AN13694

Smart Home Demo on GUI Guider for LPC546xx

Rev. 0 — 20 July 2022 Application note

Document information

Information	Content
Keywords	AN13694, GUI Guider, LVGL, Smart Home, LPC546xx
Abstract	This application note takes smart home as an example to introduce the GUI Guider-based development flow, including GUI design and simulation, code generation and debugging, downloading to and running on the evaluation board.

1 Introduction

The smart home has had a rapid and significant development in recent years. To provide services such as lighting control, telephone remote control, anti-theft alarm, and environmental monitoring, the smart home interconnects various household electrical appliances. The smart home also integrates network communication, equipment automation, and information appliances. Therefore, the smart home can allocate resources and play an important role in reducing energy consumption and environmental protection.

Embedded systems with graphical interface, low-power consumption, and high performance play a pivotal role in the smart home field. NXP provides a complete GUI solution called GUI Guider, which utilizes the LVGL graphics library.

This application note takes smart home as an example to introduce the GUI Guider-based development flow, including:

- GUI design and simulation
- Code generation and debugging
- Downloading to and running on the evaluation board

1.1 LVGL

LVGL is an open-source graphics library. It provides everything that you require to create an embedded GUI with easy-to-use graphical elements, beautiful visual effects, and a low memory footprint.

Key features:

- Open source and free to use under MIT license
- Written in C (C++ compatible) and hosted on GitHub
- More than 30 powerful, fully customizable widgets, such as button, image button, checkbox, switch, slider, label, arc, bar, line, canvas, image, roller, slider, meter, table, text area, animation, calendar, chart, list, menu, message box, tabview, and so on
- Display of any resolution, GPU support, multi-display support
- Supports multiple types of input devices:
  - Pointer-like input devices such as touchpad or mouse
  - Keypads such as a normal keyboard or simple numeric keypad
  - Encoders with left/right turn and push options
  - External hardware buttons which are assigned to specific points on the screen
- Drawing features:
  - Anti-aliasing
  - Shadow
  - Line, arc, polygon
  - Mask
- Text features:
  - UTF-8 support
  - Kerning
  - Word wrap and auto texts scrolling
  - Arabic and Persian support
  - Font compression
  - Subpixel rendering

  - Online and offline font converter
  - Interface for custom font engine
  - FreeType integration example
  - Multi-language support
- Image features:
  - Various color formats: RGB, ARGB, Chroma keyed, indexed, alpha only
  - Real-time recoloring of images
  - Real-time zoom and rotation
  - Images can be stored in flash or files (for example, SD card)
  - Online and offline image converter
  - Image decoder interface for caching
  - PNG integration example
- Styles:
  - Cascade styles (like in CSS)
  - Reuse the styles in multiple widgets
  - Local styles for simple changes
  - Themes to give a default appearance
  - Transitions (animations) on state change
- Micropython support
- Rich demo examples and documents

For more details, please refer to LVGL.

1.2 GUI Guider

GUI Guider, a user-friendly graphical user interface development tool from NXP, enables the rapid development of high quality displays with the open-source LVGL graphics library. The drag-and-drop editor of GUI Guider makes it easy to utilize the many features of LVGL. These features include widgets, animations, and styles to create a GUI with minimal or no coding at all.

With the click of a button, you can run your application in a simulated environment or export it to a target project. Generated code from GUI Guider can easily be added to an MCUXpresso IDE or IAR Embedded Workbench project, accelerating the development process and allowing you to add an embedded user interface to your application seamlessly.

GUI Guider is free to use general purpose and crossover MCUs of NXP and includes built-in project templates for several supported platforms.

For more details, please refer to GUI Guider.

2 GUI Development Flow

In general, the GUI development flow based on GUI Guider is shown in Figure 1.

