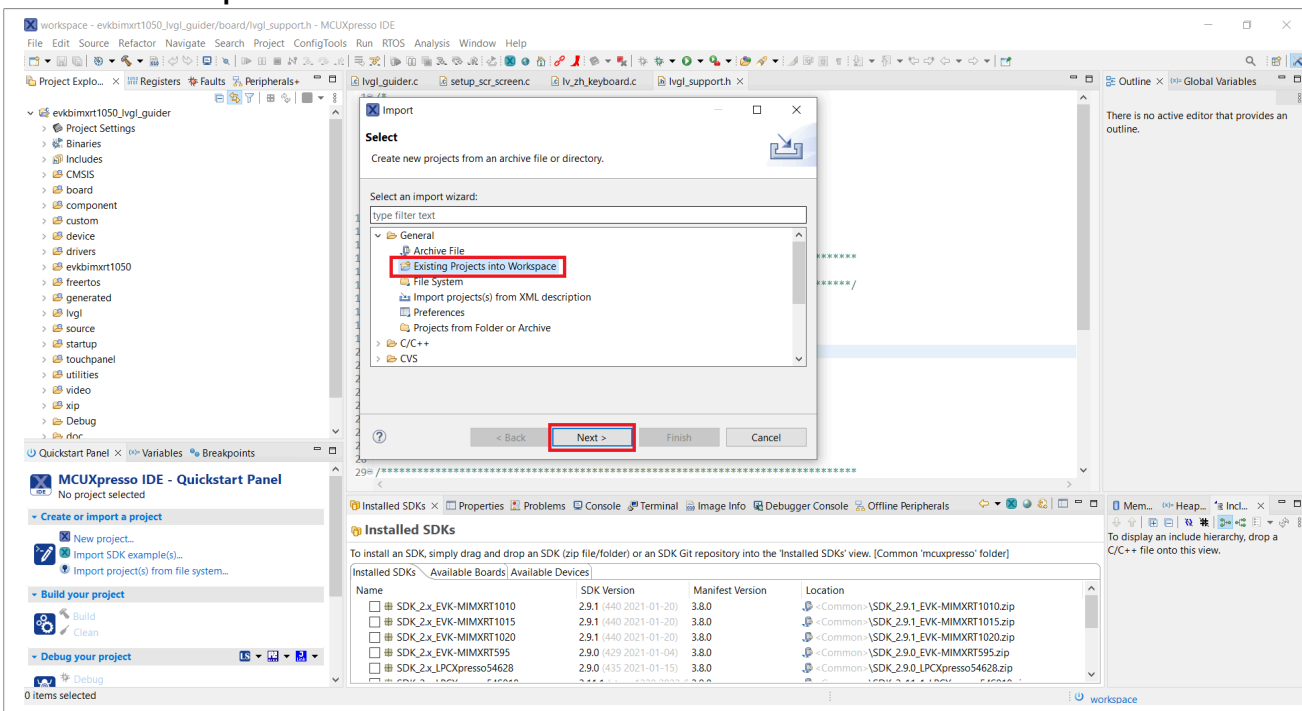

Figure 75. Importing project

9. Select the GUI Guider project MCUXpresso path.

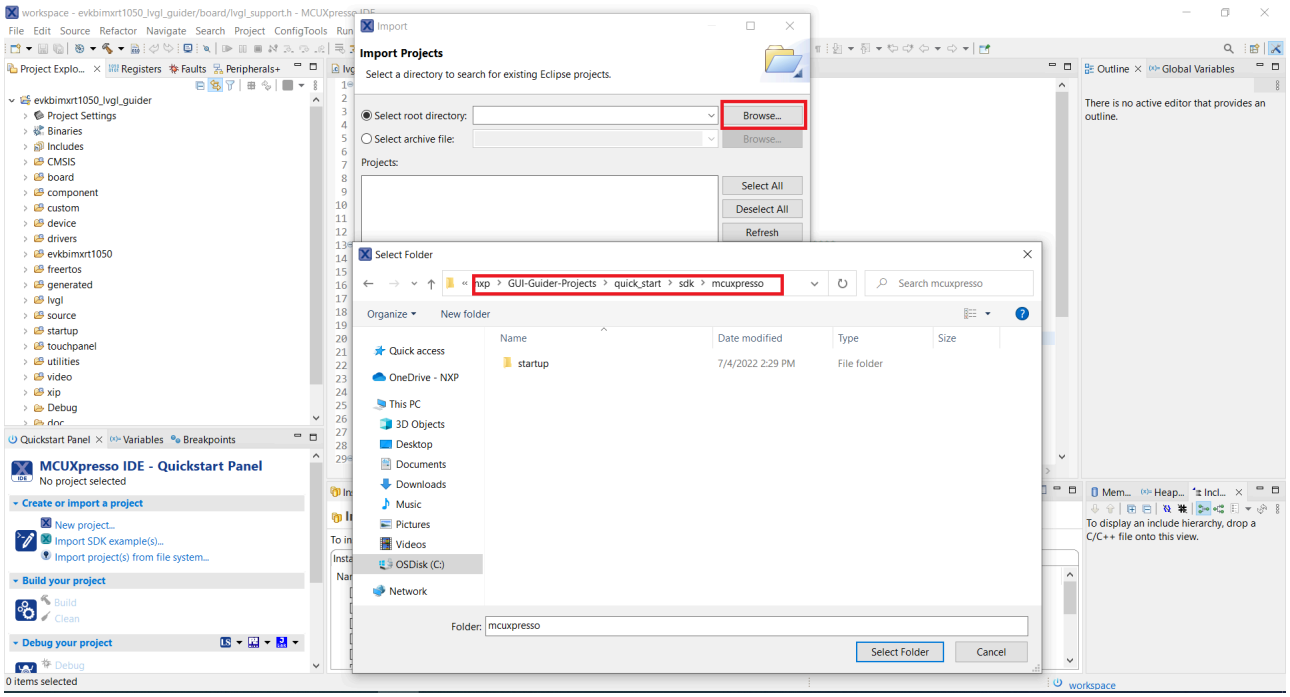

Figure 76. Selecting the GUI Guider project path

5.1.1.2 IAR

Find the path named "iar", double click `lvgl_guider.eww`.

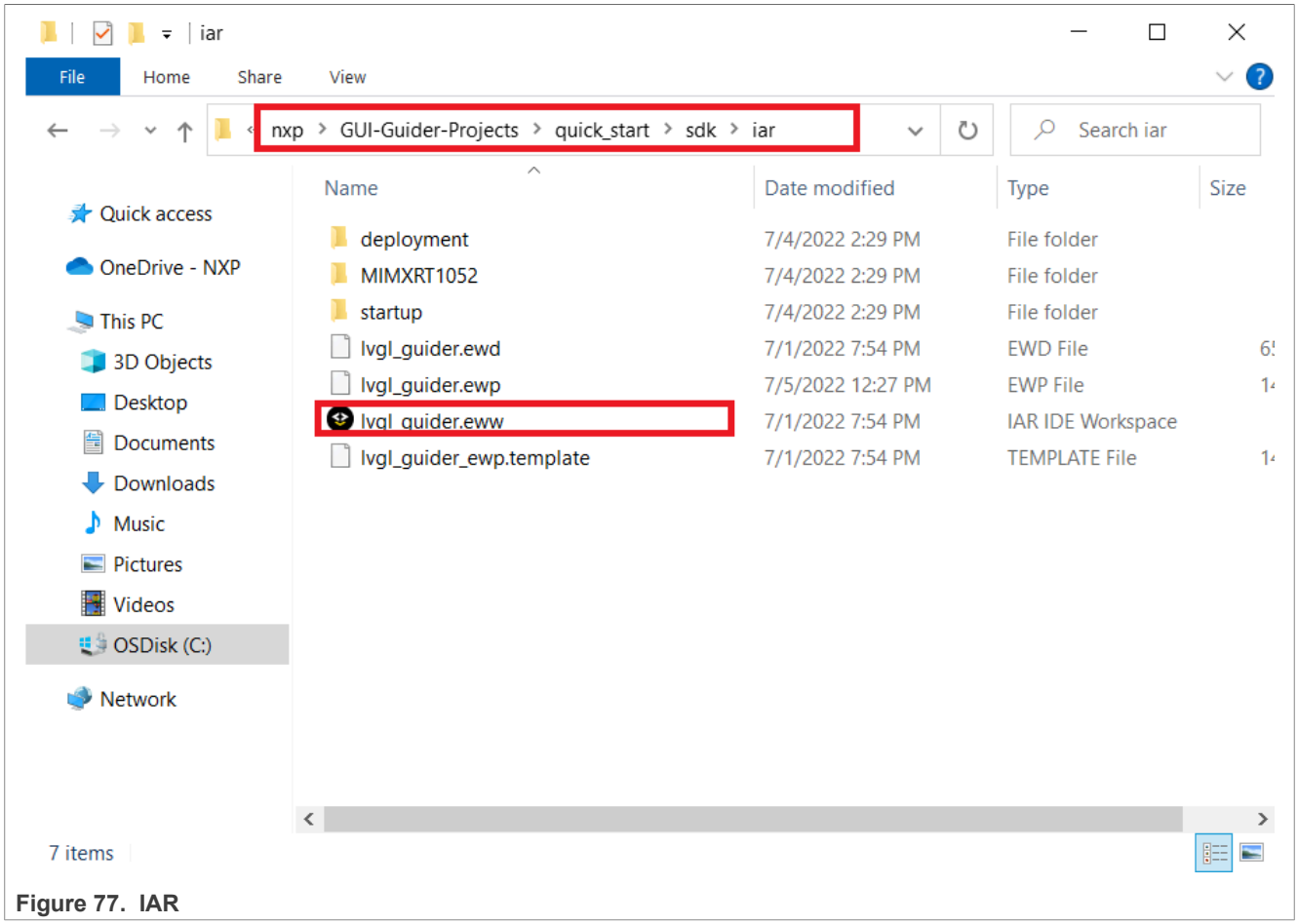

Figure 77. IAR

5.1.1.3 Keil MDK

Find the path named "mdk", double click `lvgl_guider.uvprojx`.

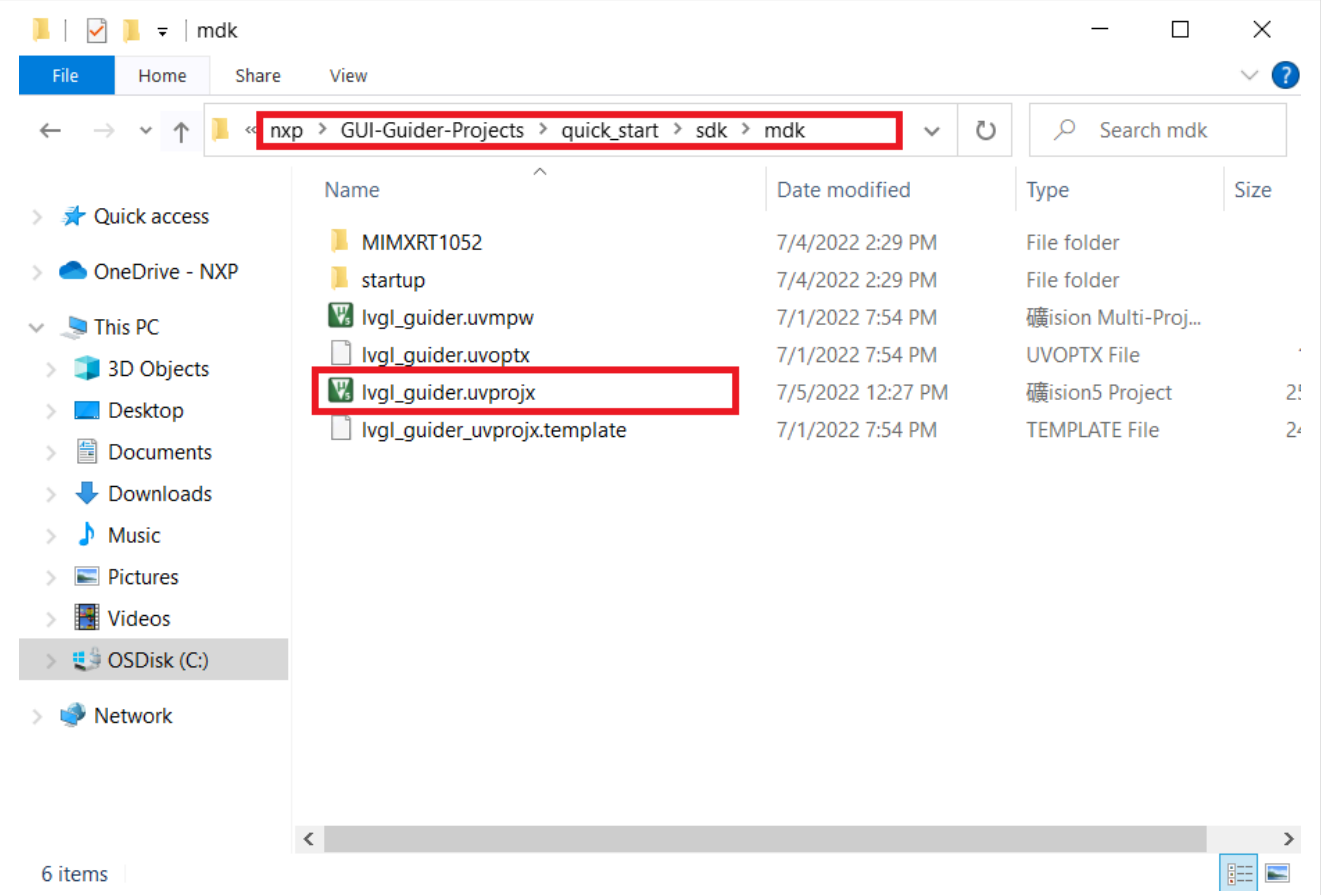

Figure 78. Keil MDK

5.2 Hardware acceleration

LVGL is a software library that fully implements and customizes a graphical user interface (drawing, partial screen refresh, input events, and animations). LVGL has software pixel-based draw engine. Several drawing features in LVGL are performed by hardware (HW) accelerators instead of CPU.

To use the CPU time while HW accelerator is running, an RTOS is required to block the LVGL drawing thread and switch to another task, or idle task, where CPU is suspended to save power. The HW accelerators process pixels faster than CPU resulting in a higher frame rendering rate.

GUI Guider can enable and disable the PXP or VGLite accelerator for the devices that support these features.

Note: *It is possible to enable or disable the HW accelerator manually.*

Table 17. LVGL hardware acceleration

Accelerator	MIMXRT1040 EVK	MIMXRT1050 EVKB	MIMXRT1060 EVK	MIMXRT1060 EVKB	MIMXRT1060 EVKC	MIMXRT1064 EVK	MIMXRT1170 EVK	MIMXRT1170 EVKB	MIMXRT1160 EVK	MIMXRT595-EVK
PXP	√	√	√	√	√	√	√	√	√	X
VGLite	X	X	X	X	X	X	√	√	√	√

5.2.1 PXP enablement

Enable the PXP accelerator in GUI Guider.

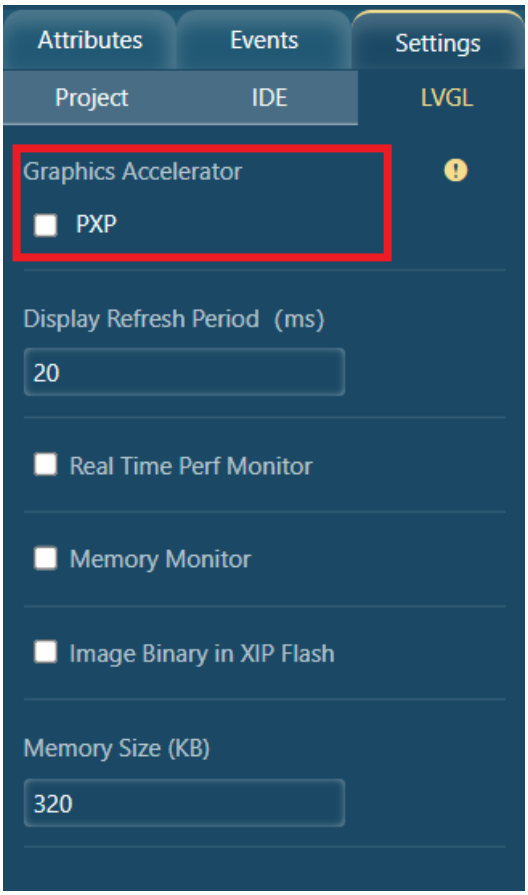

Figure 79. Enable PXP

To enable the PXP accelerator on NXP devices, set the below flag in `lv_conf.h`. This is required as currently only the color format RGB565 (16 bits) is accelerated on NXP devices.

```
#define LV_COLOR_DEPTH 16
```

PXP is a pixel processing HW engine. To check whether PXP is available on your NXP device, see the Reference Manual document or the board configuration.

To enable PXP in LVGL, set the below flags to 1 in `lv_conf.h`.

```
#define LV_USE_GPU 1
#define LV_USE_GPU_NXP_PXP 1
#define LV_USE_GPU_NXP_PXP_AUTO_INIT 1
```

In LVGL, PXP is used to accelerate the following features:

- Area fill + optional transparency
- BLIT (Block image transfer) + optional transparency
- Color keying + optional transparency
- Recoloring (color tint) + optional transparency

5.2.2 VGLite enablement

Enable the VGLite accelerator in GUI Guider.

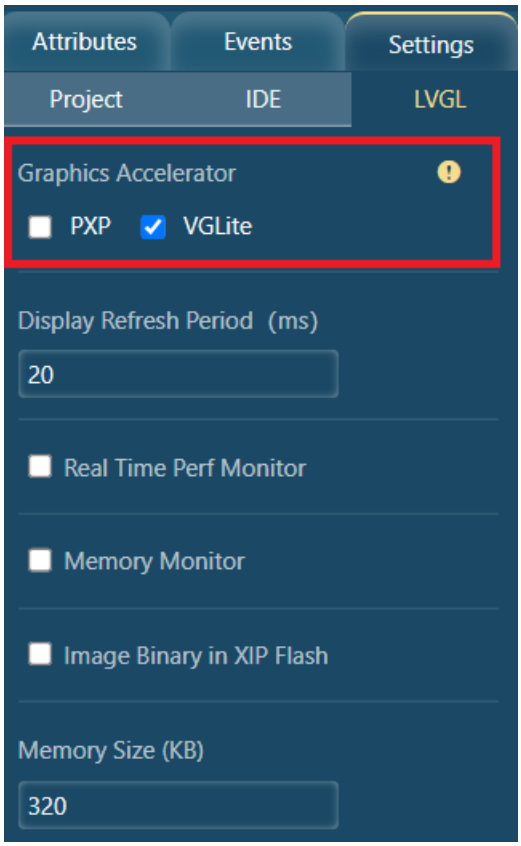

Figure 80. VGLite enablement

To enable the VGLite accelerator on NXP devices, set the below flag in `lv_conf.h`. This is required as currently only the color format RGB565 (16 bits) is accelerated on NXP devices.

```
#define LV_COLOR_DEPTH 16
```

VGLite is an API that uses the vector/raster 2D GPU. To check whether 2D GPU is available on your NXP device, see the Reference Manual document or the board configuration.

To enable VGLite in LVGL, set the below flags to 1 in `lv_conf.h`.

```
#define LV_USE_GPU 1
#define LV_USE_GPU_NXP_VG_LITE 1
```

In LVGL, VGLite is used to accelerate the following features:

- Area fill + optional transparency
- BLIT (Block image transfer) + optional transparency

5.2.3 Recommendations to improve acceleration

This section lists general and VGLite recommendations to improve acceleration.

5.2.3.1 General recommendations

As a rule when a hardware accelerator processes many pixels in a single batch, it provides better performance than processing small number of pixels multiple times.

The reasons are:

1. **Caches**: Pixels previously processed by CPU are loaded in cache, and must be cleaned and invalidated. The operation takes a few cycles.
2. **Setup time**: Each time HW is used to process pixels, the associated driver configures HW registers. This operation also takes a few cycles.

Therefore, NXP has defined a threshold for the minimum number of pixels necessary to trigger HW acceleration. These thresholds are defined as preprocessor variables.

For PXP, default values are defined in `lv_gpu/lv_gpu_nxp_pxp.h`.

- **LV_GPU_NXP_PXP_BLIT_SIZE_LIMIT**: Size threshold for image BLIT, BLIT with color keying, and BLIT with recolor (OPA > LV_OPA_MAX).
- **LV_GPU_NXP_PXP_BLIT_OPA_SIZE_LIMIT**: Size threshold for image BLIT and BLIT with color keying with transparency (OPA < LV_OPA_MAX).
- **LV_GPU_NXP_PXP_FILL_SIZE_LIMIT**: Size threshold for fill operation (OPA > LV_OPA_MAX).
- **LV_GPU_NXP_PXP_FILL_OPA_SIZE_LIMIT**: Size threshold for fill operation with transparency (OPA < LV_OPA_MAX).

For VGLite, default values are defined `lv_gpu/lv_gpu_nxp_vglite.h`.

- **LV_GPU_NXP_VG_LITE_BLIT_SIZE_LIMIT**: Size threshold for image BLIT (OPA > LV_OPA_MAX).
- **LV_GPU_NXP_VG_LITE_BLIT_OPA_SIZE_LIMIT**: Size threshold for image BLIT with transparency (OPA < LV_OPA_MAX).
- **LV_GPU_NXP_VG_LITE_FILL_SIZE_LIMIT**: Size threshold for fill operation (OPA > LV_OPA_MAX).
- **LV_GPU_NXP_VG_LITE_FILL_OPA_SIZE_LIMIT**: Size threshold for fill operation with transparency (OPA < LV_OPA_MAX).

5.2.3.2 VGLite recommendations

The 2D GPU behind VGLite has some constraints on the processed buffers:

1. **Address alignment**: Always ensure that the FrameBuffer and pixel buffers are aligned to `LV_ATTRIBUTE_MEM_ALIGN_SIZE`. Use the macro `LV_ATTRIBUTE_MEM_ALIGN` as attribute for statically allocated pixel buffers.
2. **Stride**: Stride is the byte offset between two lines of pixels. 2D GPU requires a stride multiple of 16 pixels.

In LVGL: stride = width, so use assets and widgets with a width multiple of 16 pixels.

On platforms like i.MX RT1170, which has both PXP and 2D GPU, prefer 2D GPU as it draws faster than PXP. However, if the GUI contains many pre-rendered semi-transparent images, PXP may be better.

On platforms with only 2D GPU acceleration (VGLite), try to draw widgets rather than using pre-rendered images as widget, as semi-transparent image blitting is not yet accelerated.

5.3 Performance optimization

The high graphics performance means a high frame rate (FPS) with required graphical effects. This section provides the introduction to enable/disable FPS/CPU usage monitor and the tips on how to improve the graphics performances on NXP MCU devices. i.MX RT595 is used as an example platform for performance optimization.