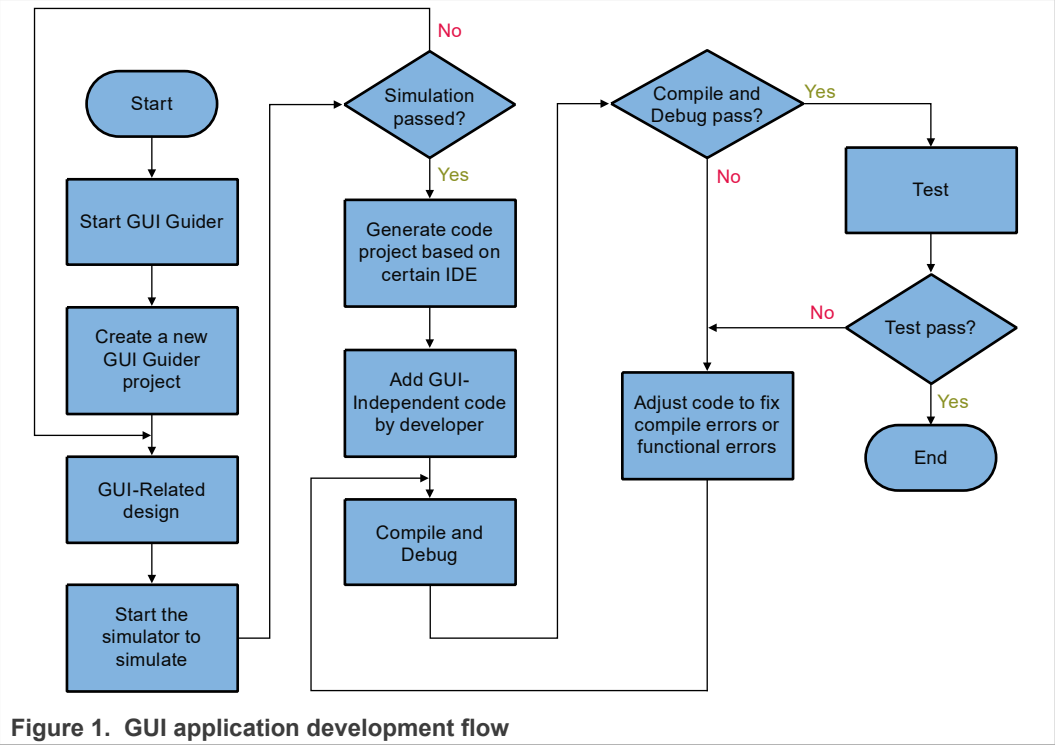

Figure 1. GUI application development flow

As shown in Figure 1, after completing the design, it is recommended to perform a simulation to verify whether the simulation results are as expected or not. If the simulation is successful and as expected, the code can be generated. Otherwise, continue to adjust the design.

One suggestion is to divide the design into GUI-related and GUI-independent parts. GUI-related parts include various widgets, such as buttons, images, sliders, animations, etc., as well as human-machine interaction, such as button click events. The GUI-independent part refers to the part related to a specific hardware system, such as hardware timers, onboard sensors, and so on. This ensures that the design can be verified to the greatest extent possible.

3 Smart home demo introduction

The smart home demo described in this application note is a GUI application with 8 screens. These screens include five main screens and three child-screens attached to the main screens. The relationship between the eight screens is shown in Table 1. The appearances of these screens are shown in Figure 2, Figure 3, and Figure 4.

Table 1. Relationship between main screens and sub screens

Screen ID	Main screen	First level of child screen	Second level of child screen
1	Home		
2	Temperature		
3	Light		
4	Security	Security pin enter	Security pin confirm

Table 1. Relationship between main screens and sub screens...*continued*

Screen ID	Main screen	First level of child screen	Second level of child screen
5	Audio player	Audio list	

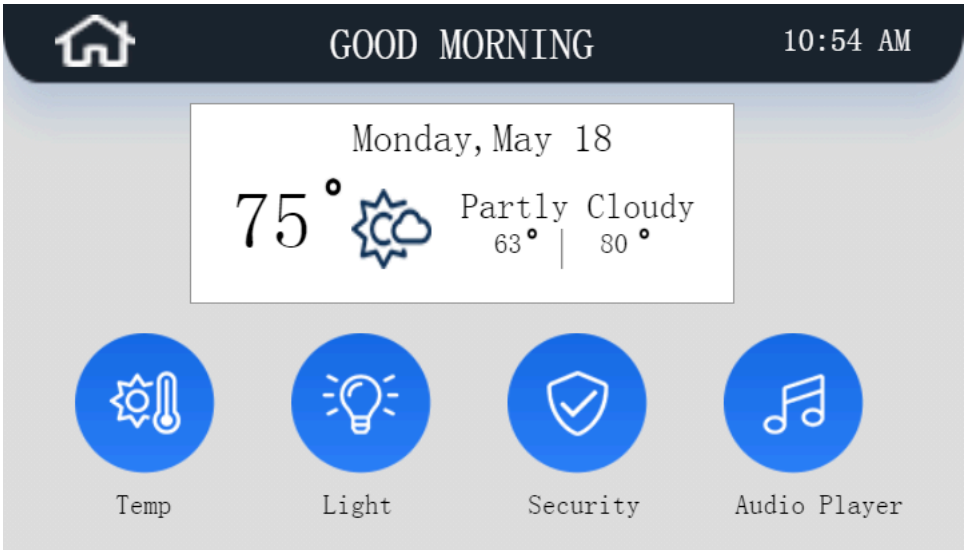

Figure 2. Home screen

Figure 3. Main screens for temperature, light, security, and audio

Figure 4. Child screens for security and audio

Note: *The date, time, and weather presented here are not functional since they are dependent on the hardware of the device. The date and time can be controlled by RTC in an embedded system. It is the same for other screens in the GUI demo.*

4 Development environment

This chapter introduces the software and hardware development environment for the smart home demo.

4.1 Hardware Environment

Hardware development environment for smart home demo consists of two parts: PC with GUI Guider tool installed and an evaluation board equipped with LPC54628 - LPCXpresso54628 (OM13098).

The LPCXpresso54628 board is a binary backward compatible with the LPCXpresso54608 and LPCXpresso54618 boards. For this reason, this demo is easily ported to LPC54608-based or LPC54618-based hardware platforms with only few modifications.