5.3.1 Performance monitor enablement

In actual development, if we want to know the performance of the app, perform the following steps:

1. Enable the performance monitor in GUI Guider.

2. Check the real-time performance results in simulator.

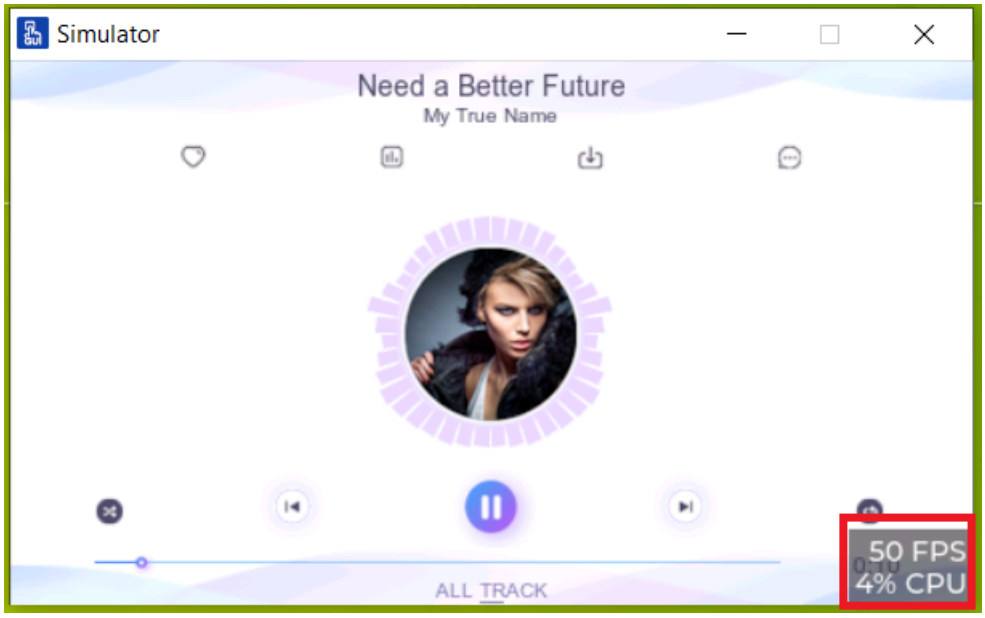

Figure 81. Performance monitor

3. Check the real-time performance results on boards.

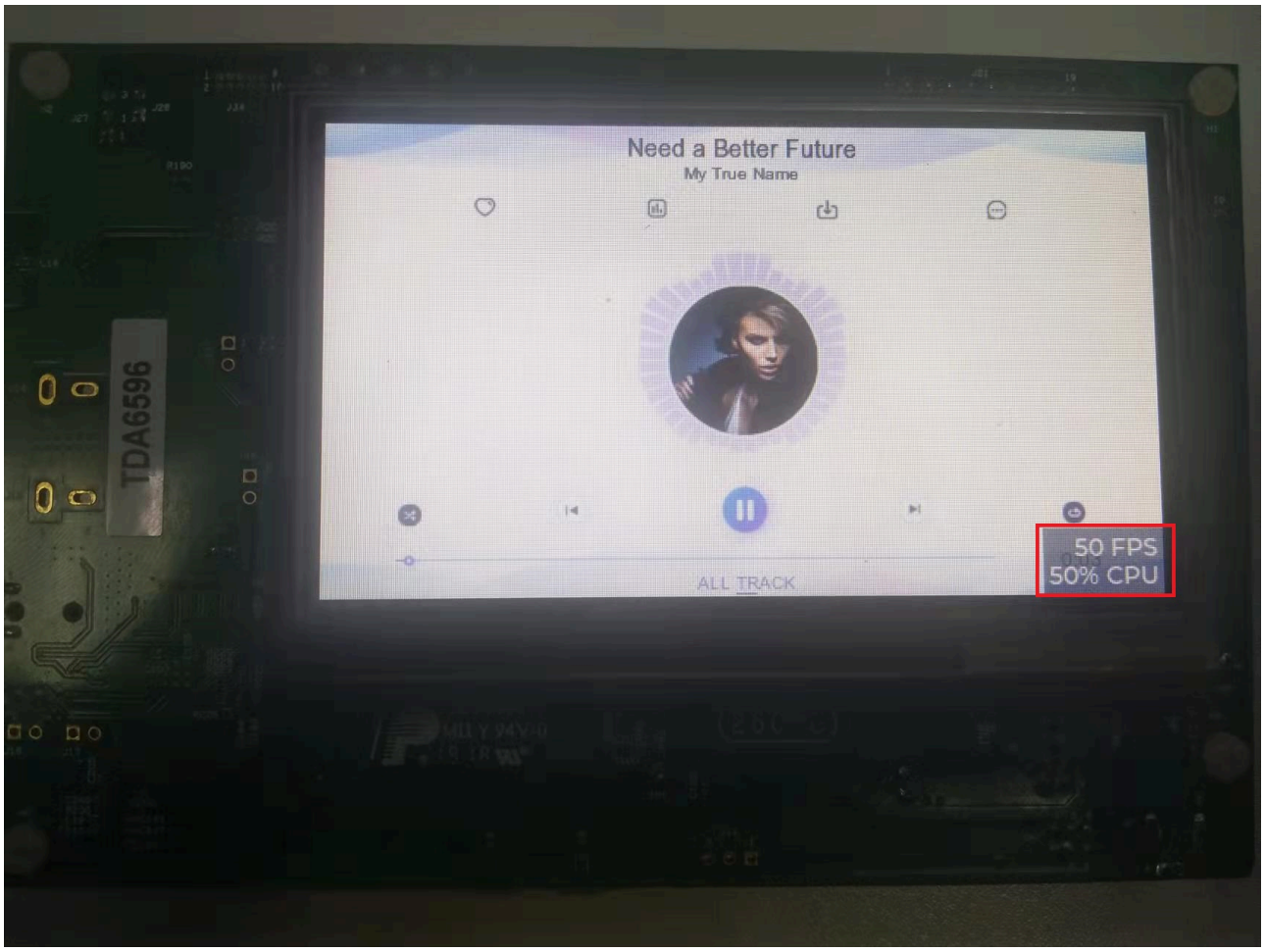

Figure 82. Checking real-time performance

5.3.2 Tips to improve the performance

Here is a summary of tips to get a good FPS performance using LVGL:

- **Use hardware acceleration**
  The capability of a board with hardware acceleration (PXP or VGLite) is often higher than a board without. Consider using a board with hardware acceleration. For details, see LVGL hardware acceleration.
- **Use Internal SRAM**
  The SRAM has better performance than other RAM. If a board has enough SRAM, the SRAM is a preferred place to store the frame buffers and other important data.
- **Use suitable C library**
  The Newlib library has good memcpy performance than the NewlibNano library. The Newlib library is preferred for applications with lots of data copy.
- **Use suitable compiler optimization level**
  In general, the -O2 and -O3 have better performance than other optimization level. GUI Guider can update the optimization level used in the demo example project.
  Note: ***Balance*** *means the -O2 option,* ***Size*** *means the -Os option,* ***Speed*** *means the -O3 option.*
- **Only redraw the changed things**
  Make sure that you only invalidate necessary parts of the display.
- **Adjust display refresh period**

The display refresh rate is a hard limit for your frame rate. In general, the frame rate is better when the display refresh period is lower. If the refresh rate of the display is 60 Hz, the refresh period is 1 s / 60 = 0.01667 s = 16.67 ms. GUI Guider supports updating the refresh period.

5.3.3 Improve the performance for i.MX RT boards

This section provides information on how to improve the performance on i.MX RT595 when working with MCUXpresso IDE.

5.3.3.1 Prerequisites

Design a GUI application using the GUI Guider and port the generated LVGL C source file to the template project imported by MCUXpresso IDE.

5.3.3.2 Improve the performance

The following are several performance optimization methods, which can be selected according to the actual needs.

- To use the Release build configuration, -O2 optimization level, and Newlib library, update the MCUXpresso setting. For details, see the MCUXpresso IDE documentation.
- To change the display refresh period, update the following line in `source/lv_conf.h`.

```
#define LV_DISP_DEF_REFR_PERIOD      30      /*[ms]*/
```

  For example, if the refresh rate of the display is 60 Hz, the value can be set to 16.67.
- Enable the hardware VGLite acceleration by changing the following line in `source/lv_conf.h`.

```
#define LV_USE_GPU_NXP_VG_LITE   0     // change to 1 to enable VGLite.
```

- If the NXP "G1120B0MIPI" MIPI Circular Display is selected, the frame buffer can be placed in SRAM. You can update the following lines in `board/display_support.h`.

```
#define DEMO_BUFFER0_ADDR 0x28000000U // i.e. Change to 0x20000000U
#define DEMO_BUFFER1_ADDR 0x28200000U //i.e. Change to 0x20100000U
```

- If the NXP "G1120B0MIPI" MIPI circular display is selected and few images are used, the image arrays can also be placed in the SRAM. To place the image array in SRAM, you can add the following macro definition in `source/lv_conf.h` first.

```
#define LV_ATTRIBUTE_IMG_CONST
__attribute__((section("DataQuickAccess")))
Then updated the C array definition in the Image C source files to add above
 macro:
const LV_ATTRIBUTE_IMG_CONST LV_ATTRIBUTE_MEM_ALIGN
LV_ATTRIBUTE_LARGE_CONST LV_ATTRIBUTE_IMG_XXXXXXXX uint8_t xxxxxxxxxxxx_map[] =
 { … },
LV_ATTRIBUTE_IMG_CONST const lv_img_dsc_t xxxxxxxxxxxxxxxxxx = { … },
```

5.4 External storage

This section describes how to use external storage (SD card) in the GUI Guider project.

5.4.1 SD card

This chapter describes how to invoke the SD card.

5.4.1.1 Image

The images (BIN, BMP, JPG, PNG) can be stored on an SD card and used by image-related widgets (imgbtn, image, Aimg, 3Dimg). The images are decoded in runtime.

The JPG (SJPG) format image cannot be zoomed and rotated when using external storage to runtime decode. If necessary, use a PNG image instead.

5.4.1.1.1 Prerequisites

Connect the supported device to the host by a USB cable.

5.4.1.1.2 Project template

There is an application template for supported boards. The usage tips are displayed on the screen.

1. Create a project by selecting one supported board and the "SDcardStorage" application template.

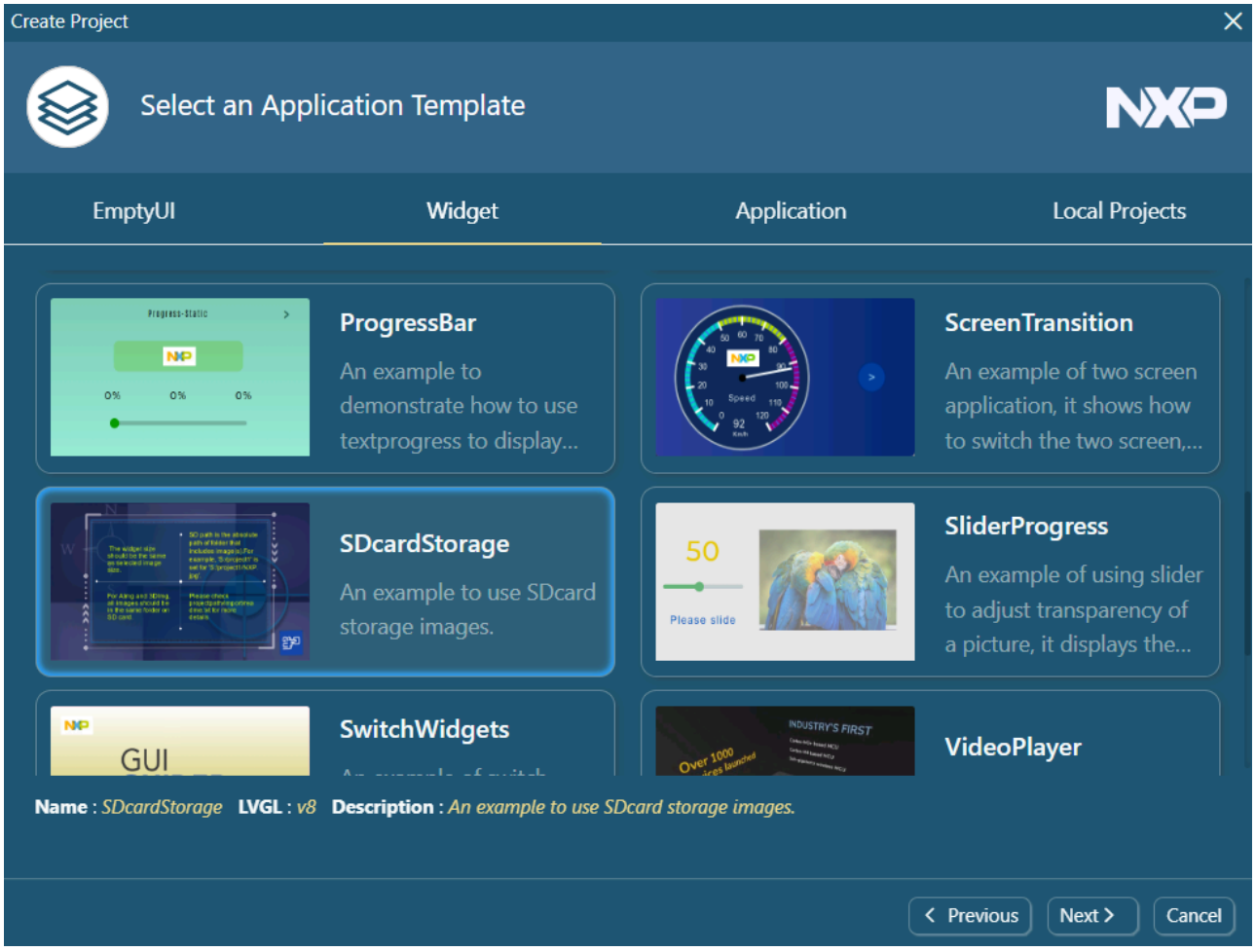

Figure 83. Create a project

2. The image files are located in the import folder.

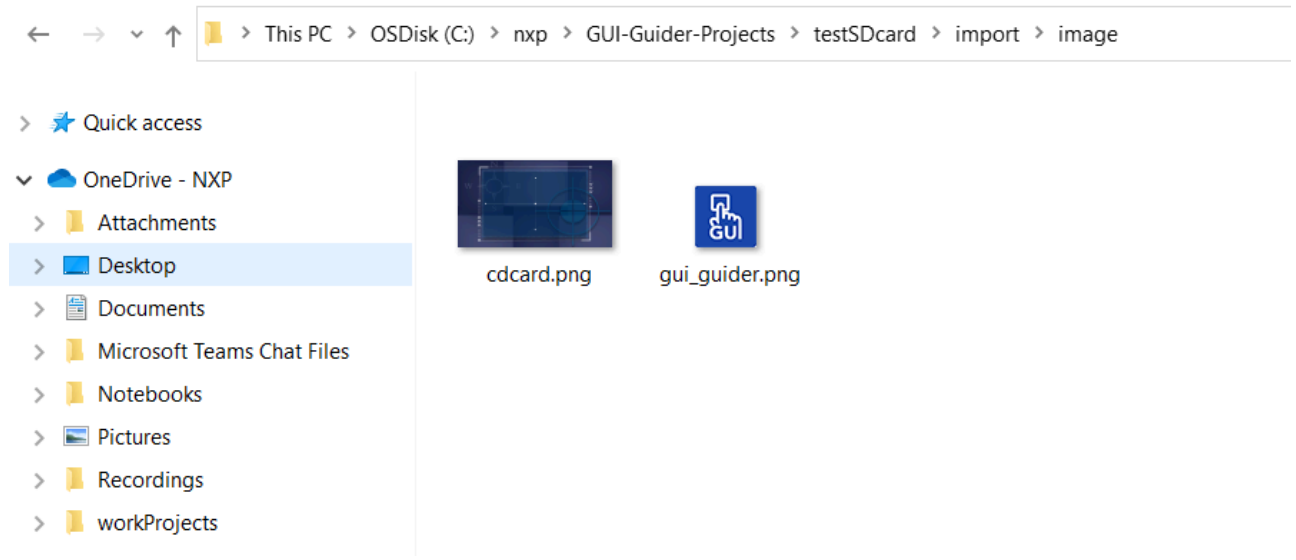

Figure 84. Import folder

3. The SD card storage can be enabled in the attribute setting window of image-related widgets.

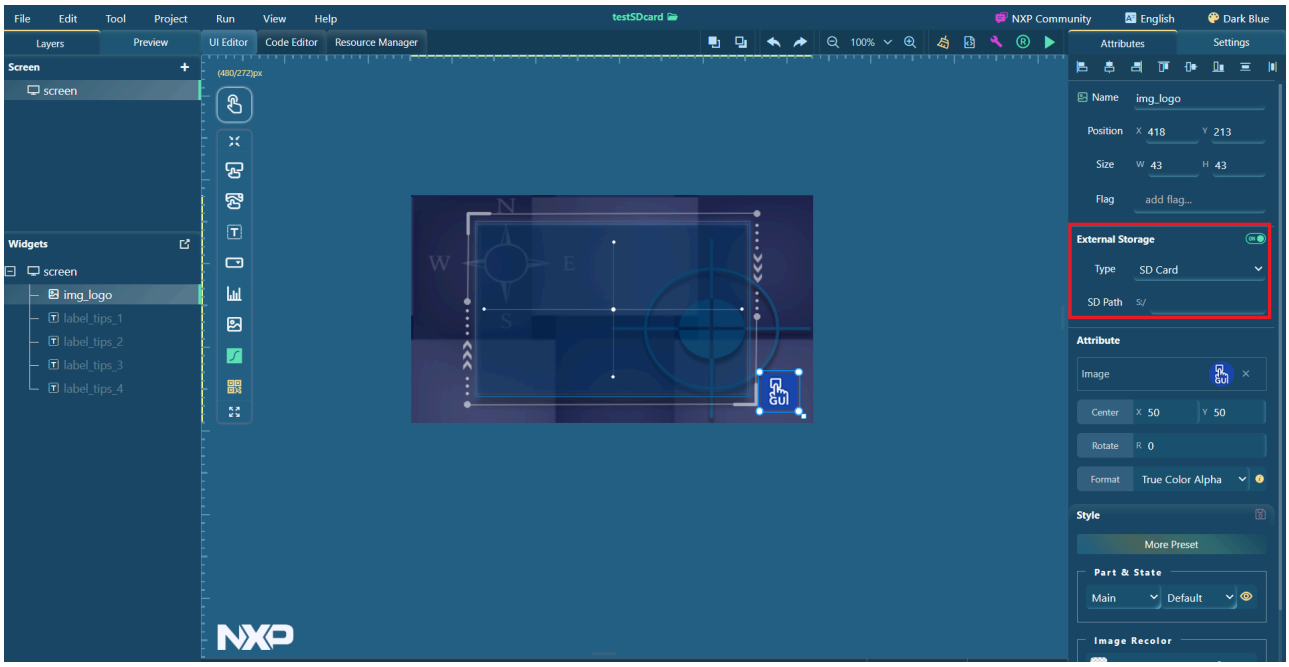

Figure 85. Attribute setting window

Copy the images into the SD card of the MCU device. If SD card storage is enabled, keep the SD path the same as the actual file folder on SD.

Note: *For the LVGL file system, use single "/" as a path separator.*

5.4.1.1.3 Customized project

When using image-related widgets, follow the steps to store images on the SD card.

1. Drag and drop the widget into the editor and set the attributes.

2. Open project folder > import. Confirm one or more image files that you want to save in SD, and copy those images to the SD card of the MCU device.
   Note: *All image files of Aimg and 3Dimg must be in the same folder on the SD card.*
3. Enable "External Storage" in the attributes setting window, and input the absolute path of the folder that includes one or more images.
4. Insert SD card to MCU device. To deploy the application on the board, click **Run** > **Target**.

5.4.1.2 Video

The format of the input video must be *.h264. The demo application implements video play on NXP MCU by using the LVGL library.

5.4.1.2.1 Prerequisites

- MCUXpresso 11.8.0/IAR 9.40.1/Keil MDK 5.38
- Connect the supported device to the host by a USB cable.

5.4.1.2.2 Prepare H264 file

To prepare an H264 video file, perform the following steps:

1. Install FFmpeg on your host.
2. Use the following command to convert the *.mp4 video file to an *.h264 video file.

```
$ ffmpeg -i input.mp4 -vf scale=480:272 -c h264 output.h264
```

3. To get the best play effect, make the *scale* identical to the expected size in the GUI application.
4. Load the *.h264 video on the SD card of the MCU device.

5.4.1.2.3 Create project with video player demo template

To create a project with the video player demo template, perform the following steps:

1. Go into the import folder and locate the `demo.h264` file.

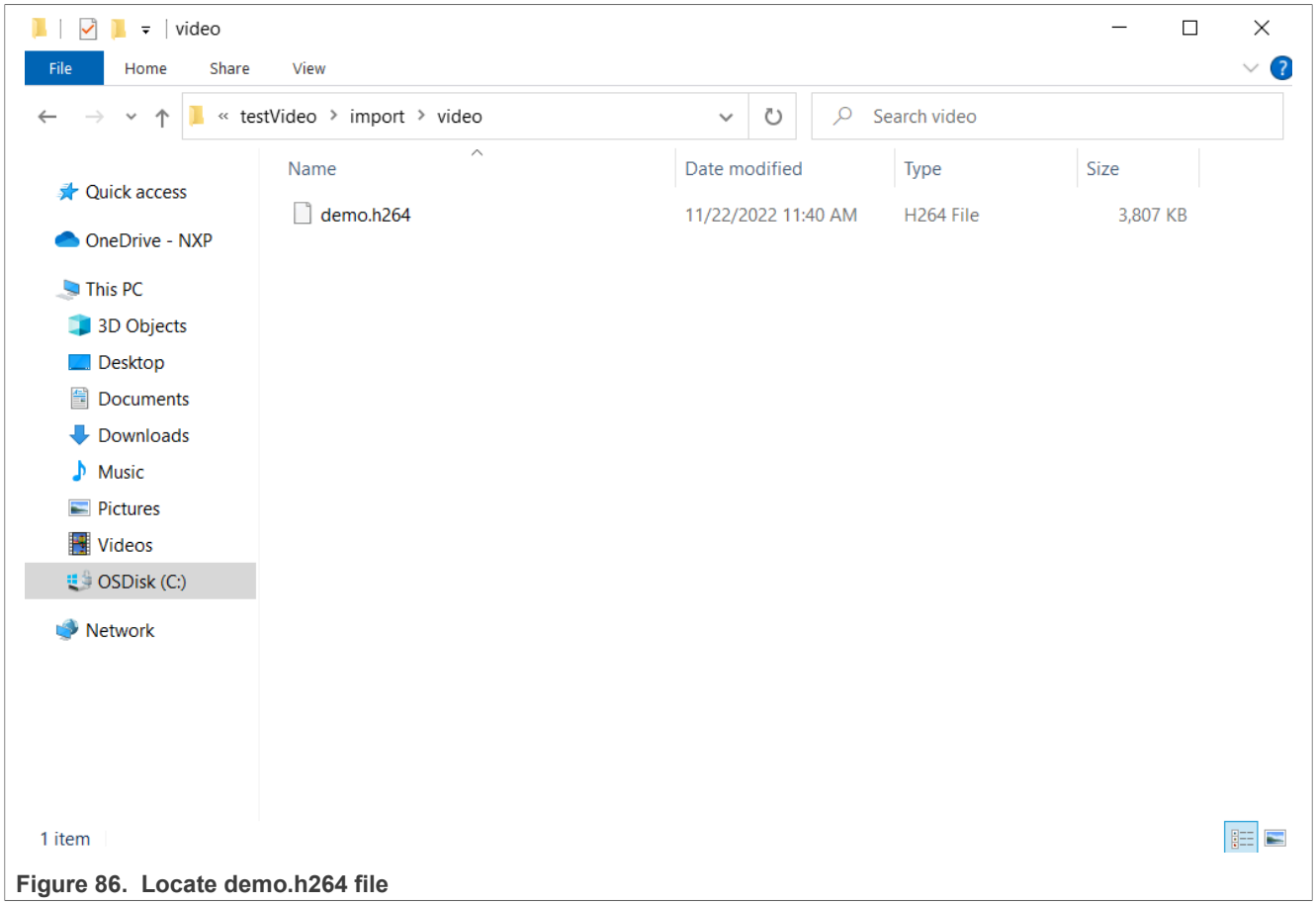

Figure 86. Locate demo.h264 file

2. Replace the `demo.h264` file from the SD card of the MCU device with the *.h264 file generated by yourself.
3. To load the code into your device, click **Run** > **Target** > **MCUXpresso**. After finishing, you can play this demo video on display.