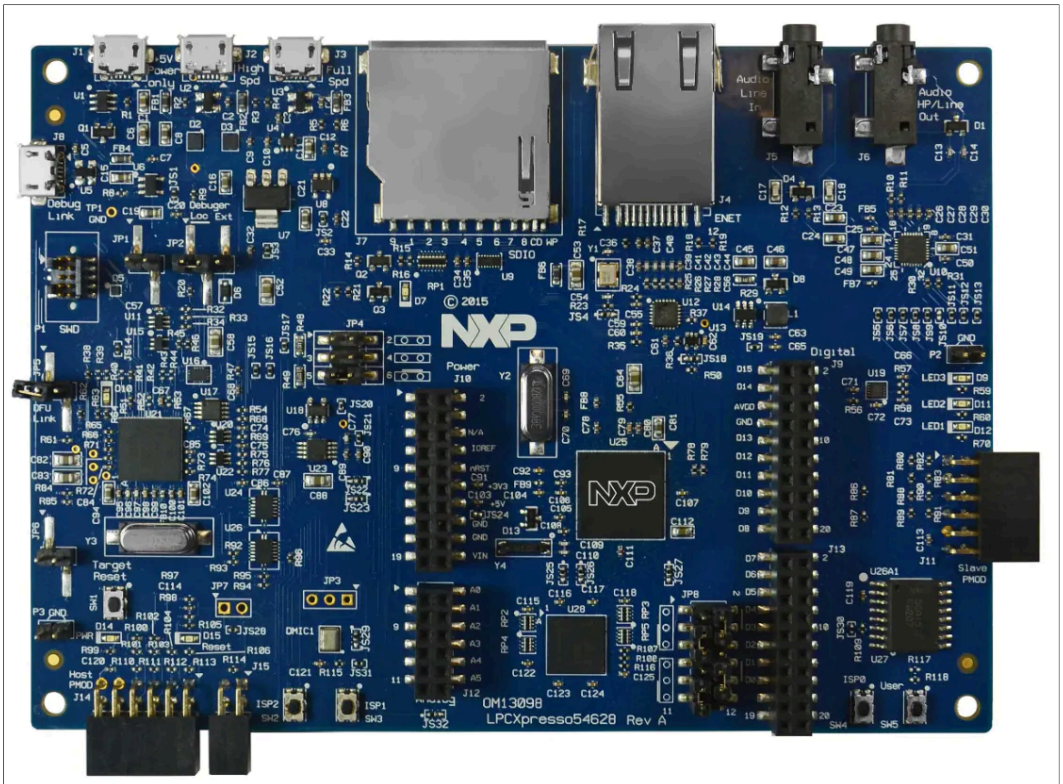

Figure 5. LPCXpresso54628 evaluation board

4.2 Software environment

Smart home demo requires GUI Guider with version 1.3.1 to set up the software environment. GUI Guider with version 1.3.1 supports LVGL with versions 7.10.1 and 8.0.2. The demo described in this application note is based on LVGL with version 8.0.2.

5 Smart home demo implementation

This chapter introduces the key points in the smart home demo implementation process.

5.1 Create a new GUI Guider project

To create a GUI Guider project, follow the steps below:

1. To launch the GUI Guider, double-click the **GUI Guider<version>** icon.

Figure 6. Launch GUI Guider

2. To start the process of project creation, click the **Create a new project** button.

Figure 7. Create a project

3. Select LVGL version as **v8** and click the **Next** button.

Figure 8. Select LVGL version

4. Select **LPC54628** as target board template and then click the **Next** button.

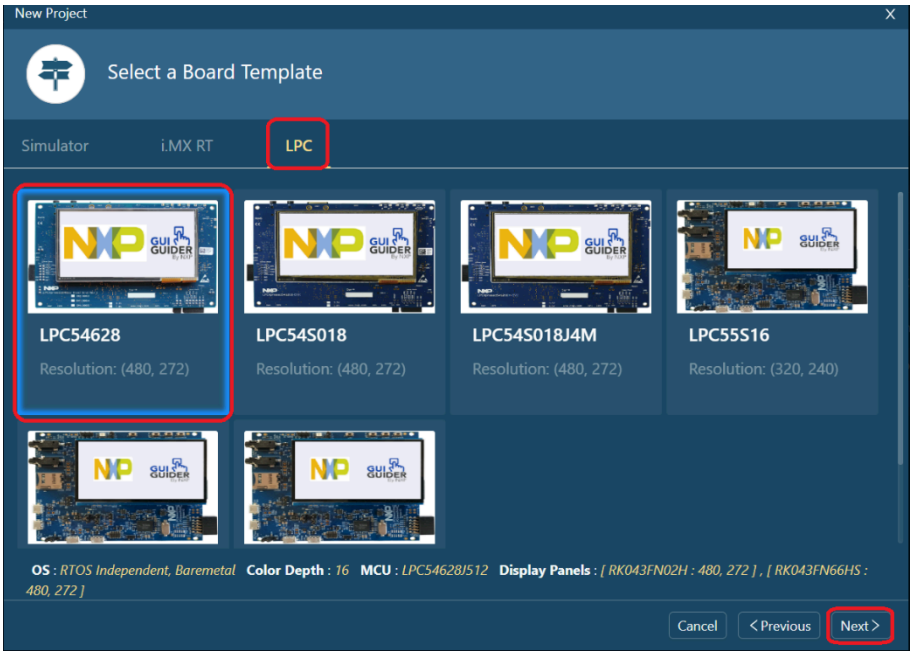

Figure 9. Select a board template

5. Select an empty application template, **EmptyUI**.

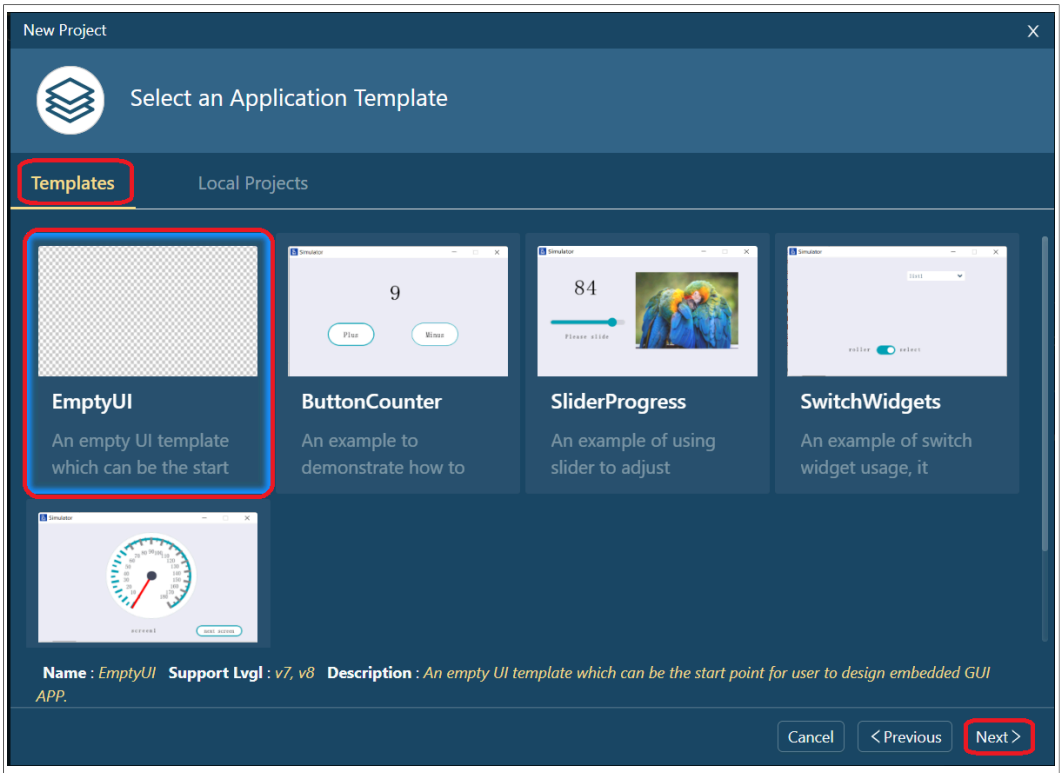

Figure 10. Select an application template

6. Perform project settings, such as project name, project location, panel type, and color depth.