Note: *Because the conversion of YUV420 to rgb565 at the board end is through PXP, PXP line selection cannot be enabled in the video demo in the GUI Guider.*

5.4.1.2.4 Customize project

The name and path of the *.h264 video file can be changed in `custom.c`. Different variables are used to set the simulator and target.

5.4.2 QSPI flash

To provide maximum flexibility, LVGL supports the following image sources:

- A variable in code (a C array with the pixels).
- A file stored externally (for example, on an SD card or QSPI flash).
- A text with symbols.

An attractive GUI is reliant upon well-designed images. The more complex the GUI is, the more these assets are required. Building the images into firmware is impossible if the selected MCU does not have adequate flash storage. Therefore, storing the images externally is a way to fit this case with reduced firmware size.

To use an external image, you must use LVGL's file system module and register a driver with some functions for the basic file operation. Go to the File system to learn more. Currently, GUI Guider supports the following:

- Images with PNG/JPG/BMP/binary format on an SD card with FatFs file system
- Images with binary format on QSPI flash with rawfs file system

As a demo, this document mainly focuses on putting images on IMXRT1060EVK QSPI flash.

5.4.2.1 Add image widget and set property

To add an image widget and set its properties, perform the following steps:

1. Click **img** in the **Widgets** tab and a new image widget is added into the workspace. Set "External Storage" as "on" and make sure "flash" type is selected.

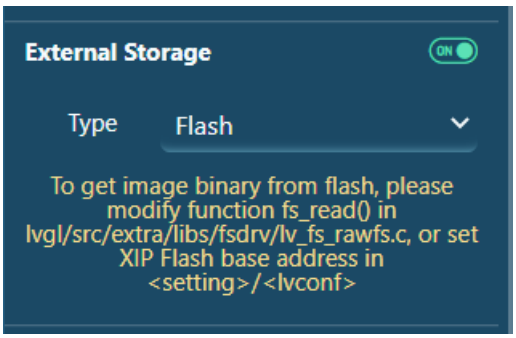

Figure 87. Set external storage

2. Import the local image file and set it in the image widget attribute.

5.4.2.2 Build and deploy

Connect the supported device to the host by a USB cable. To build and deploy, perform the following steps:

1. Select **Image Binary in XIP flash** in **Setting** > **lvConf**, and set "Base Address" as **0x60600000**.

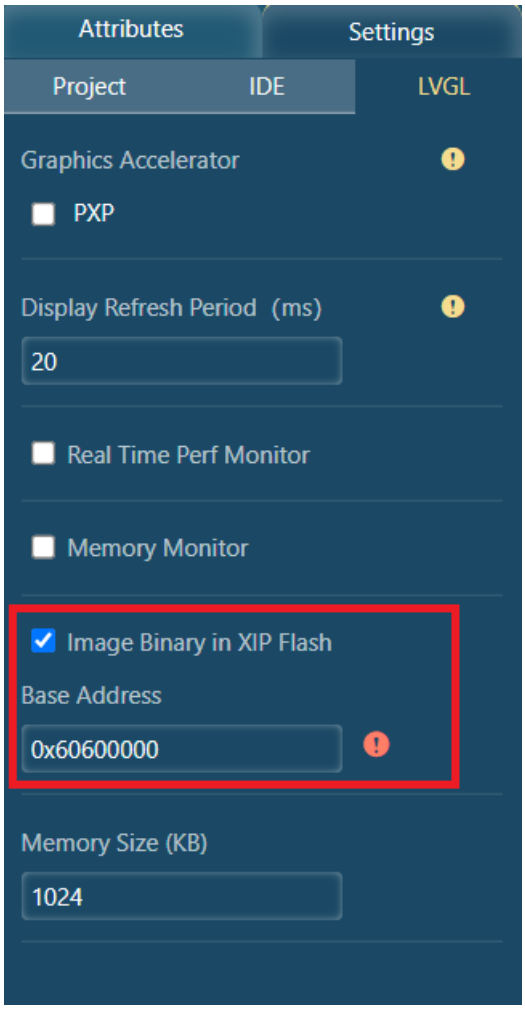

Figure 88. Set Base Address

Note: *In computer science,* ***execute in place (XIP)*** *is a method of executing programs directly from long-term storage. For this to work, one requirement is that the storage must provide a similar interface to the CPU as regular memory. In this demo, we put the image binary on XIP flash, which can be accessed directly via a base address. This is just for the demonstration of how external images are supported. It is more efficient and simple to link the image as a C array with the pixels in the firmware.*

Note: *Another real case in which the flash storage cannot be accessed directly via a base address is covered in the subsection below.*

Note: *IMXRT1060EVK ships an 8 MB Quad SPI flash, with the address from 0x60000000 to 0x607FFFFF. In this demo, we put the image binary to flash starting at address* ***0x60600000****, leaving 2 MB space for the image binary and 6 MB space for the firmware. You can adjust it based on the size of the image binary and firmware.*

2. Generate source code, build, and deploy. We choose "Keil" as an example here.
   Note: *There is no output as the image binary is not programmed to the flash until now.*
3. Program the merged image binary to flash at address **0x60600000**. The image binary can be found at `<project path>/generated/images/mergeBinFile.bin`.

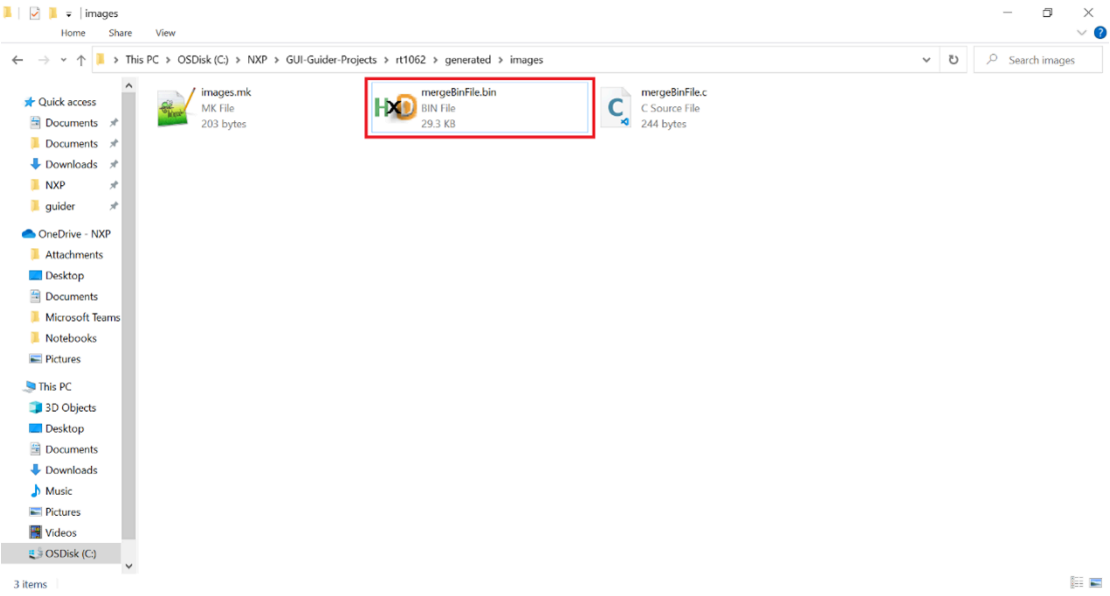

Figure 89. Program the merged image binary

4. Reset the board. The image widget should be shown normally.

5.5 Porting (RT)OS

This section lists the steps to port GUI APP to (RT)OS.

5.5.1 Zephyr

This section describes how to port the LVGL C source file generated by GUI Guider for Zephyr.

5.5.1.1 Set up Zephyr build environment

To set up Zephyr build environment on build machine, perform the steps below. For more details, refer to Zephyr getting started guide.

1. Install chocolatey (https://chocolatey.org/install).
2. Open the `cmd.exe` window as Administrator. To do so, press the Windows key, type `cmd.exe`, right-click the result, and choose "Run as Administrator".
3. To avoid having to confirm the installation of individual programs, disable global confirmation.

```
choco feature enable -n allowGlobalConfirmation
```

4. Use choco to install the required dependencies.

```
choco install cmake --installargs 'ADD_CMAKE_TO_PATH=System' choco install
 ninja gperf python311 git dtc-msys2 wget 7zip
```

5. Install pyocd.

```
python3 -m pip install -U pyocd
```

6. Close the window and open a new `cmd.exe` window as a regular user to continue.

5.5.1.2 Get Zephyr and install Python dependencies

To download Zephyr and install Python dependencies, perform the following steps:

1. Install west as Administrator.

```
pip3 install -U west
```

2. Get the Zephyr source code.

```
west init zephyrproject
cd zephyrproject
cd zephyr
git checkout origin/v3.5-branch -B v3.5-branch
cd ..
west update
```

3. Export a Zephyr CMake package. It allows CMake to load boilerplate code required for building Zephyr applications automatically.

```
west zephyr-export
```

4. Zephyr's `scripts\requirements.txt` file declares additional Python dependencies. Install them with pip3.

```
pip3 install -r %HOMEPATH%\zephyrproject\zephyr\scripts\requirements.txt
```

5.5.1.3 Install Zephyr SDK

To install Zephyr SDK on build machine, perform the following steps:

1. Open the `cmd.exe` window.
2. Download the latest Zephyr SDK bundle.

```
cd %HOMEPATH%
wget https://github.com/zephyrproject-rtos/sdk-ng/releases/download/v0.16.3/
zephyr-sdk-0.16.3_windows-x86_64.7z
```

3. Extract the Zephyr SDK bundle.

```
7z x zephyr-sdk-0.16.3_windows-x86_64.7z
```

Note: *Extract the Zephyr SDK bundle at one of the following locations.*

```
%HOMEPATH%
%PROGRAMFILES%
```

The Zephyr SDK bundle contains the zephyr-sdk-0.16.3 directory and, when extracted under `%HOMEPATH%`, the resulting installation path is `%HOMEPATH%\zephyr-sdk-0.16.3`.

4. Run the Zephyr SDK bundle setup script.

```
cd zephyr-sdk-0.16.3
setup.cmd
```

Note: *Run the setup script once after extracting the Zephyr SDK bundle.*
Note: *If you relocate the Zephyr SDK bundle directory after the initial setup, rerun the setup script.*

5.5.1.4 Design GUI and export code by GUI Guider

To export the source code of GUI application designed by GUI Guider to Zephyr, perform the following steps:

1. Design GUI application using GUI Guider.
2. To generate the GUI source code, click the "Generate Code" button.
3. Export the source code of GUI application to a folder under zephyr workspace. For example, `C:\Users\user1\zephyrproject\gui_guider_demo`.

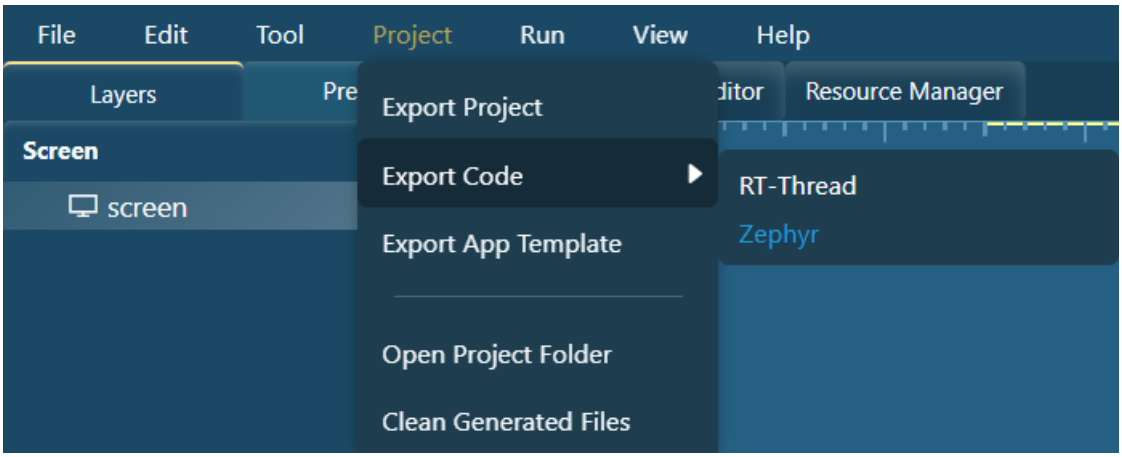

Figure 90. Export source code

5.5.1.5 Build and deploy Zephyr image

To build and deploy Zephyr images on target board, run the following commands.

```
> west build -c -p always -b mimxrt1050_evk C:\Users\user1\zephyrproject
\gui_guider_demo
> west flash --runner pyocd
```

5.5.2 RT-Thread

To port the LVGL C source file generated by GUI Guider to the RT-Thread project, see the following sections.

5.5.2.1 Prerequisite

- Keil v5.35 or newer.
- Latest GUI Guider GA.
- Connect i.MX RT1060 to the host with a USB cable.

Note: *In the working environment, all paths are not allowed to have Chinese characters or spaces.*

5.5.2.2 Install Git

Git supports the software package management. Download Git from https://git-scm.com/downloads. Install and add the install path into the system environment variable PATH.

5.5.2.3 Configure the Env tool

To configure the Env tool, perform the following steps:

1. Download the Env tool: windows-1.4.1.7z.
2. Extract the file `env-windows-1.4.1` to a local folder. For example, `D:\rt-thread\`.
3. In the env directory (`D:\rt-thread\env`), run `env.exe`. If it fails to open, use `env.bat`.
4. Register env utility in the right-click menu.

Figure 91. Click settings

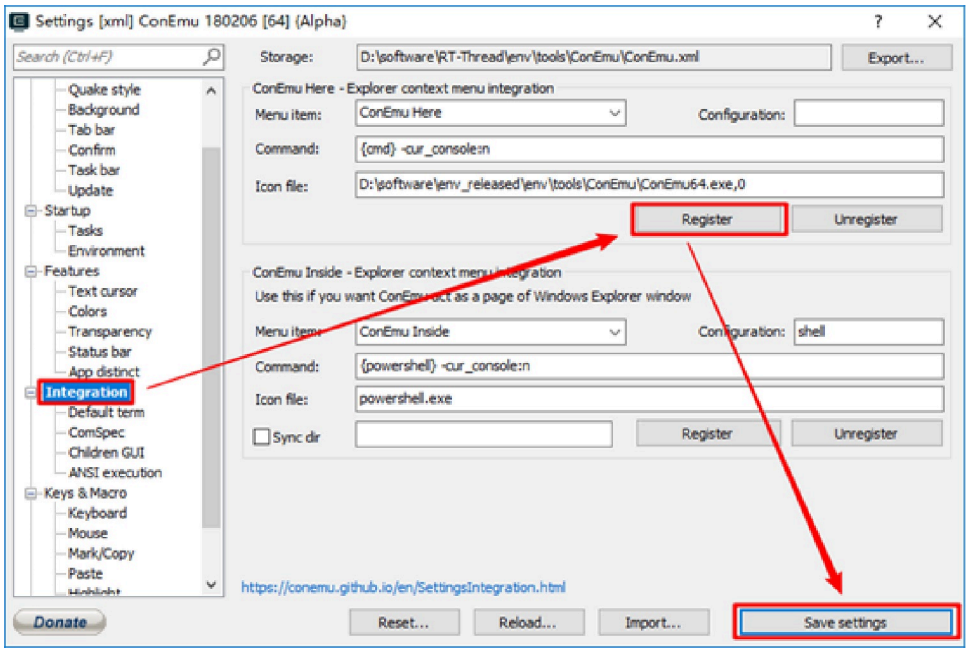

Figure 92. Save settings

5.5.2.4 Download RT-Thread and apply patches

To download RT-Thread, perform the following steps:

1. Go to the root folder of RT-thread. For example, `D:\rt-thread\`.
2. Run Git clone https://github.com/RT-Thread/rt-thread.git to download RT-thread source code. Then, run below commands:

```
$ cd rt-thread
$ git reset --hard v5.0.2
$ git cherry-pick a275a92b4af423faef812699c133a5243c68f6d2
```

.

3. Go to the i.mxrt1060 bsp folder (`D:\rt-thead\rt-thread\bsp\imxrt\imxrt1060-nxp-evk`). Right-click the window and select **ConEmu Here** to open env console.

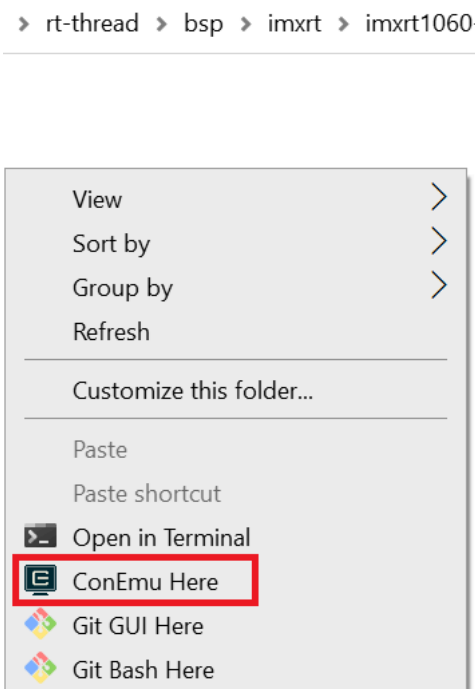

Figure 93. ConEmu Here option

5.5.2.5 Enable GUI demo project in RT-Thread

To enable a GUI demo project, perform the following steps:

1. In the env console, run the below commands:

```
$ pkgs --upgrade
$ menuconfig
```

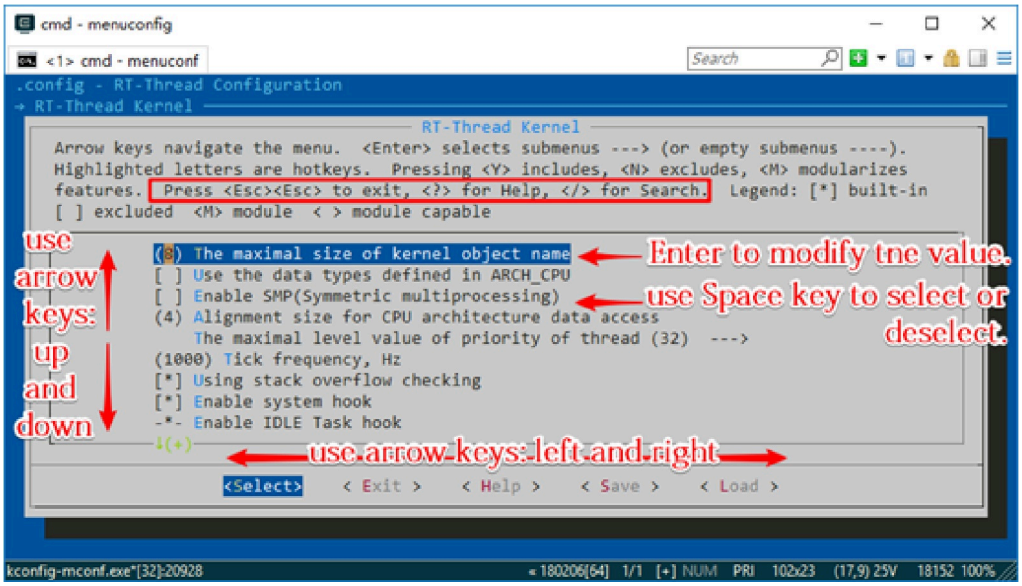

Figure 94. menuconfig

2. Enable LVGL GUI Guider support.

Location:
- Hardware drivers config
- Onboard peripheral drivers
- Enable LVGL for LCD
- Support NXP GUI Guider

3. Choose LVGL-8.3.10 release from `RT-Thread online packages ---> multimedia packages ---> LVGL: powerfull and easy-to-use embeded GUI library ---> LVGL (official): Light and Versatile Graphic Library ---> Version --->`.
   Note: *To avoid build error, deselect the "Enable LVGL demo" option. The option location is the same with "Support NXP GUI Guider".*
4. To download the selected packages, run `pkgs --update`.

5.5.2.6 Export source of GUI designed by GUI Guider

To export the source, perform the following steps:

1. Use GUI Guider to design a GUI application.
2. Click **Generate Code** in the GUI Guider IDE.
3. Click **File** > **Export Code** > **RT-Thread** on menu bar to export source code of GUI designed by GUI Guider to a template project folder (`D:\rt-thread\rt-thread\bsp\imxrt\imxrt1060-nxp-evk\packages\gui_guider_demo-latest\`).
   Note: *Remove the previous "custom" and "generated" folders before exporting.*

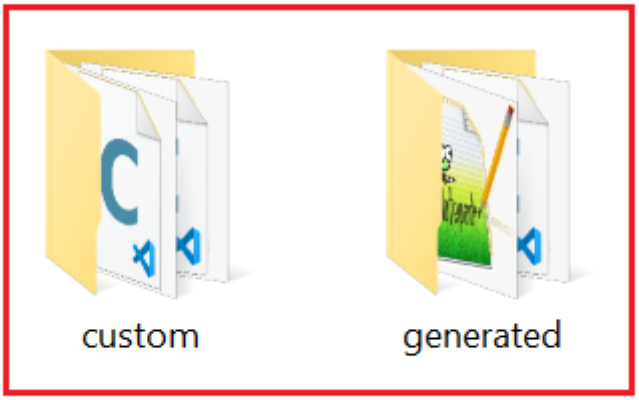

Figure 95. Template project folder

4. Run `scons --target=mdk5 -s` to generate/update Keil project file `project.uvprojx`, which is available at `D:\rt-thread\rt-thread\bsp\imxrt\imxrt1060-nxp-evk`.