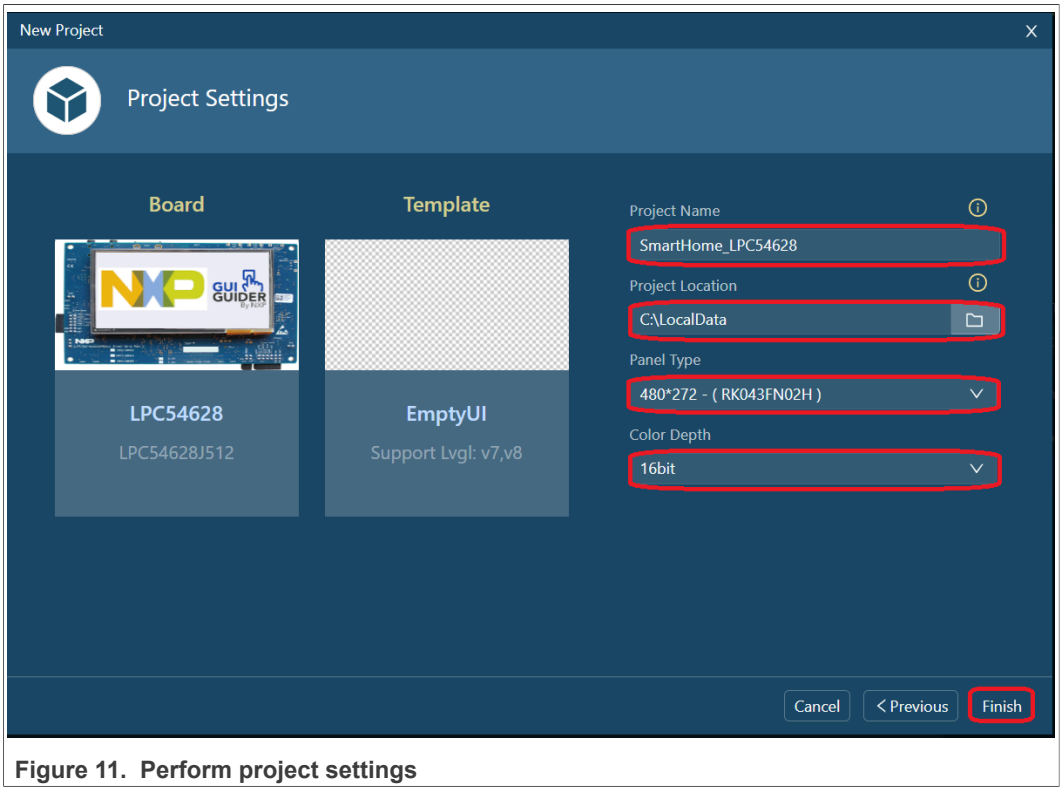

Figure 11. Perform project settings

7. The created project is shown in Figure 12.

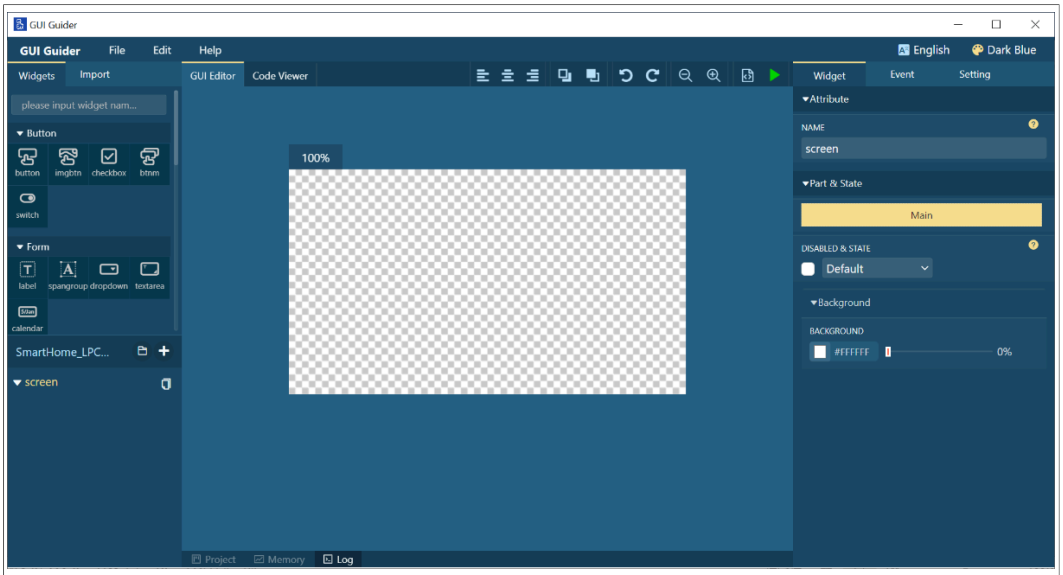

Figure 12. Created project

5.2 Add GUI widgets and set widgets property

GUI Guider supports adding widgets such as button, image, image button to the workspace by dragging and dropping. To illustrate how to add widgets to the workspace, consider adding an image button as an example.

To add widgets to the workspace, follow the steps below:

1. Type **imgbtn** in the search box of the **Widgets** tab, and then drag the image button to the workspace, as shown in Figure 13.