5.5.2.7 Build and compile

To build and compile, perform the following steps:

1. Double-click Keil project file `project.uvprojx` in `D:\rt-thread\rt-thread\bsp\imxrt\imxrt1060-nxp-evk` and rebuild all the files.
   Note: *If the following error appears, update the corresponding source file to replace `lvgl/lvgl.h` with `lvgl.h`.*
   Error: src.c(10): error: 'lvgl/lvgl.h' file not found

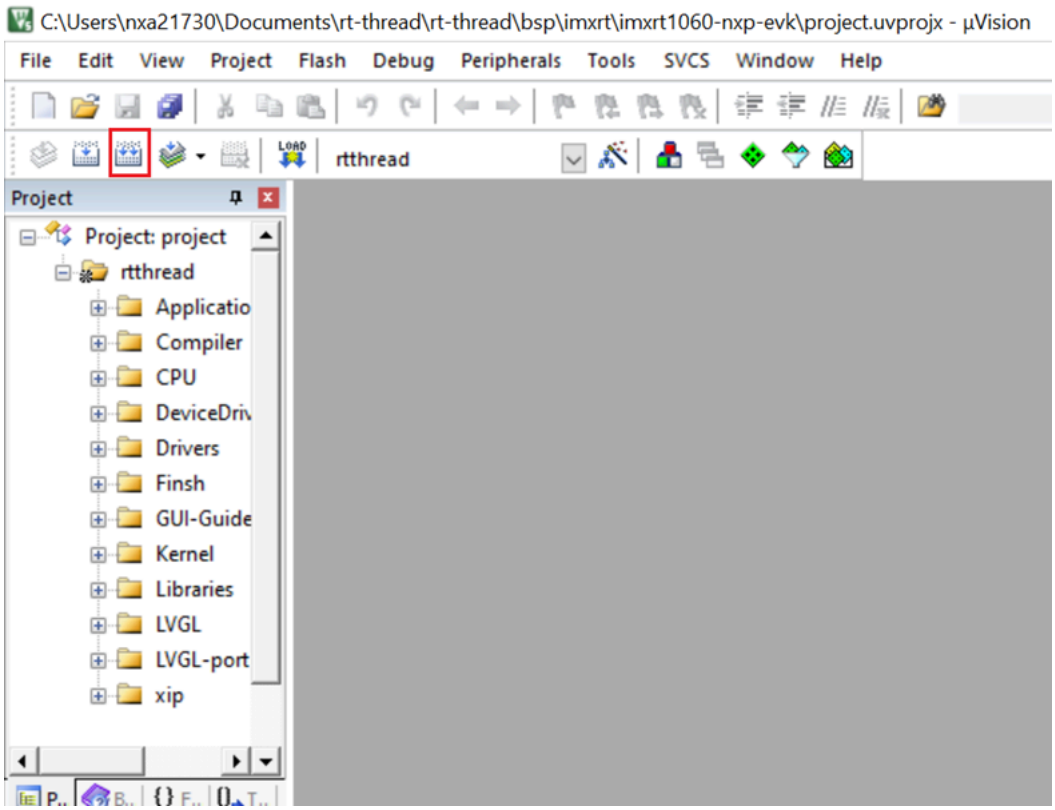

Figure 96. Rebuild Keil project files

2. Connect i.MX RT1062 to PC with a USB cable. Click "Download (F8)".

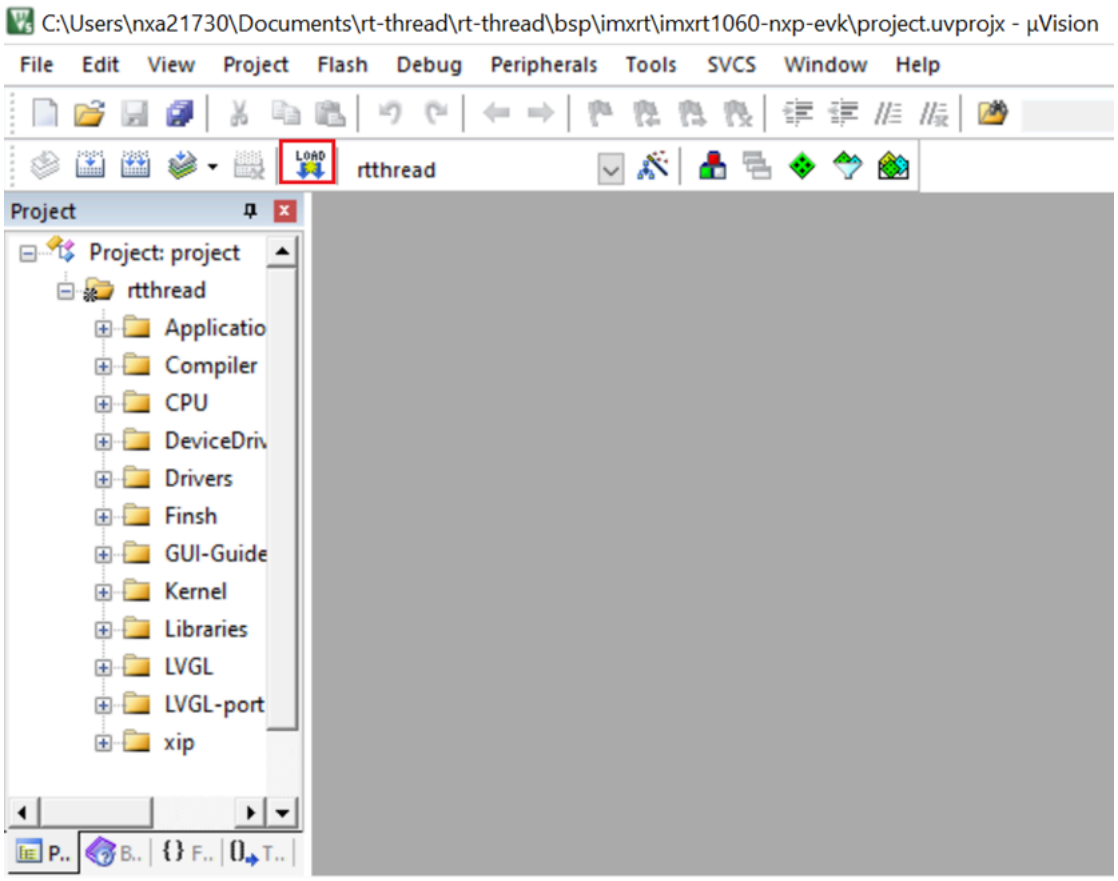

Figure 97. Click download or press F8 key

After performing the above steps, the GUI application designed by GUI Guider can be compiled in RT-Thread and run on i.MX RT1060 board.

5.5.2.8 Known Issues

If option **Event** > **load screen** > **Delete current screen** is enabled, then the PC hangs when switching between different screens. The workaround is to disable the **Delete current screen** when loading a new screen.

5.5.3 Yocto

The i.MX family Linux board support package (BSP) supports the Linux operating system (OS) on the i.MX application processors by using the Yocto project build environment.

GUI Guider supports generating app codes on the MCIMX93EVK board.

This chapter describes the steps to port the GUI application to Linux.

5.5.3.1 Prerequisite

NXP Linux Factory v6.1.55-2.2.0 or v6.1.1_1.0.0 release

5.5.3.2 Create a project

To create a project, perform the following steps:

1. When the GUI Guider is launched, click the **Create a new project** button from the Wizard, or select **File > New**.
2. Select LVGL v8.3.10 and MCIMX93EVK board template.
3. Select an application template or Empty UI.
4. Design your GUI application, then continue.

5.5.3.3 Build GUI application binary

GUI Guider provides two options to build the GUI application binary (gui_guider):

- Standalone build using the GUI Guider IDE.
- Export the Yocto layer and build using Yocto build environment.

Standalone build

In this way, GUI Guider IDE launches the Yocto toolchain to cross-compile the GUI application. When the application binary is built, user must copy it to pre-installed Yocto rootfs and execute it on the board.

Note: *Only Linux host is supported.*

1. Steps to build the Yocto toolchain are:
   a. Set up the Yocto build environment.
      i. Install the 'repo' utility.

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo
 > ~/bin/repo
$ chmod a+x ~/bin/repo
$: PATH=${PATH}:~/bin
```

      ii. Download the Yocto Project BSP.

```
$ mkdir imx-bsp
$ cd imx-bsp
$ repo init -u https://github.com/nxp-imx/imx-manifest -b <branch name>
 [ -m <release manifest>]
$: repo sync
```

   b. Setup Yocto build project.

```
$ MACHINE=imx93evk DISTRO=fsl-imx-xwayland source ./imx-setup-release.sh -
b bld-imx93evk
```

   c. Build the Yocto toolchain.

```
$ bitbake -c populate_sdk imx-image-multimedia
```

   d. Install the Yocto toolchain.
   For v6.1.55-2.2.0 release:

```
$ sudo sh ./fsl-imx-xwayland-glibc-x86_64-imx-image-multimedia-
armv8aimx93evk-toolchain-6.1-mickledore.sh -y
```

   For v6.1.1-1.0.0 release:

```
$ sudo sh ./fsl-imx-xwayland-glibc-x86_64-imx-image-multimedia-
armv8aimx93evk-toolchain-6.1-langdale.sh -y
```

2. Install ninja utility on the build host.

```
$ sudo apt install ninja-build
```

3. Click menu **Project** > **Build** to start standalone build.

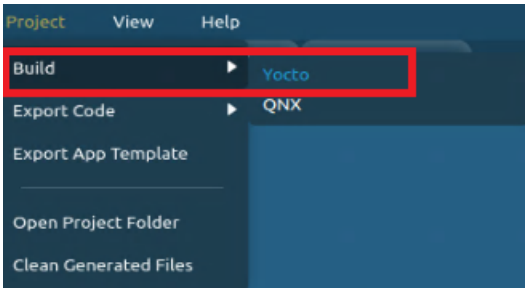

Figure 98. Click Build

4. Check build logs in log window, the gui_guider binary gets generated under *<project path>/build/*.

Export Yocto layer and build GUI application by Yocto

GUI Guider IDE exports a Yocto layer, which includes GUI application codes and related Yocto recipes. User must plug in the exported Yocto layer into an existing Yocto build environment, and use bitbake to build the gui_guider binary.

1. Export Yocto Layer
   a. Click **Generate Code** to generate code.
   b. Click menu **Project** > **Export Code** > **Yocto** to export Yocto layer.

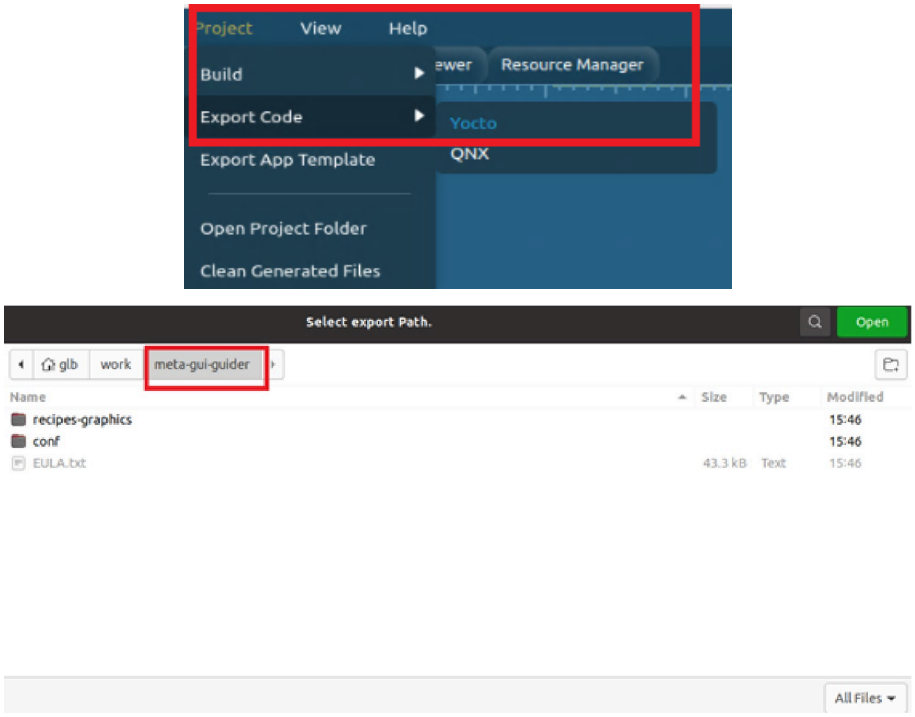

Figure 99. Export Yocto layer

Note: *"meta-gui-guider" folder is the Yocto layer which can be used as a plugin of Yocto SDK. Refer to the following section on how to build using Yocto.*

2. Build GUI application in Yocto.
   a. Copy the exported "meta-gui-guider" folder to the `<path>/imx-bsp/sources/` folder.
   b. Build using Yocto.
      i. If this is the first time to run the Yocto build, use the following steps:

```
$ MACHINE=imx93evk DISTRO=fsl-imx-xwayland source ./imx-setup-
release.sh -b bld-imx93evk
$ bitbake-layers add-layer ../sources/meta-gui-guider
$ echo "INHERIT += \"rm_work\"" >> conf/local.conf
$ echo "RM_WORK_EXCLUDE += \"gui-guider\"" >> conf/local.conf
$ bitbake gui-guider
```

      ii. If this is the subsequent build, use the following steps:

```
$ source sources/poky/oe-init-build-env bld-imx93evk/
$ bitbake gui-guider
```

Note: *The gui_guider binary gets generated in `<path>/imx-bsp/bld-imx93evk/tmp/work/armv8a-poky-linux/gui-guider/1.7.0-r0/images/` folder.*

5.5.3.4 Run GUI application on i.MX 93

The LVDS panel (**Part number: EV121WXM-N12-3GP0**) is used. You can select other supported panel to try the generated GUI application.

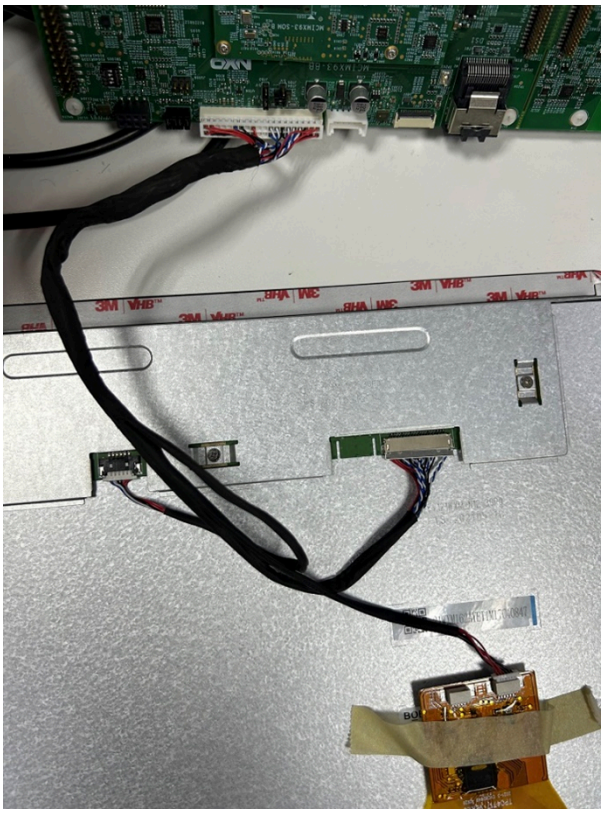

Figure 100. Run GUI application on i.MX 93

1. Deploy the image on SD card.

```
$ zstd -d -f imx-image-multimedia-imx93evk.wic.zst
```

Note: *If there is no* `zstd` *command, install using:* `sudo apt-get install zstd`.

```
$ sudo dd if=imx-image-multimedia-imx93evk.wic of=/dev/sdx bs=4M &&
sync
```

2. Insert SD on i.MX 93 and bootup board.
3. Replace default `dtb` with `lvds-panel dtb`.

```
$ cd /run/media/boot-mmcblk1p1
$ cp imx93-11x11-evk-boe-wxga-lvds-panel.dtb imx93-11x11-evk.dtb
```

4. Reboot the board and transfer gui_guider binary onto the board.
5. Run the GUI application.

```
$ ./gui_guider&
```

5.5.4 QNX

GUI Guider supports to design and build an HMI application for QNX RTOS.

5.5.4.1 Prerequisite

Install QNX SDP:

1. Download QNX Software Development Platform 7.1.
2. Install QNX SDP at **$HOME/qnx710**.
   - Windows 10: C:\Users\<username>\qnx710
   - Ubuntu 22.04: /home/<username>/qnx710

Note: *Currently GUI Guider searches QNX SDP in the above fixed path.*

5.5.4.2 Design HMI application

When the QNX SDP is installed, you can use GUI Guider to design an HMI application by following the steps described in the following sub-chapters.

5.5.4.2.1 Create a project

To create a project, follow the steps below:

1. Open a new project wizard.
2. Select LVGL 8.3.10 as the LVGL version and MCIMX93EVK as a board template.
3. Do the project settings and create a project.

5.5.4.3 Build image binary

GUI Guider provides two options to build the HMI application:

- Build the application directly by GUI Guider IDE.
- Export the generated source code and build a GUI application in QNX Software Development Platform.

5.5.4.3.1 Build by GUI Guider IDE

To build the HMI application using the GUI Guider IDE, follow the steps below:

1. To start build, click **Project > Build > QNX**.
2. To get the binaries, check build logs and open the project folder.

5.5.4.3.2 Export code and build application

This section describes exporting the code and building application.

5.5.4.3.2.1 Export code and build in QNX SDP

To export the code and build it in QNX Software Development Platform, perform the following steps:

1. To generate code, click **Generate Code**.
2. To export the codes, click **Project > Export Code > QNX**.
3. Select an empty folder for the codes to be exported to.
4. The structure of exported codes is as below:

```
|-- custom    # custom codes
|-- generated    # generated codes
|-- lvgl     # lvgl codes
`-- ports/qnx
    |-- build-qnx.bat  # build script for windows
    |-- build-qnx.sh  # build script for linux
    |-- main.c   # code entry for lvgl_demo
    |-- Makefile   # for building lvgl_demo binary
    |-- Makefile.lvgl  # for building liblvgl.so
    |-- qnx.c
    |-- qnx.h
    `-- README.md
```

5.5.4.3.2.2 Build application

This chapter describes how to build the application for QNX.

For Windows, run the following commands in the console:

```
# cd <exported codes folder>
# run ports\qnx\build-qnx.bat
```

For Ubuntu, run the following command on the terminal:

```
# run ports/qnx/build-qnx.sh
```

`liblvgl.so` and `lvgl_demo` are generated under folder build.

5.5.4.4 Run application on board

To run the application on target board, follow the steps below:

1. Ensure that the screen is not running.

```
slay screen
```

2. Enable mouse + keyboard and start screen server.

```
io-hid -dusb upath=/dev/usb1/io-usb-otg
screen -c /usr/lib/graphics/iMX93/graphics.conf
```

Note: *Check the `io-usb-otg` path. The default path is `/dev/usb/io-usb-otg`. Use `upath=parameter` for customized path.*
Note: *`io-hid` and `devi-hid` binaries might not be present on your target by default. You can find the binaries in the "qnx710" folder.*

3. Upload `liblvgl.so`, `lvgl_demo` to `/tmp` (or integrate to your QNX image).
Note: *When connected to the board using the WinSCP, select the FTP protocol. The user name and password are "qnxuser" and "qnxuser".*
4. Run the demo (in case of binaries, put under `/tmp`).

```
chmod a+x /tmp/lvgl_demo
LD_LIBRARY_PATH=/tmp /tmp/lvgl_demo
```

5.6 FreeMASTER debug in GUI Guider

FreeMASTER is a user-friendly real-time debug monitor and data visualization tool that enables runtime configuration and tuning of embedded software applications. GUI Guider can be a new option for embedded engineers to design the debug UI by drag and drop and use the C language for development work.

GUI Guider can remotely debug embedded application on target by simulator on host. It supports FreeMASTER variables binding, reading, and writing with widgets.

5.6.1 Prerequisite

- FreeMASTER IDE 3.2
- FreeMASTER project
- Target board

For more details on FreeMASTER, refer to *FreeMASTER for Embedded Applications* (document FMSTERUG).

5.6.2 Debug UI design

To create a project, refer to Section 2.3.1. Then, refer to the following sections to enable FreeMASTER settings and bind variables with widgets.

5.6.3 Enable FreeMASTER

Before binding variables with widgets, set up the FreeMASTER environment and enable the FreeMASTER setting in GUI Guider.

1. Enable the FreeMASTER setting by selecting the "FreeMASTER" submenu under the view menu.

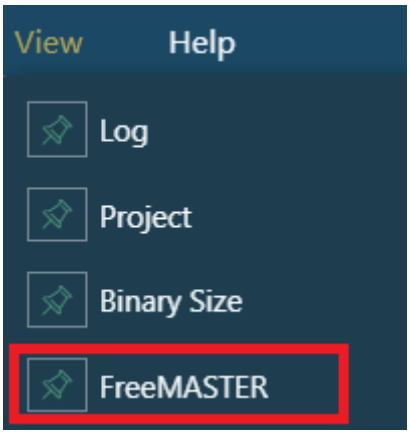

Figure 101. Enable FreeMASTER

2. Open the FreeMASTER setting window in the GUI Guider bottom bar.

Figure 102. Open the FreeMASTER setting window

3. Start the FreeMASTR IDE.
4. Click the "Link To FreeMASTER Server" button on the GUI Guider IDE, then confirm the FreeMASTER address and port.
5. Click the "Local FreeMASTER Project" to import the FreeMASTER project in the GUI Guider.

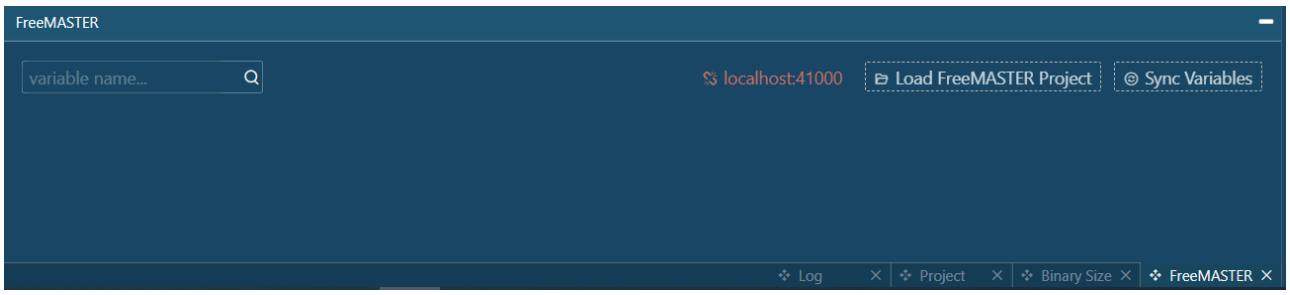

Figure 103. Import FreeMASTER project

Table 18 shows the description of function buttons in the FreeMASTER setting window.

Table 18. FreeMASTER operations supported in the GUI Guider

Function buttons	Description
Load FreeMASTER Project	Select the debug FreeMASTER project.
Sync Variables	Sync variables information of imported FreeMASTER project
localhost:41000	Support to link and unlink the FreeMASTER server

Note: *When there is a variable change in a FreeMASTER project, click "Sync Variables" to refresh the variables information in the GUI Guider.*

5.6.4 Binding variable with widget

GUI Guider supports to read values from variables and display in widgets and write a value from a widget to a variable. All widgets can bind a variable to write value to the variable. Table 19 lists widgets that support read value from a variable and display it.

Table 19. Widgets that support binding FreeMASTER variables

ID	Widget	Description
1	Arc	The value from a variable is displayed as the **value** of attributes.
2	Bar	The value from a variable is displayed as the **value** of attributes.
3	Chart	The value from a variable is displayed as **Dataset-> data_0,data_1…** of attributes.
4	Label	The value from a variable is displayed as the **Text** of attributes.
5	Meter	The value from a variable is displayed as **Scale_0->needles_0, needles_1…** of attributes.
6	Slider	The value from a variable is displayed as the **Init Value** of attributes.

There are two ways to find variables with widgets for value read and display. The following section describes how to bind a variable with a widget and display the variable value on this widget.

- Select a widget in the GUI editor. The  icon is displayed in the FreeMASTER setting window. To bind the variables with the selected widget, click this icon.

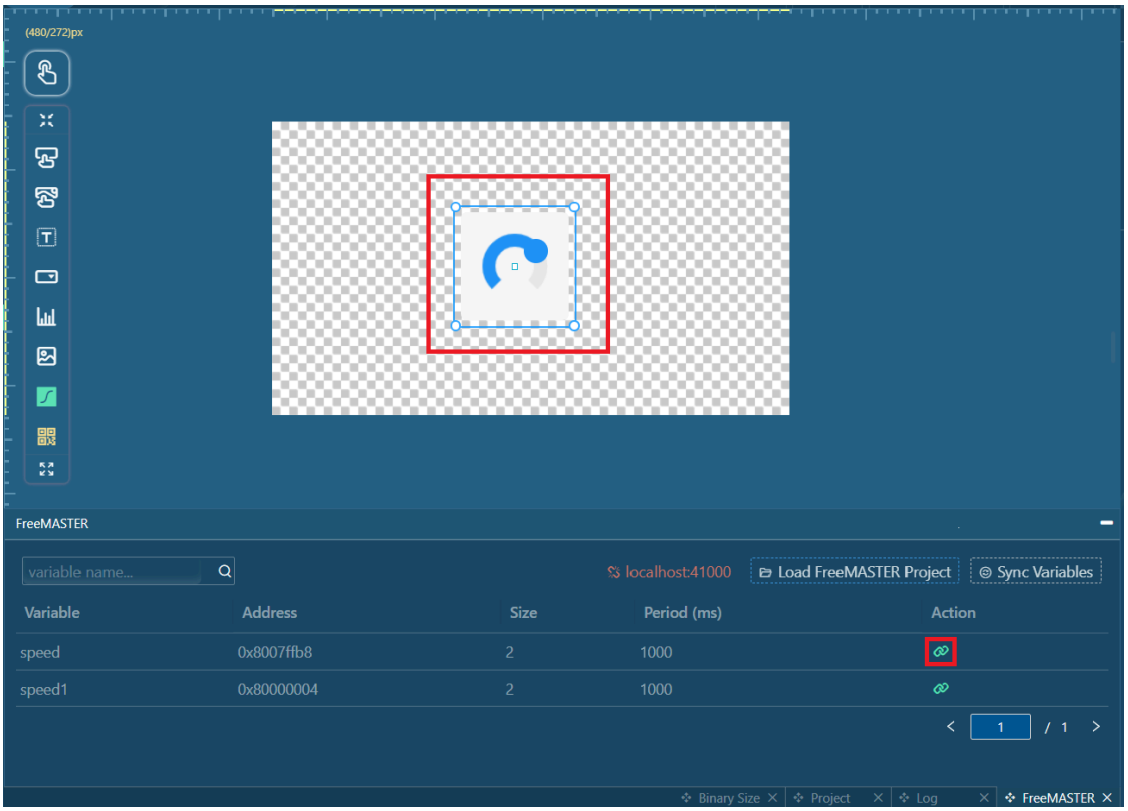

Figure 104. Select a widget

- Another way is to bind in the widget attribute setting window. The following section takes a label for an example to bind variable in the attribute setting window. To open the variable binding window, click the  icon in the "Text" attribute. Then, select a variable to bind with the current label widget.

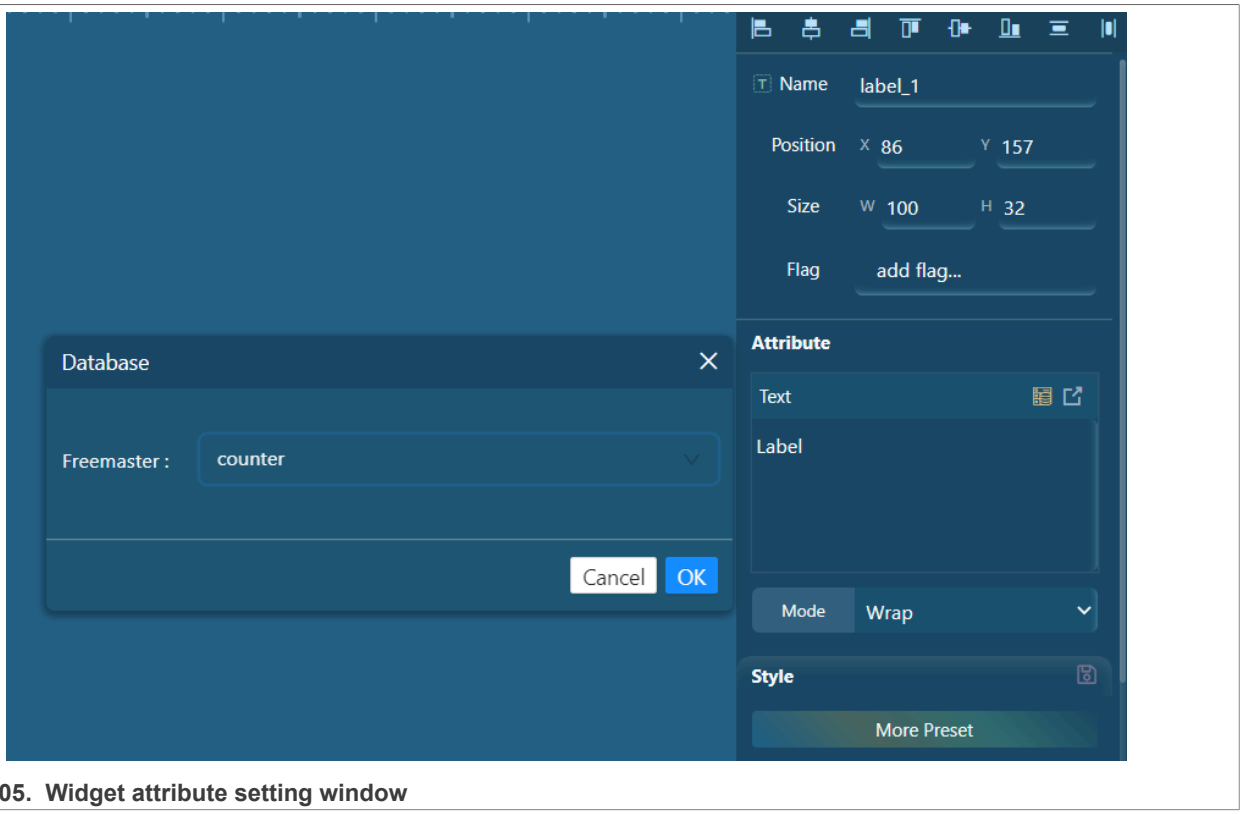

Figure 105. Widget attribute setting window

Note: *When the FreeMASTER function is disabled, the*  *icon is displayed. When clicking the icon, a variable list is emptied in a variable binding window.*

The variable writing is configured in event setting of widgets. The FreeMASTER action is available for all events. The following section describes how to bind a variable with a widget and write a value from a widget to a variable.

1. Select a widget in the GUI editor and open an event setting window. Then click the "value changed" event.
   - For switch checkbox widgets that have true and false states, the following window is opened. The action can be either set for each value separately or for the widget directly. Click each option of such widgets in the left window. The FreeMASTER setting window is opened.

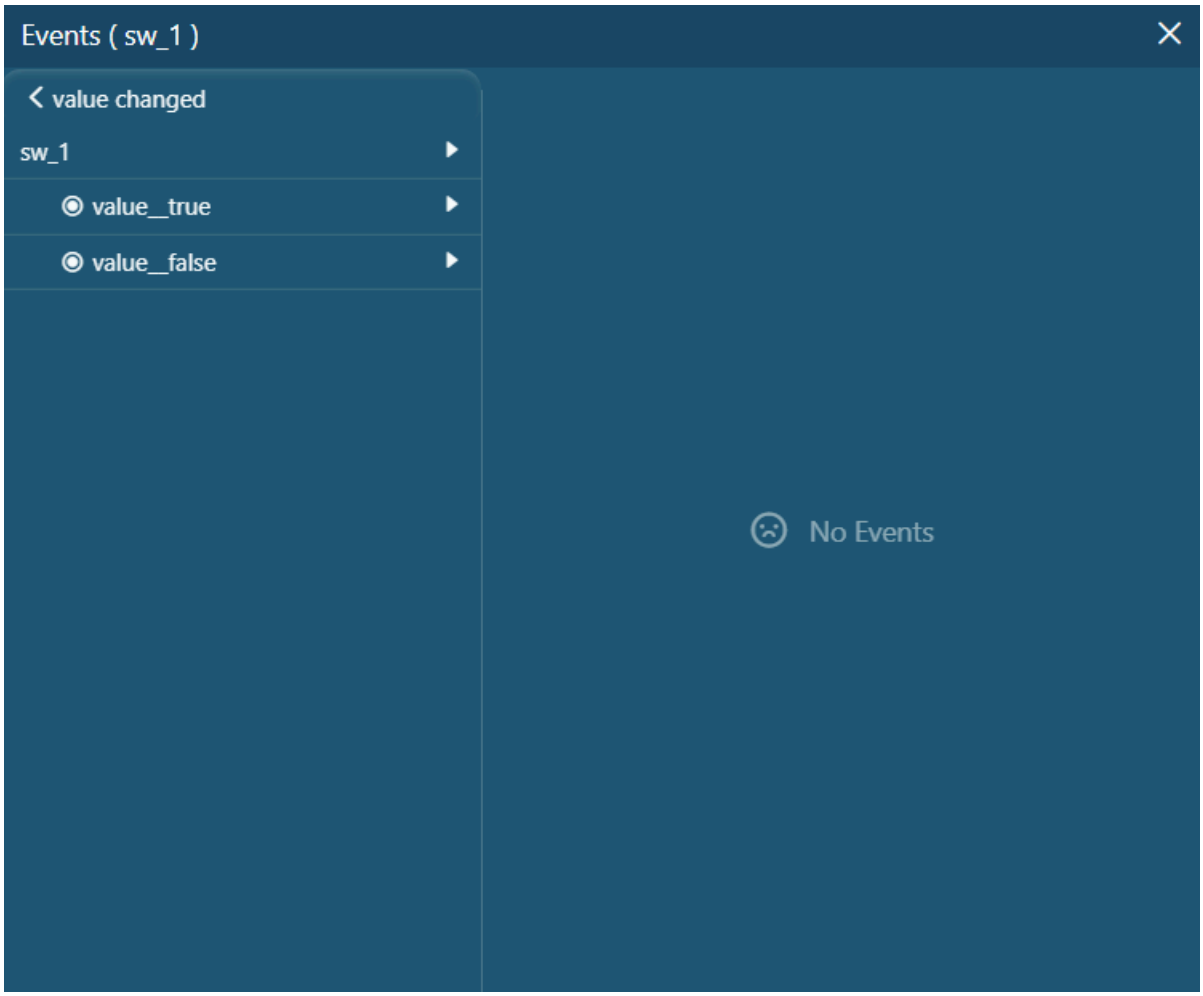

Figure 106. Event setting window

- For other widgets, when clicking an event, the following FreeMASTER setting windows is opened.

2. Select the target variable to bind for value writing. The value written to a variable can be either a fixed value or a dynamic value from the widget.

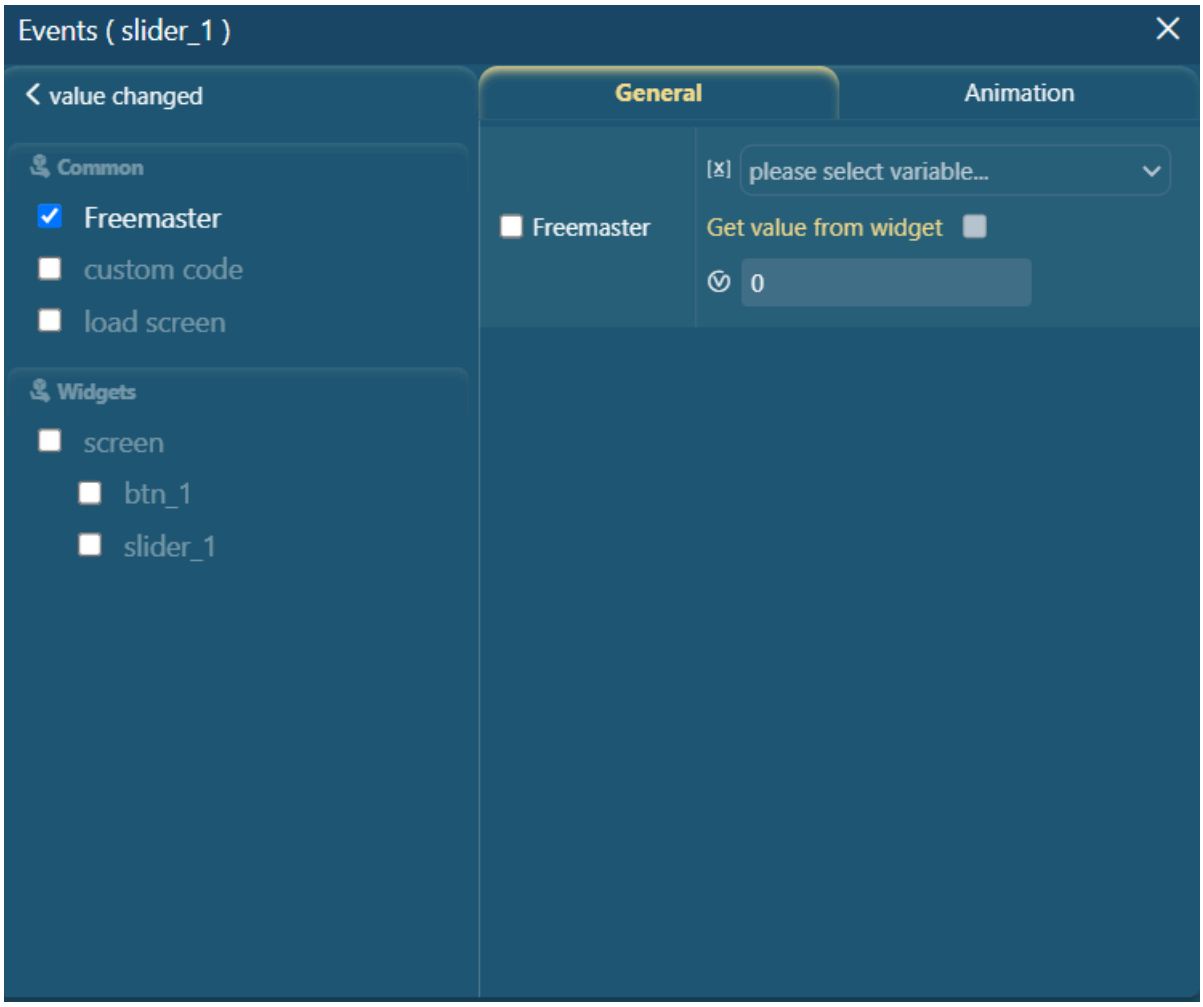

Figure 107. Select a target variable

Note: *The value '1' is written to a variable when the switch is in ON state and the checkbox is in checked state. The value '0' is written to a variable when the switch is in the OFF state and the checkbox is unchecked.*

When you complete the debug UI design and variable binding. Switch FreeMASTER IDE to click "GO" for starting the communication. This starts the data communication and runs a C simulator to start the debug of an embedded application on the target.

6 Tutorials

This chapter describes some frequently used user cases.

6.1 Interact with peripherals by custom code

For more information about GUI Guider peripheral interaction, refer to *GUI Guider Peripheral Interaction* (document AN13217).

6.2 Add custom attributes and styles after setup screen

This tutorial describes how to add custom attributes and styles for the loading screen.

On the screen attribute setting window, click the **custom C** button to launch the edit code page.

- The left side is to write your references and variables.
- The right side is to add custom style or attributes code. If you want to change the widget in the current screen, you can find the corresponding object using <ui> tag.

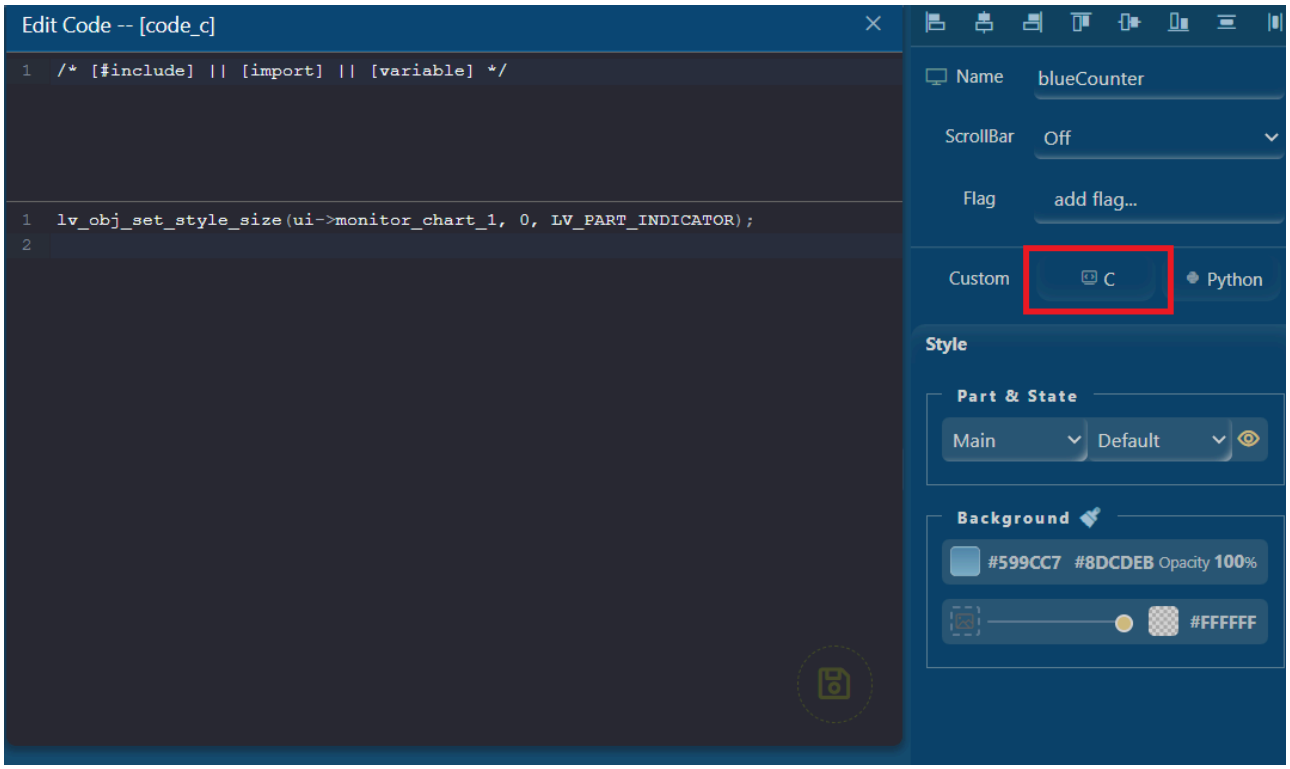

Figure 108. Cutom attributes and styles

6.3 Reuse GUI design on different boards and panels

You can create a new project with the local project, and select a different board and panel. For more information, see Section 2.3.2.

6.4 Rotate screen and widgets

This chapter describes how to rotate the panel of i.MX RT1060EVK from landscape mode to portrait mode using PXP. To support the rotate panel, we must update the flash display function in `sdk\Core\board\lvgl_support.c`.

Here is the example code to show how to update the flash display function:

```
#if DEMO_USE_ROTATE
#if LV_USE_GPU_NXP_PXP /* Use PXP to rotate the panel. */
lv_area_t dest_area =
{
    .x1 = 0,
    .x2 = LCD_HEIGHT - 1,
    .y1 = 0,
    .y2 = LCD_WIDTH - 1,
};
```