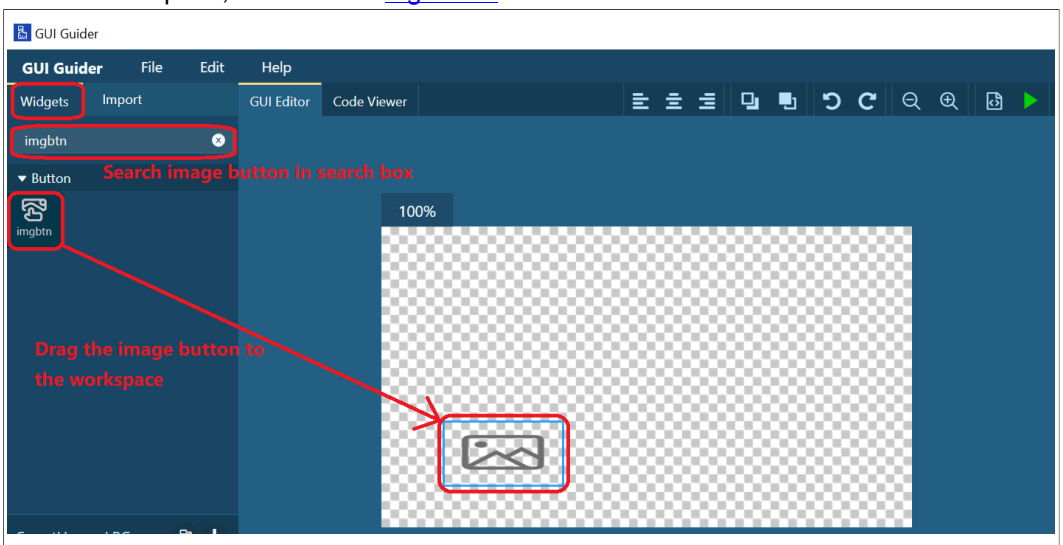

Figure 13. Add image button to workspace

2. Once the image button is placed in the workspace, set the properties of the image button, as shown in Figure 14.

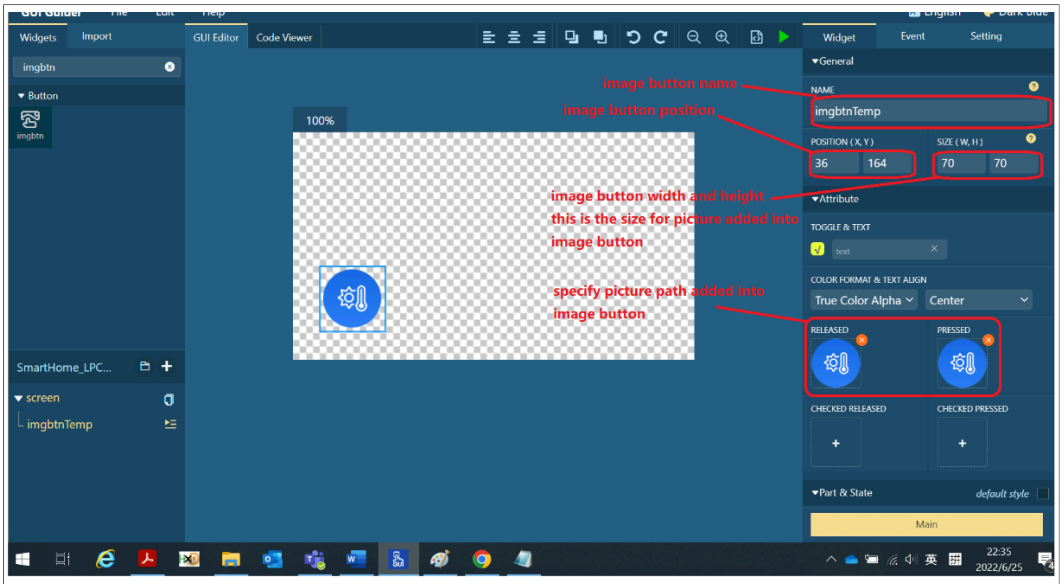

Figure 14. Set image button properties

Note: *For widgets that require image path, before specifying the image for the widget, add the required image files to the import folder, as shown in Figure 15 and Figure 16.*