```
lv_gpu_nxp_pxp_blit(((lv_color_t *)s_inactiveFrameBuffer), &dest_area,
 LCD_WIDTH, color_p, area, LV_OPA_COVER, LV_DISP_ROT_270);
#else /* Use CPU to rotate the panel. */
for (uint32_t y = 0; y < LVGL_BUFFER_HEIGHT; y++)
{
    for (uint32_t x = 0; x < LVGL_BUFFER_WIDTH; x++)
    {
        ((lv_color_t *)s_inactiveFrameBuffer)[(LCD_HEIGHT - x) * LCD_WIDTH + y]
 = color_p[y * LVGL_BUFFER_WIDTH + x];
    }
}
#endif
DEMO_FLUSH_DCACHE();
ELCDIF_SetNextBufferAddr(LCDIF, (uint32_t)s_inactiveFrameBuffer);
#else /* DEMO_USE_ROTATE */
```

For more information, refer to the i.MX RT1170 SDK from GUI Guider template.

6.5 Design multiple page application by titleview

Tileview is implemented as a standard widget in GUI Guider. You can design the GUI in tileview by drag and drop operation.

To use the **tileview** widget, perform the following steps.

1. Drag the tileview widget to the editor.
   Note: *If you are unable to find the widget, type the name of the widget in the search field and press Enter. The widget name appears in the search results.*
2. To add a page in the **Attributes** group on the right, click the **+** icon.

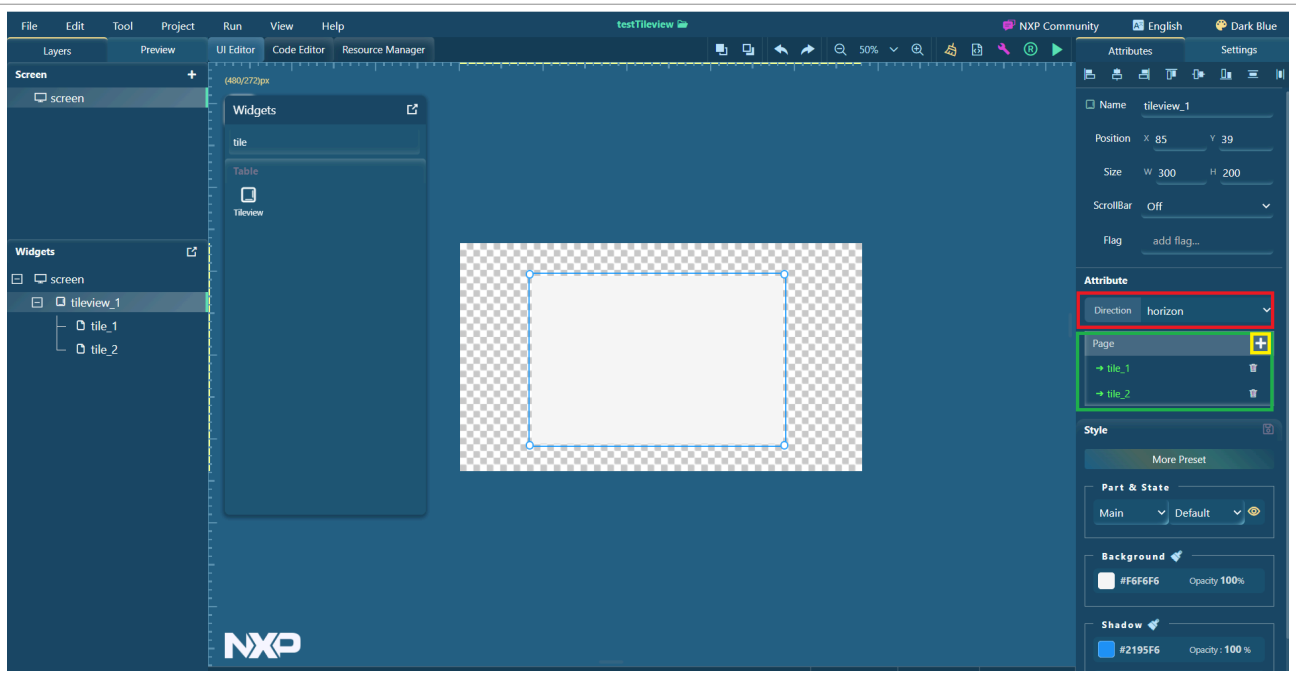

Figure 109. Adding a page

3. Select the **tile_1** tab.
4. Drag a button widget to the **tileview** widget.

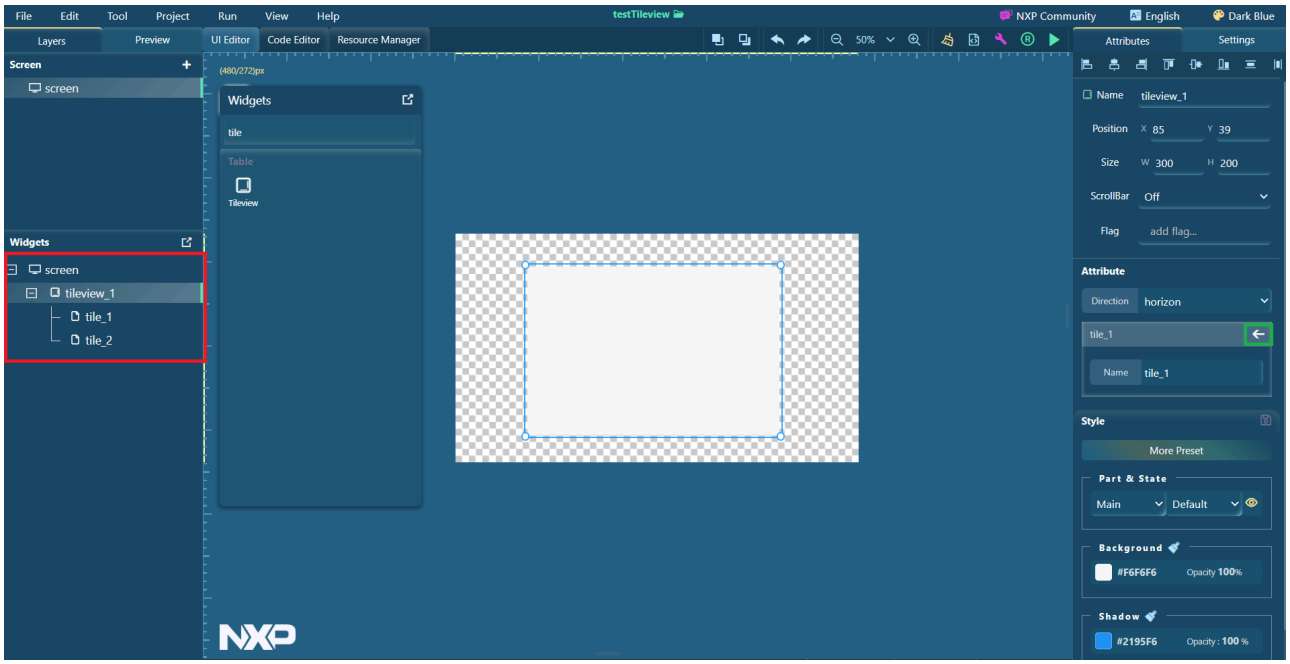

Figure 110. Button widget

5. Select the **tile_2** tab.
6. Drag an **arc** widget to the tileview widget.

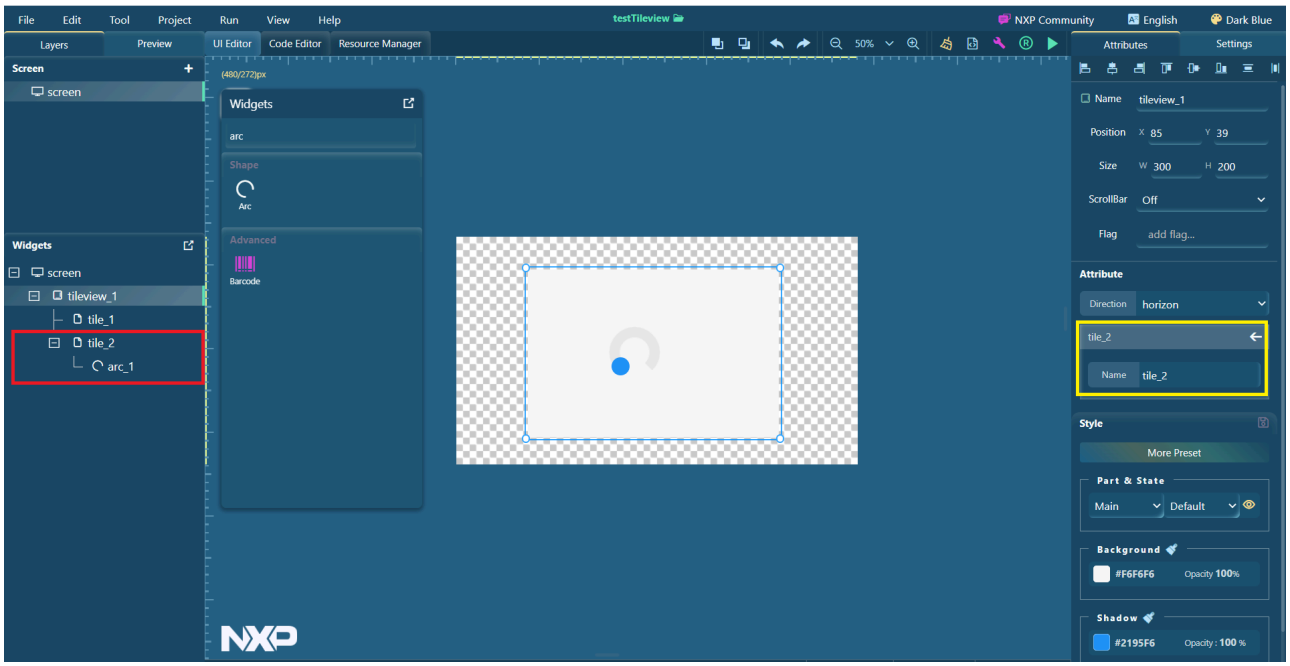

Figure 111. Arc widget

6.6 Customize variables in lv_conf.h

There is a configuration header file for LVGL called `lv_conf.h`. It sets the basic behavior of the library, disables unused modules and features, adjusts the size of memory buffers in compile time, and so on. In GUI

Guider, `lv_conf.h` is based on variations and variations of the widgets used in the project. When you generate a code, the file gets overwritten. But sometimes, it is necessary to modify it.

To customize `lv_conf.h`, follow the steps below:

1. Find the `lv_conf_ext.h` location in the folder named "custom".
2. Remove the variable define, then redefine it.
   Example:

```
#undef LV_FONT_FMT_TXT_LARGE
#define LV_FONT_FMT_TXT_LARGE 1
```

6.7 Experience with MicroPython in GUI Guider

MicroPython is a lean and efficient implementation of the Python 3 programming language. MicroPython includes a small subset of the Python standard library and is optimized to run on microcontrollers and in constrained environments.

By building LVGL as a MicroPython module, a user can have a high-level GUI library for fast prototyping GUI, taking advantage of Python's language features. These features include Inheritance, Closures, List Comprehension, Generators, Exception Handling, Arbitrary Precision Integers, and others.

GUI Guider ships prebuilt MicroPython binaries by default. For more information on how to build, see the `lv_MicroPython` README.

6.7.1 Generate code

The user can select "C" or "MicroPython" after clicking the **Generate code** button on the GUI Guider UI. The code for both C and Python is generated separately. The C source code and MicroPython file `gui_guider.py` is created under the folder *<GUI Guider Project name>/generated*.

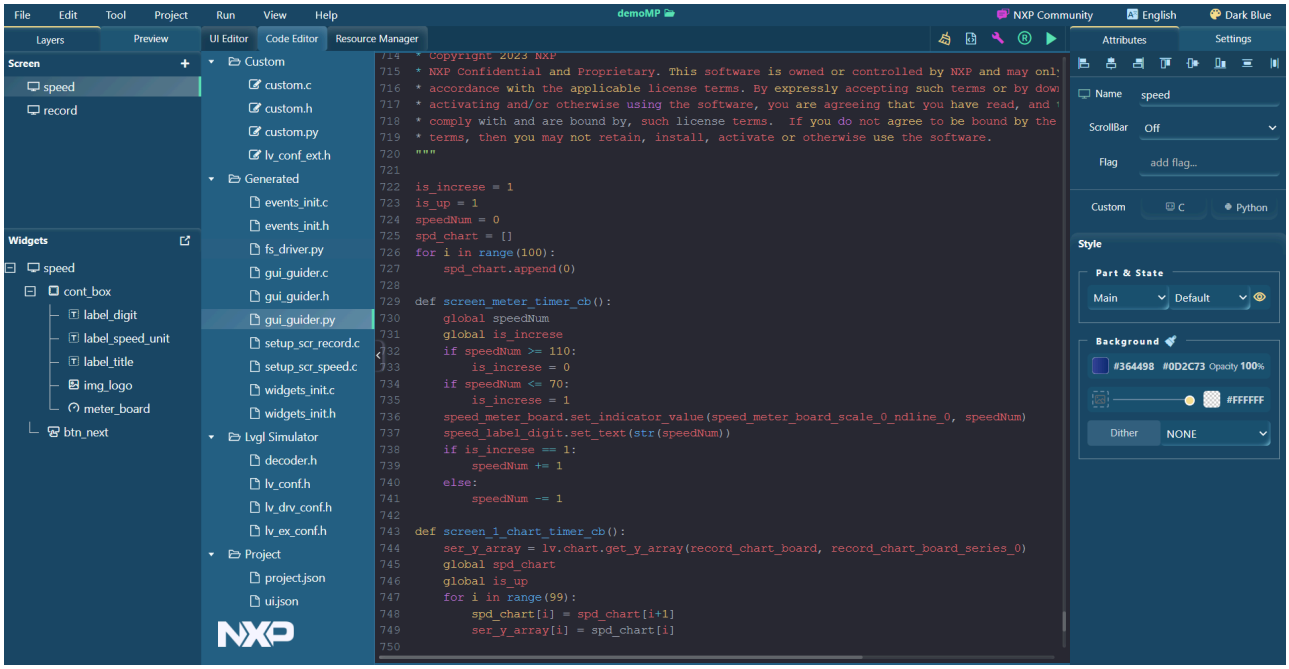

Figure 112. View MicroPython code

6.7.2 Run simulator

Click the **Run simulator** > **MicroPython** button. The GUI Guider generates code and launches the simulator in a separate window.

6.7.3 Add custom code

Like C, GUI Guider supports adding custom Python code, either as event action, or as independent `custom.py` file under the folder "custom".

Note: *Indentation is a very important concept of Python because without proper indenting the code, IndentationError appears and the code is not compiled. To avoid this error, GUI Guider follows the below assumptions during the code generation:*

- *Each line of a block is indented with four spaces.*
- *Tab is replaced with four spaces automatically.*

6.7.3.1 As event action

Table 20 provides a description of the custom Python code options.

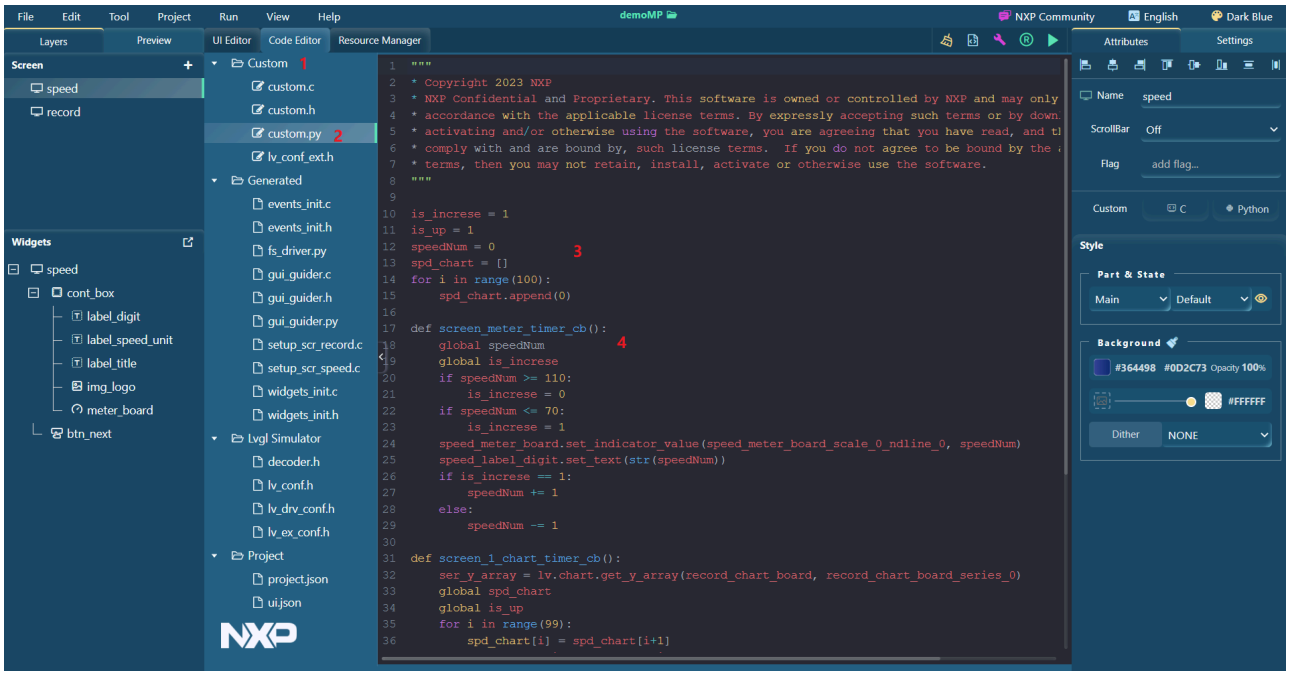

Figure 113. Python code options

Table 20. Custom Python code options

Label	Description
1	Event Type: Customer code
2	Event Code Type: Python code
3	Global variable or function
4	Codes that are wrapped in event callback

6.7.3.2 As custom.py

Put the custom.py file into the folder *<GUI-Guider-Project-name>/custom/*. The content appears merged into the `final gui_guilder.py` file, replacing the tab with four spaces.

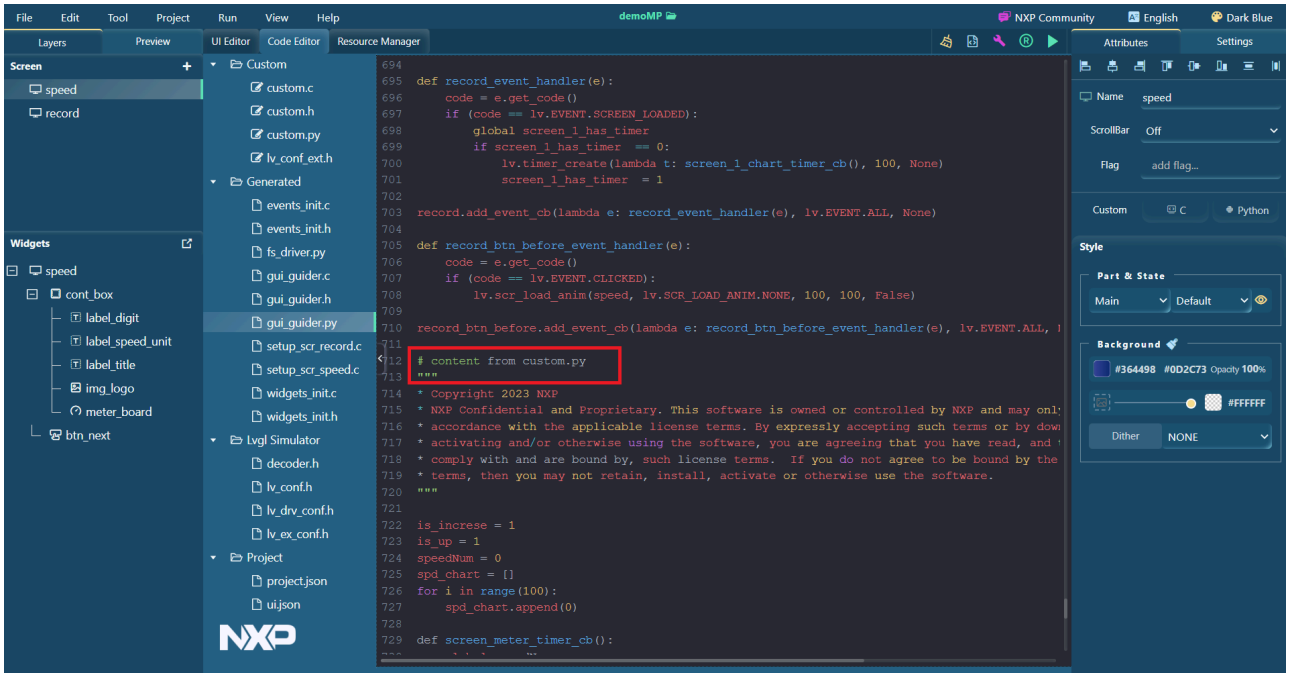

Figure 114. Content appears merged into the final gui_guilder.py file

6.7.4 Limitations

Following are the limitations of MicroPython:

- Only LVGL v8 is supported.
- Compared to C, MicroPython runs slower. Due to this, some animations are not added in music player demo.

6.8 Upgrade a project with a newer GUI Guider

Some projects are designed based on lower versions of the LVGL graphics library and GUI Guider. As LVGL remains updated, new versions are constantly released, adding many new and useful features. To support the updated LVGL graphics library that includes new functions and bug fixes, GUI Guider is being upgraded to provide users with a better experience.

Based on the above reasons, users hope to upgrade existing designs to adapt to the latest version of GUI Guider.

The current upgrade rules for GUI Guider are: if you use a design made by GUI Guider earlier than version 1.5.0, import and upgrade the project with version 1.5.0 or 1.5.1 and execute generate code. Then import and upgrade the project to the next major version until you update to the latest version.

This example takes a design made in GUI Guider 1.5.1 to explain the process of upgrading a lower version design to a higher version. The following are the specific steps:

1. If the project is opened by the latest GUI Guider, you get the following prompt.

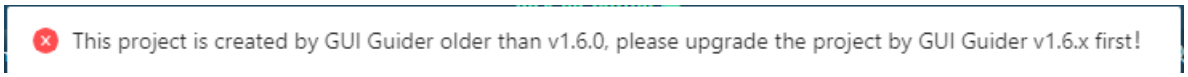

Figure 115. Prompt

2. Open the project in version 1.6.1.

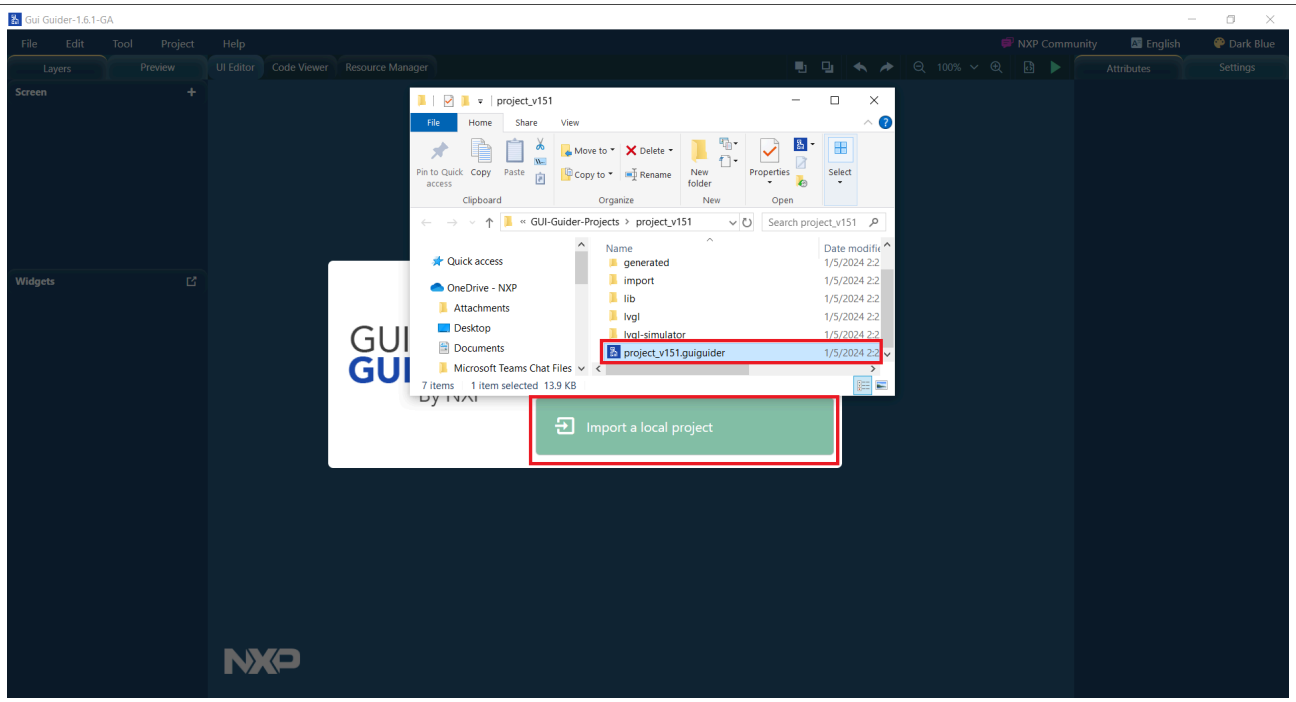

Figure 116. Open project

3. In the project update prompt window, click OK to start the upgrade. if "Backup before update" is selected, GUI Guider backs up the design before version upgrade.

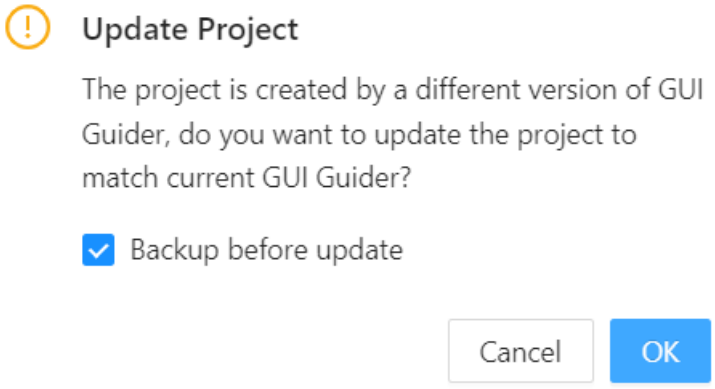

Figure 117. Project update prompt window

4. After the upgrade is completed, the project version can be checked in the project file.

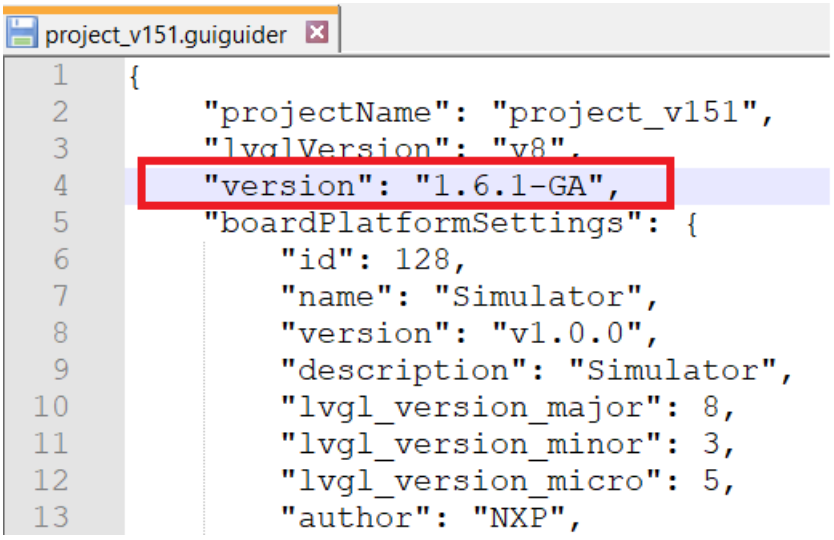

Figure 118. Check the project version

5. Open the GUI Guider 1.7.0. Repeat steps 2 and 3. After repeating the steps, the project is successfully upgraded.

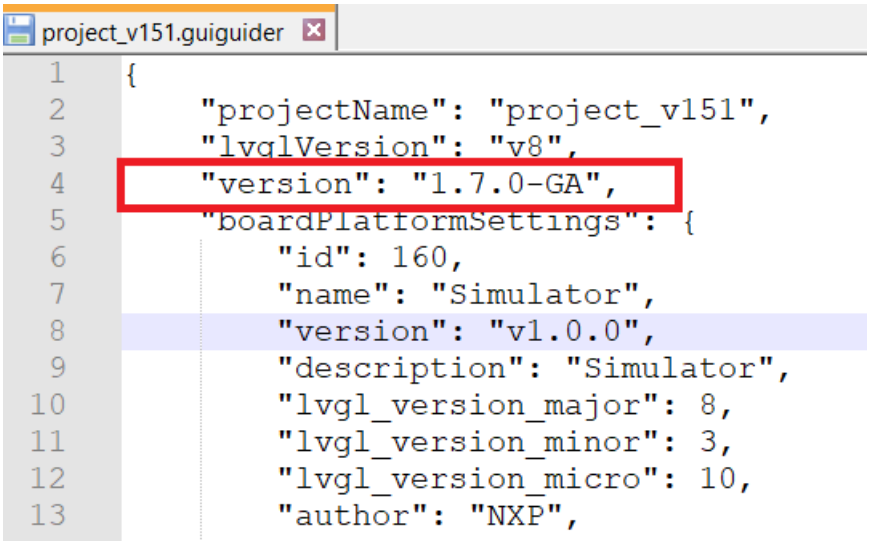

Figure 119. Project upgrade successful

6.9 How to set the gradient color

Gradient color is a commonly used style in UI design. This example describes how to set the gradient color for a widget through a color picker.

1. To open the color picker window, click the color setting icon.

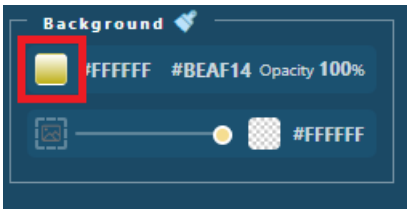

Figure 120. Color setting icon

2. To enable the gradient color, click the highlight icon.

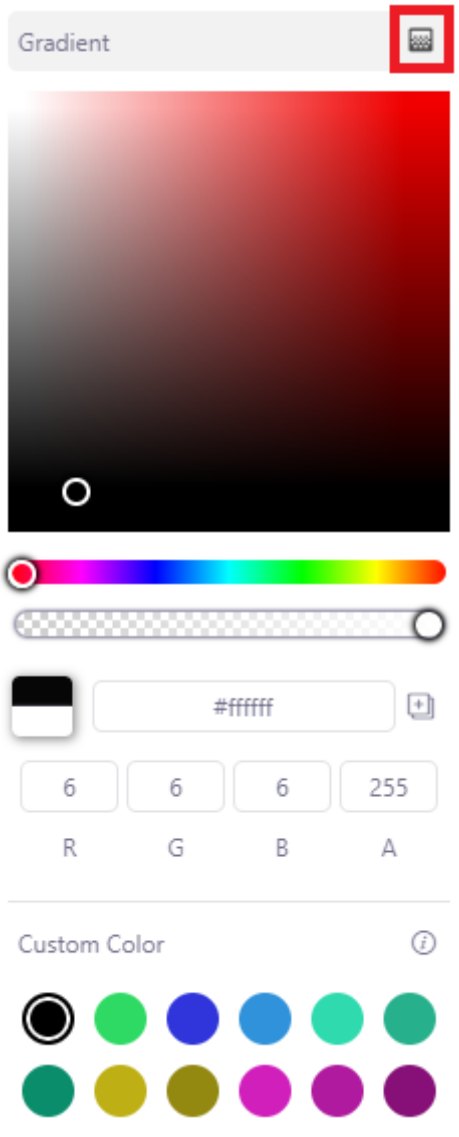

Figure 121. Enable the gradient color

3. To enable the gradient color, click the highlight icon.
   - Set the point from which the background color should start for gradients: The position from which the gradient starts.
   - Set the point from which the background's gradient color should start: The position to which the gradient finishes.
   - Back to the previous color: Resume to original color.

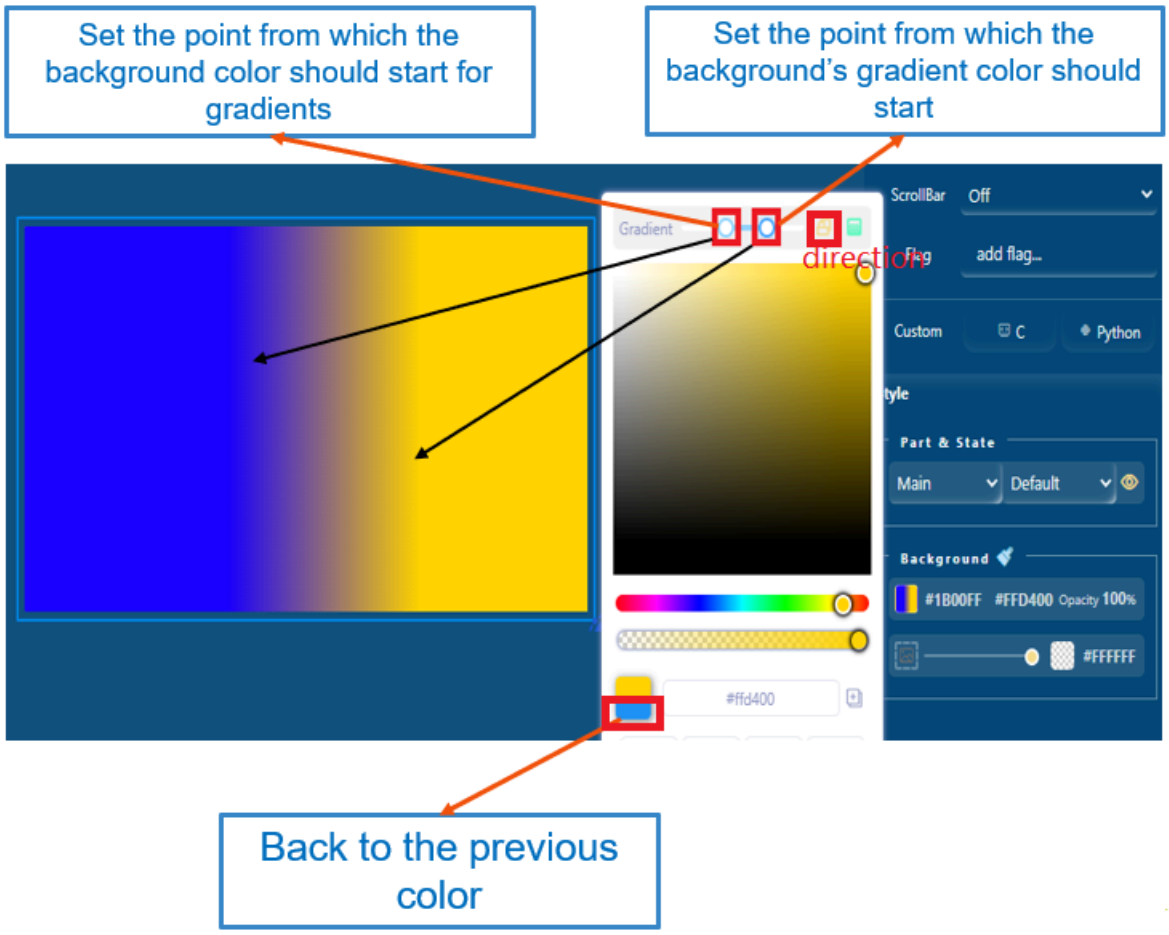

Figure 122. Color picker window

6.10 How to develop a multi-language application

Currently GUI Guider does not support multiple language application design by configuring through the IDE. The custom code must be developed to achieve this function. The following example describes how to develop a multi-language application and switch languages by a dropdown-list during the runtime.

1. Add the following widgets into the GUI editor with the default language characters.

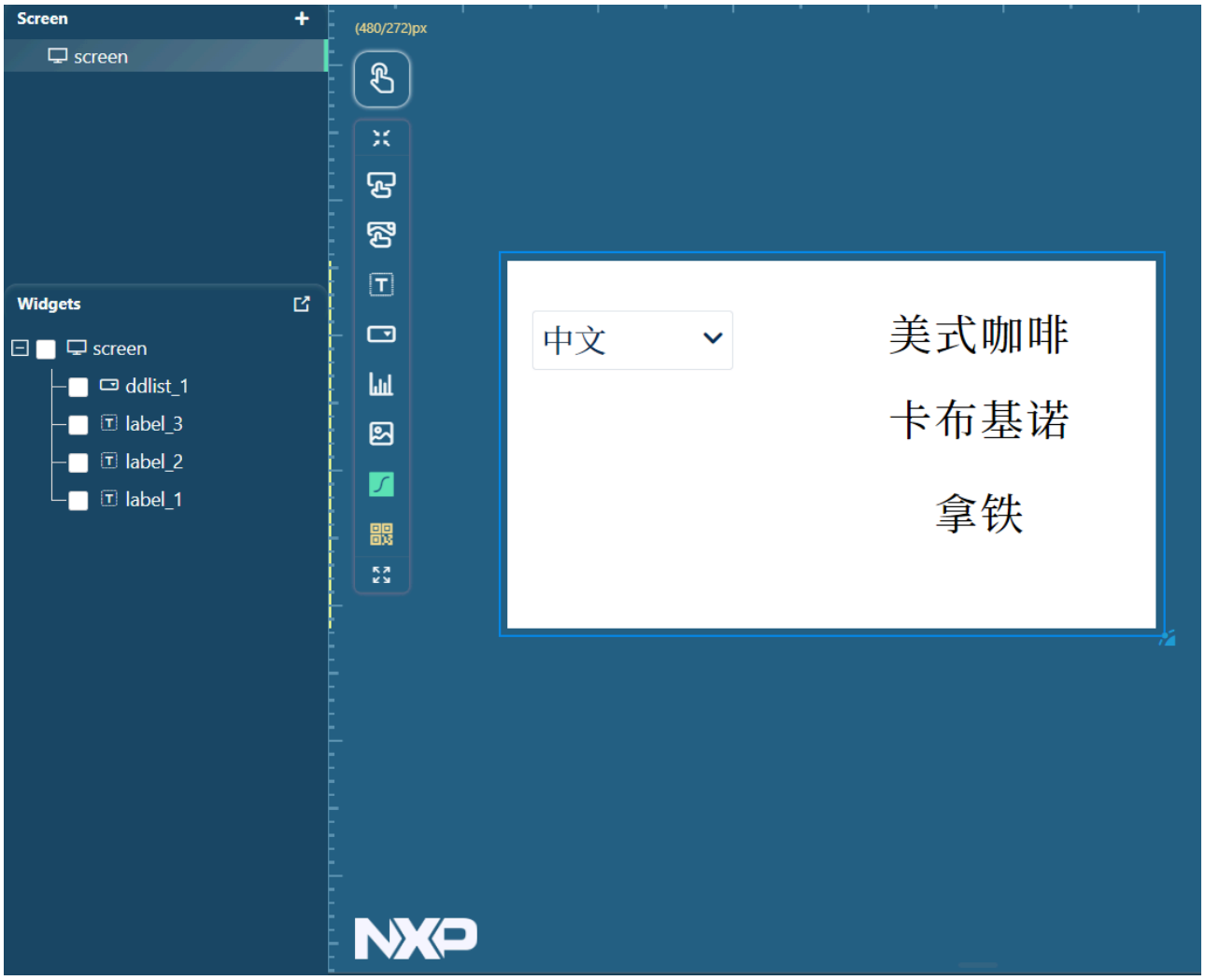

Figure 123. Add the widgets into the GUI editor

2. Write functions that display various languages in custom/custom.c.

```c
/**********************
 *  STATIC VARIABLES
 **********************/

void en_cn_font_change()
{
    lv_style_value_t v = {
        .ptr = &lv_font_SourceHanSerifSC_Regular_32
    };
    lv_obj_set_local_style_prop(guider_ui.screen_label_1, LV_STYLE_TEXT_FONT, v, LV_PART_MAIN|LV_STATE_DEFAULT);
    lv_label_set_text(guider_ui.screen_label_1, "美式咖啡");
    lv_obj_set_local_style_prop(guider_ui.screen_label_2, LV_STYLE_TEXT_FONT, v, LV_PART_MAIN|LV_STATE_DEFAULT);
    lv_label_set_text(guider_ui.screen_label_2, "卡布基诺");
    lv_obj_set_local_style_prop(guider_ui.screen_label_3, LV_STYLE_TEXT_FONT, v, LV_PART_MAIN|LV_STATE_DEFAULT);
    lv_label_set_text(guider_ui.screen_label_3, "拿铁");
}

void cn_en_font_change()
{
    lv_style_value_t v = {
        .ptr = &lv_font_SourceHanSerifSC_Regular_24
    };

    lv_obj_set_local_style_prop(guider_ui.screen_label_1, LV_STYLE_TEXT_FONT, v, LV_PART_MAIN|LV_STATE_DEFAULT);
    lv_label_set_text(guider_ui.screen_label_1, "Americano");
    lv_obj_set_local_style_prop(guider_ui.screen_label_2, LV_STYLE_TEXT_FONT, v, LV_PART_MAIN|LV_STATE_DEFAULT);
    lv_label_set_text(guider_ui.screen_label_2, "Cappuccino");
    lv_obj_set_local_style_prop(guider_ui.screen_label_3, LV_STYLE_TEXT_FONT, v, LV_PART_MAIN|LV_STATE_DEFAULT);
    lv_label_set_text(guider_ui.screen_label_3, "CafeLatte");
}
```

Figure 124. Write functions

3. To call the function in the above steps, add an event to the widget's item.

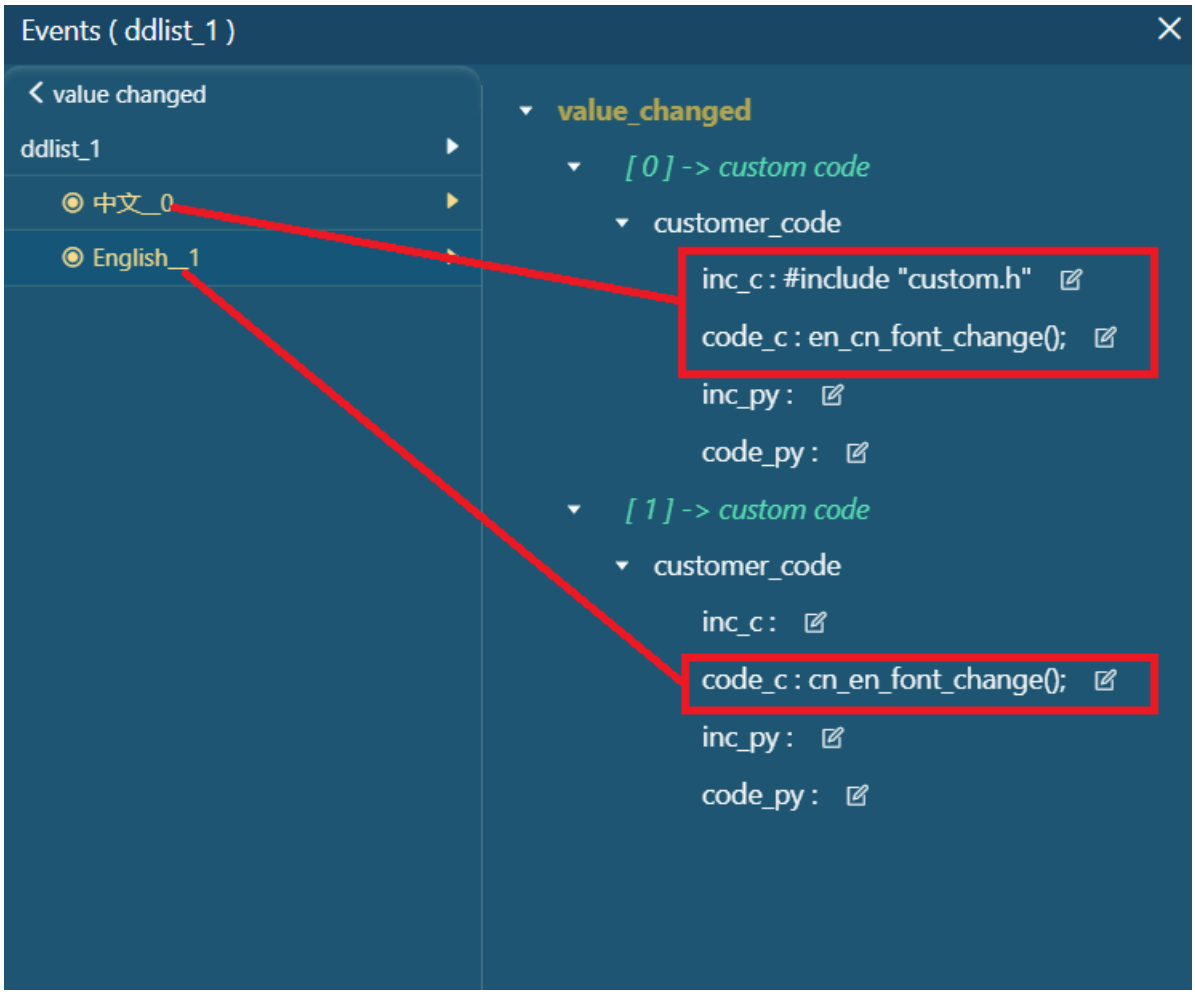

Figure 125. Add an event

4. Run and build the simulator. Then, you can change the label text by switching the dropdown-list item.

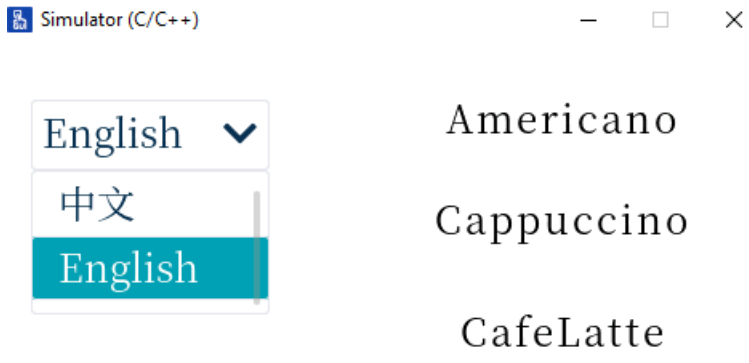

Figure 126. Change the label text

6.11 How to use symbols

GUI Guider supports the following built-in symbols of the **FontAwesome** font in LVGL.

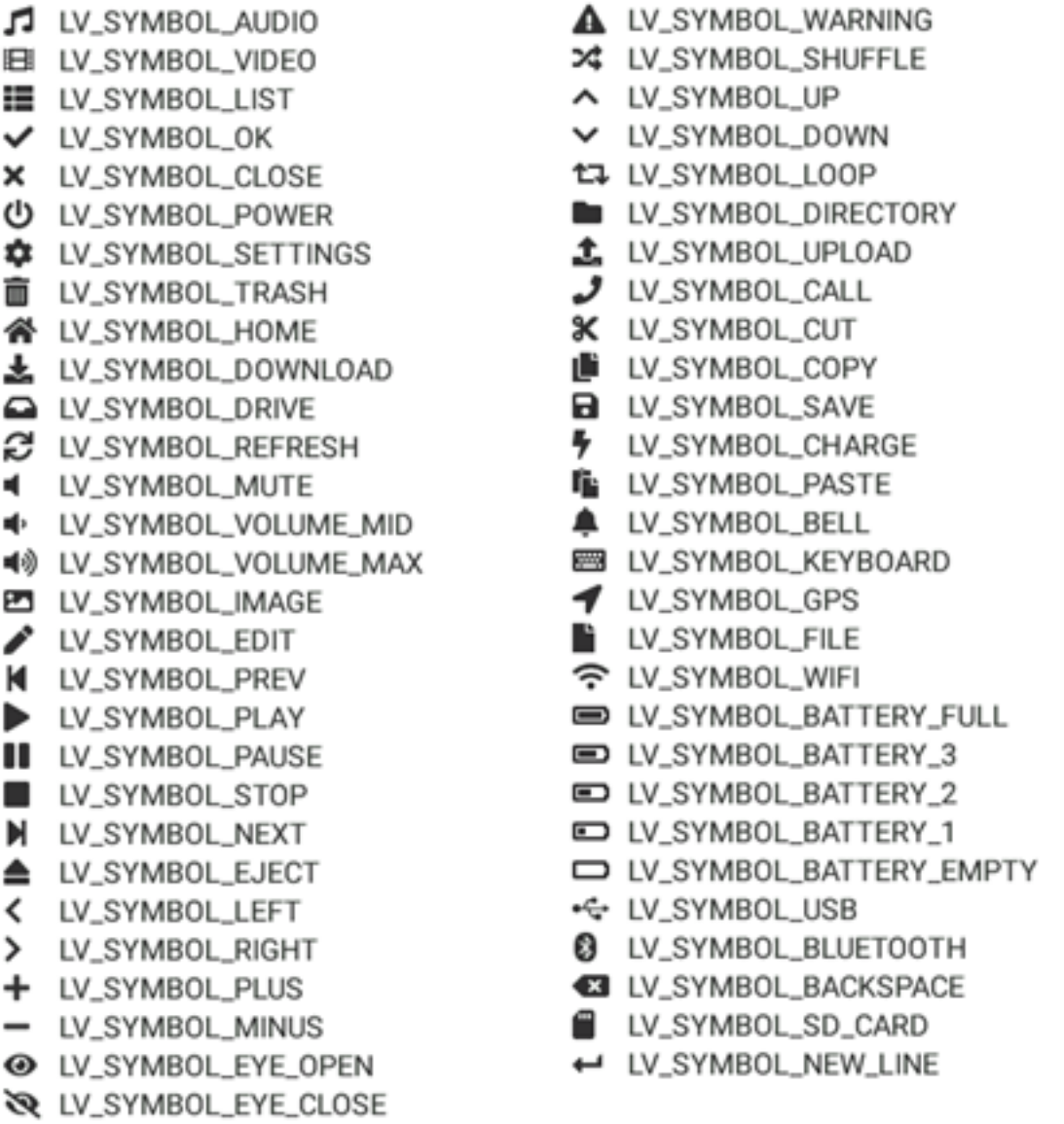

Figure 127. Built-in symbols

There are two ways of using symbols in GUI Guider. The first way is to use the built-in symbols in the attribute setting window. The second way is to write custom code to use custom symbols. The following steps introduce how to use symbols in two different ways.

First, create a GUI Guider project and add a label into the screen.

- Using GUI Guider built-in symbols.
  1. To open the symbol list window and choose the symbols, click the data source icon. Make sure that the font family is **FontAwesome**.

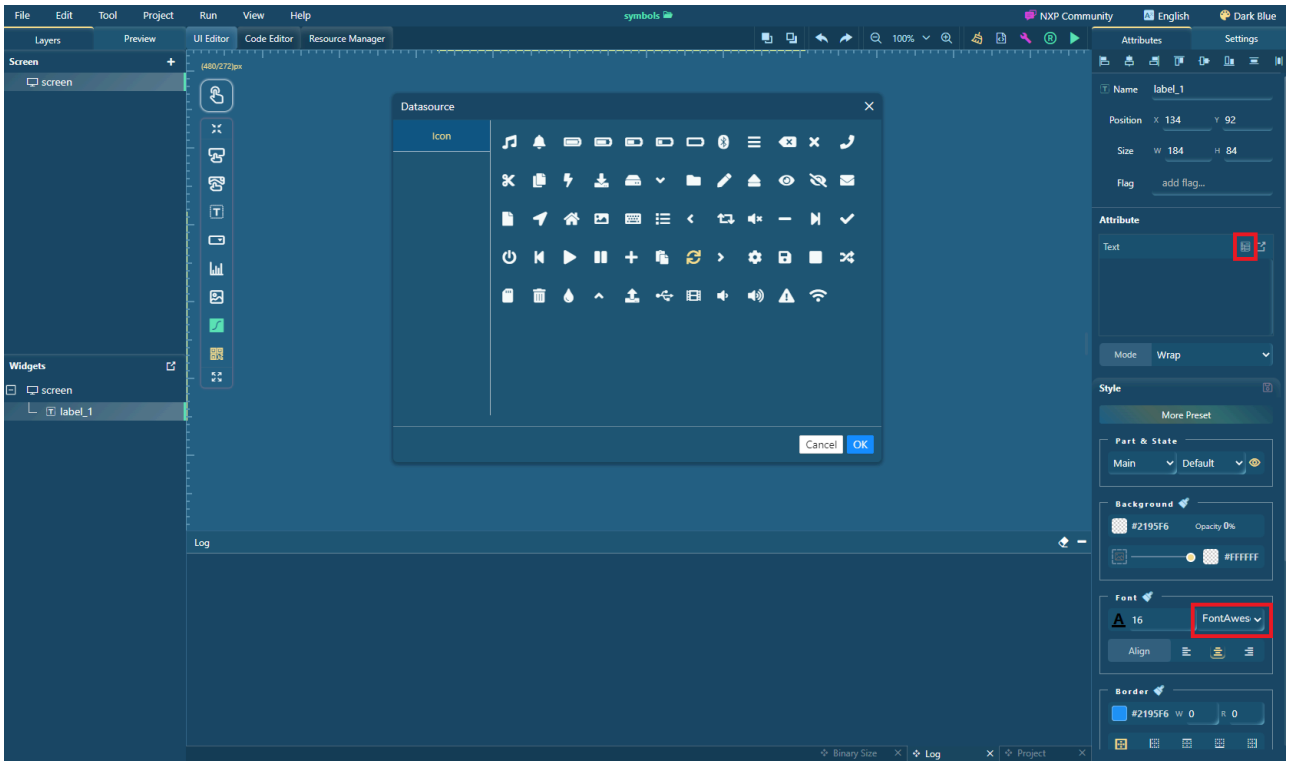

Figure 128. Symbol list window

2. Build and run the project.

- Using custom symbols.
  1. Search the symbols to be used from https://fontawesome.com. For example, choose the truck, and copy the unicode ID, which is **0xf0d1.**

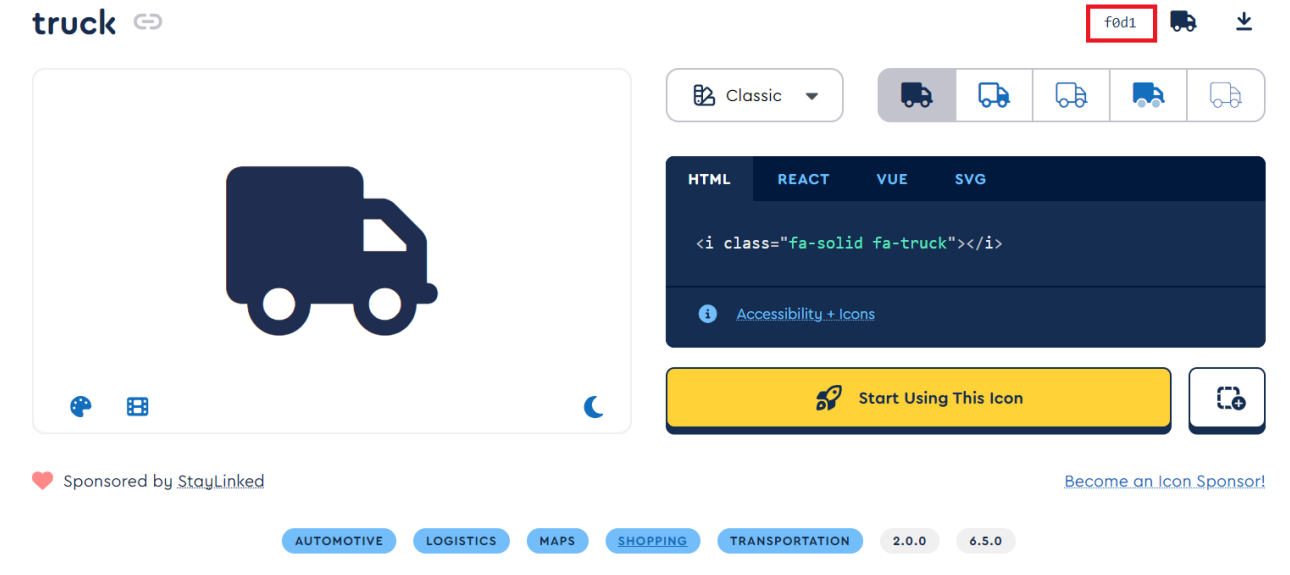

Figure 129. Copy the unicode ID

2. Convert this symbol to the C file on Online font converter, and set parameters, including name, size, BPP, and font file (FontAwesome.woff). Then, add the generated C source code into the project. In this example, put the `demo_truck.c` in the GUI Guider project custom folder.

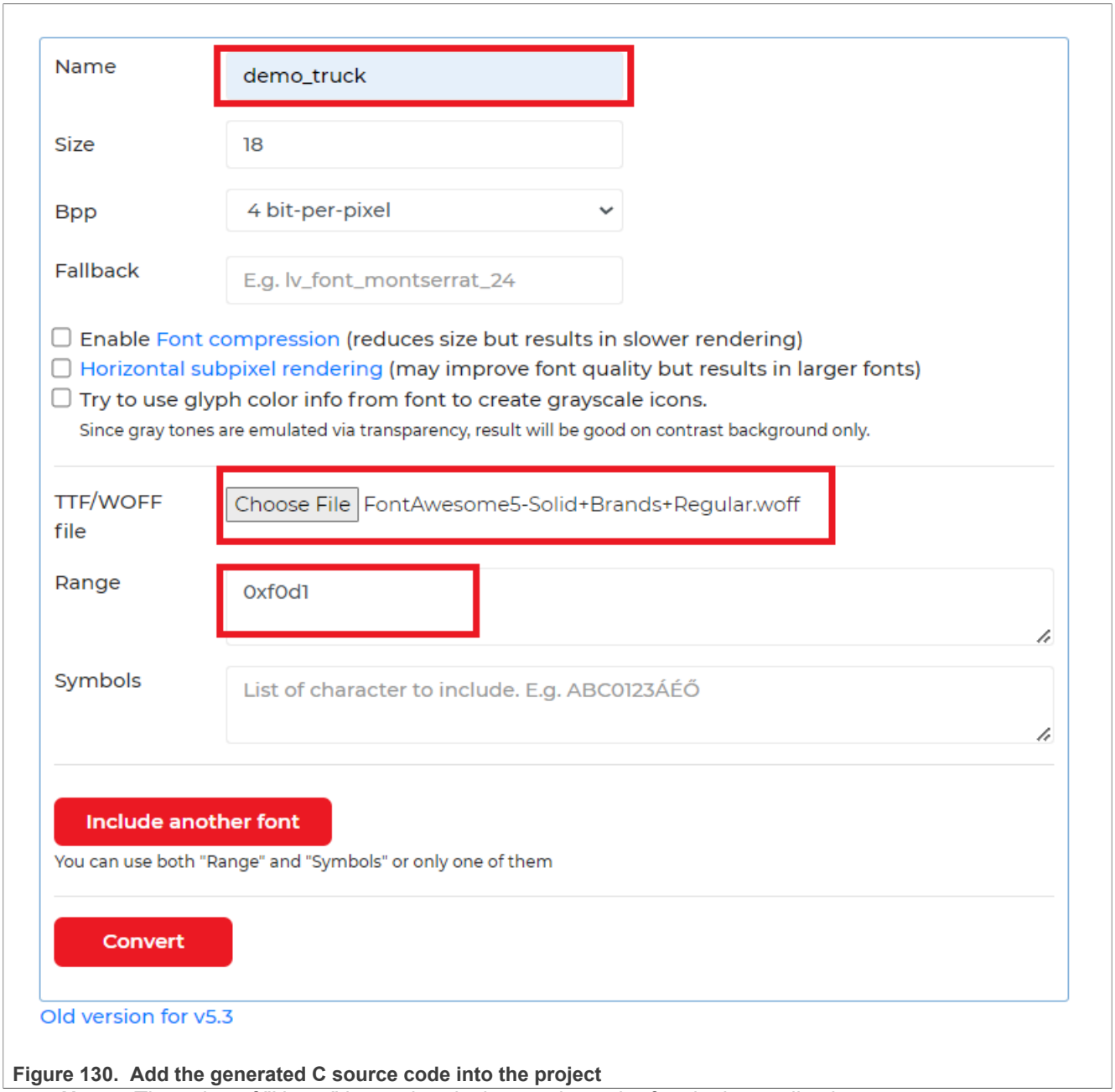

Figure 130. Add the generated C source code into the project

Name: The value of "Name" is used to declare and use the font in the application.
TTF/WOFF file: Choose a font file that can support the symbol character. In this example, it is "FontAwesome.woff".
Range: Add the Symbol unicode ID. More symbols can be enumerated with ",".

3. Covert the unicode value to UTF-8. For **0xf0d1**, the Hex UTF-8 bytes are **EF 83 91**.
4. Define the symbol string to UTF-8 values in `custom.h`.

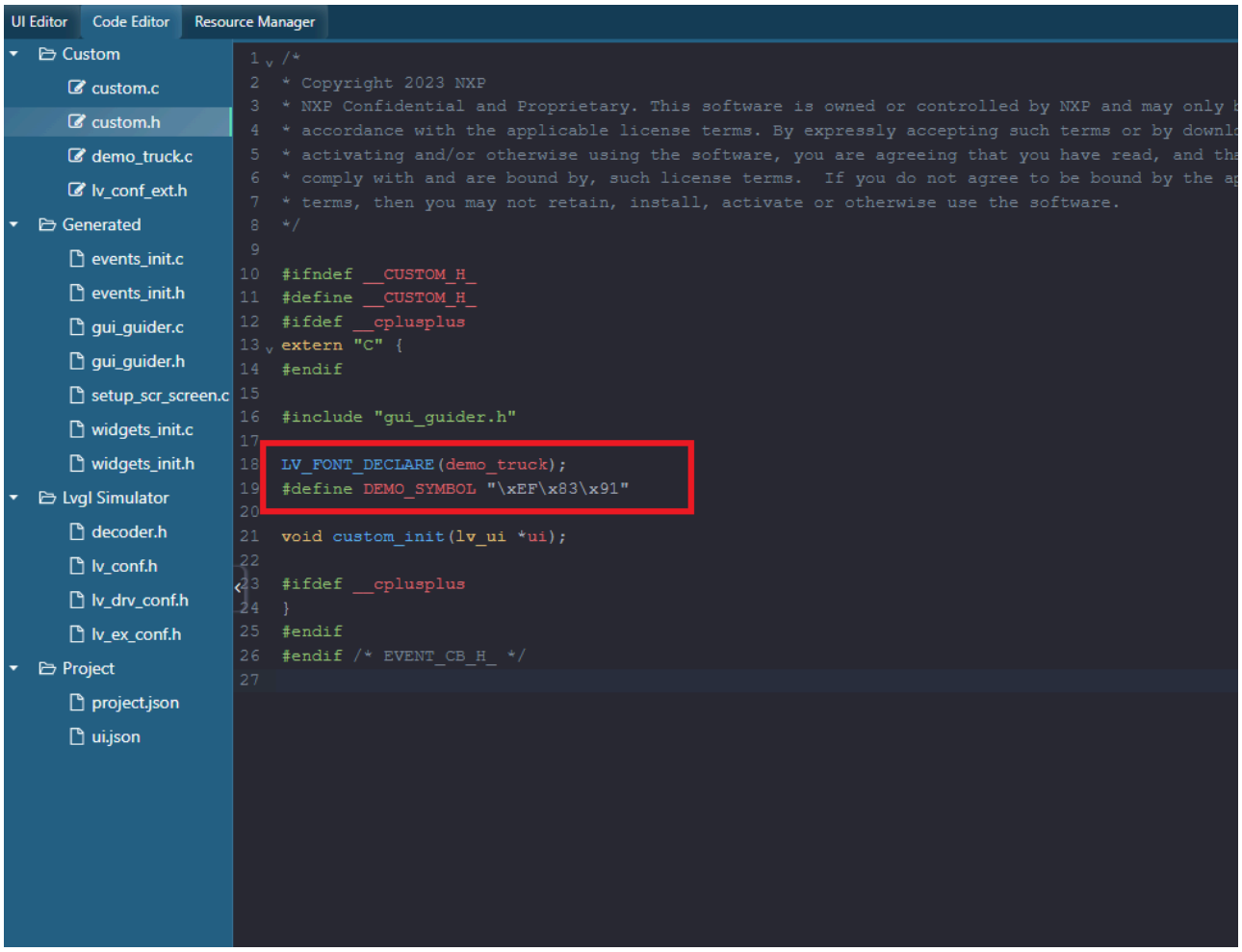

Figure 131. Define the symbol

5. Use the symbol in GUI Guider project by adding the following code to redefine the label font text and font style in the screen custom code window.

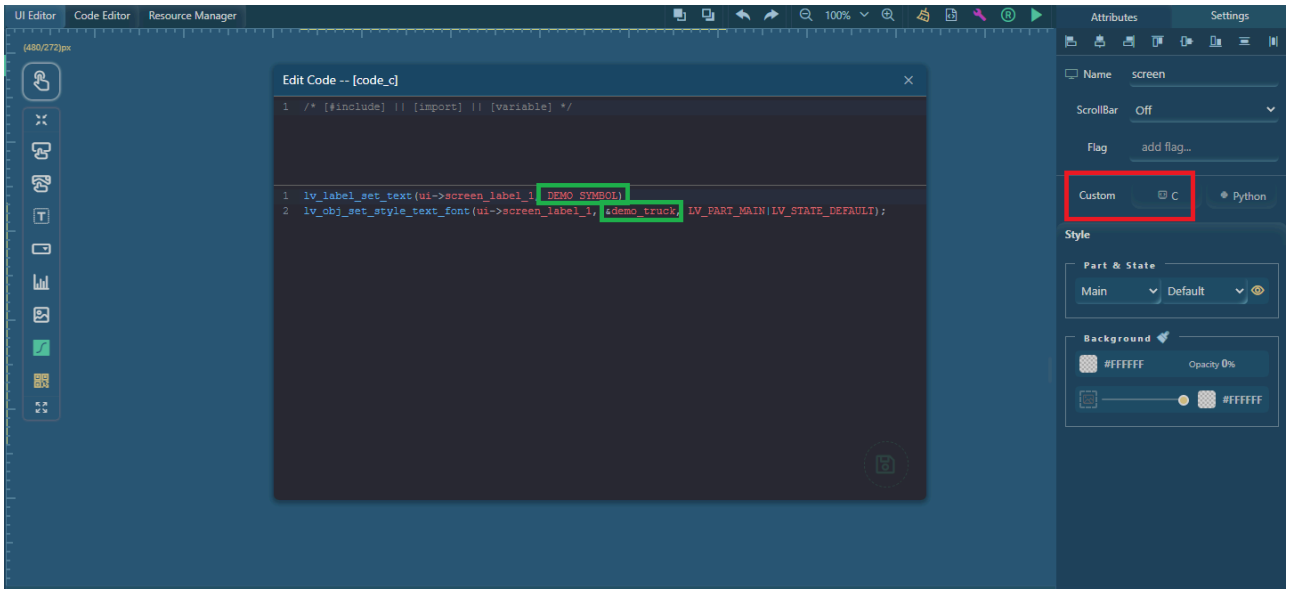

Figure 132. Screen custom code window

6. Build and run the project. The custom symbol is displayed in the label widget.

Figure 133. Build and run the project

7 Miscellaneous

This chapter describes various miscellaneous information for GUI Guider users.

7.1 Frequently Asked Questions (FAQs)

This chapter lists the Frequently Asked Questions (FAQs) about GUI Guider.

Question: How to avoid simulator running the MCU-specific code?

Answer: GUI Guider provides a predefined macro `LV_USE_GUIDER_SIMULATOR` in `lv_conf.h`. Do the following changes in your source files:

```
#if !LV_USE_GUIDER_SIMULATOR // or LV_USE_GUIDER_SIMULATOR == 0
...(MCU specific Code)
```

```
#endif
```

Question: What should I do if the project upgrade fails?

Answer: The backup project can be found in the workspace folder if you enable the backup function during upgrade. Use previous version for development and report the issue in https://community.nxp.com/t5/GUI-Guider/bd-p/GUI-Guider.

Question: What to do if the effect is inconsistent after upgrading the project?

Answer: It is necessary to manually adjust the style to compare with the old project, for example, font size, font position, and so on.

Question: How to reuse applications on other boards?

Answer: When creating a project, select an existing project as a template, according to the auto-size function configuration board display.

Question: How to make the Meter control detect the rounded rectangle needle?

Answer: Use the image needle.

Question: How to resolve the issue if touch or display does not work?

Answer: The issue is possibly caused if an incorrect panel type is selected. Check the panel type of your project.

Question: How to customize and modify `lv_conf` macro without being automatically overwritten by GG?

Answer: Customize `lv_conf.h` options using `lv_conf_ext.h` in custom folder.

Question: How to add custom code for different widgets in event setting window?

Answer: When the custom code is updated for a widget, close the custom code editor and click another widget. Then, open the custom code editor in event setting window and input the custom code for selected widget.

Question: What should I do if the build fails when running the LVGL v7 project with "Run > MCUXpresso"?

Answer: Check the MCUXpresso IDE version. Make sure that the version is lower than 11.8.0.

8 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT

SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

9 Revision history

Table 21 summarizes the revisions to this document.

Table 21. Revision history

Document ID	Release date	Description
GUIGUIDERUG_1.7.1 v.15.0	29 March 2024	Updated and added multiple sections for v1.7.1
GUIGUIDERUG_1.7.0 v.14.0	31 January 2024	Updated and added multiple sections for v1.7.0
GUIGUIDERUG v.13.0	29 September 2023	Updated and added multiple sections for v1.6.1
GUIGUIDERUG v.12.0	31 July 2023	Updated multiple sections for v1.6.0
GUIGUIDERUG v.11.0	31 March 2023	Updated multiple sections for v1.5.1
GUIGUIDERUG v.10.0	10 January 2023	Added and updated multiple sections for v1.5.0
GUIGUIDERUG v.9.0	23 September 2022	Updated multiple sections for v1.4.1
GUIGUIDERUG v.8.0	25 July 2022	Updated multiple sections for v1.4.0
GUIGUIDERUG v.7.0	31 March 2022	Updated multiple sections for v1.3.1
GUIGUIDERUG v.6.0	07 January 2022	Updated multiple sections for v1.3.0
GUIGUIDERUG v.5.0	29 September 2021	Updated multiple sections for v1.2.1
GUIGUIDERUG v.4.0	30 July 2021	Added and updated multiple sections for v1.2
GUIGUIDERUG v.3.0	10 May 2021	Added and updated multiple sections for v1.1
GUIGUIDERUG v.2.0	11 January 2021	Added and updated multiple sections
GUIGUIDERUG v.1.0	17 November 2020	Updated Work with MCUXpresso IDE

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

For more information, please visit: https://www.nxp.com

Date of release: 29 March 2024
Document identifier: GUIGUIDERUG_1.7.1