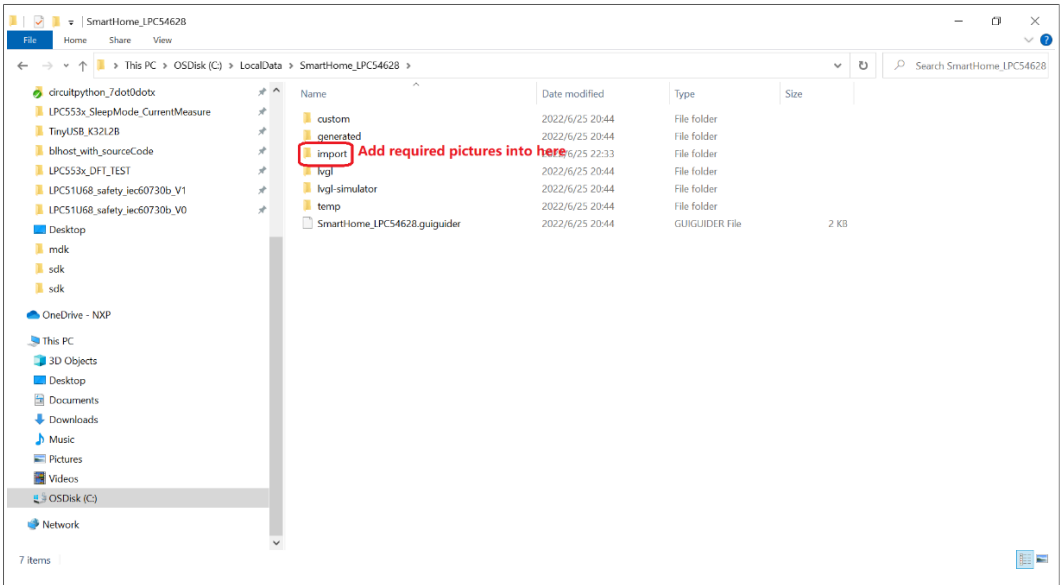

Figure 15. Add picture files to import folder

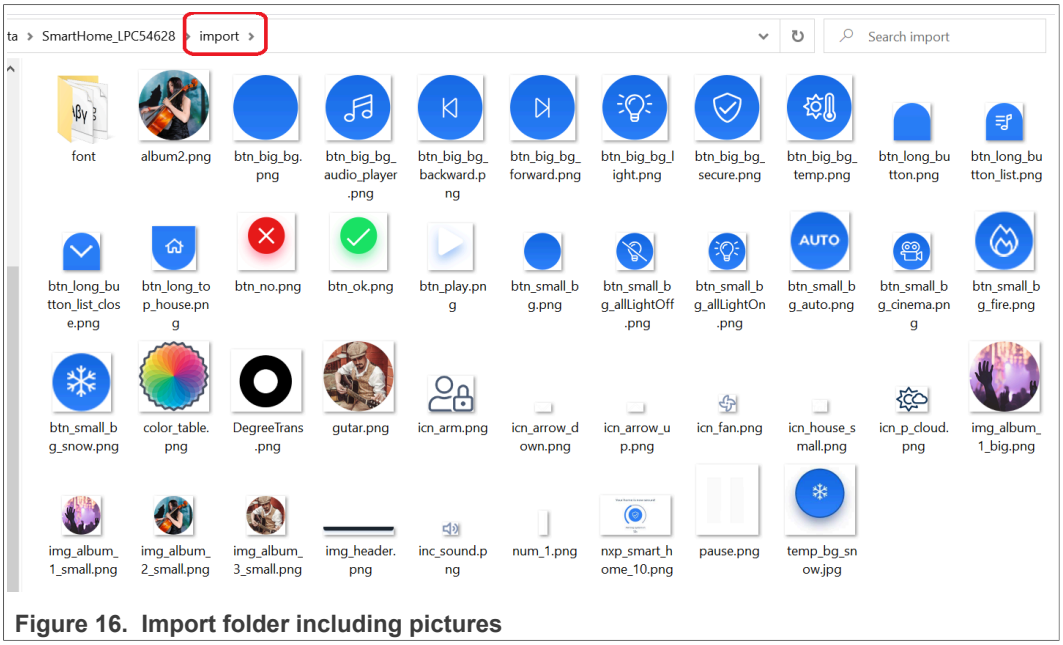

Figure 16. Import folder including pictures

5.3 Human–machine interaction

Different screens show different functions in multi-screen applications. Therefore, it is necessary to provide human-machine interaction to perform screen switching or other functions. Human-machine interaction can be achieved by adding events to the widgets of the GUI.

Consider an example of clicking a button to switch from one screen to another to introduce how to add events to the widget.

The home screen of the smart home demo has a button called Temp. To switch to the TEMPERATURE screen, Temp button on the screen must be clicked, as shown in Figure 17.

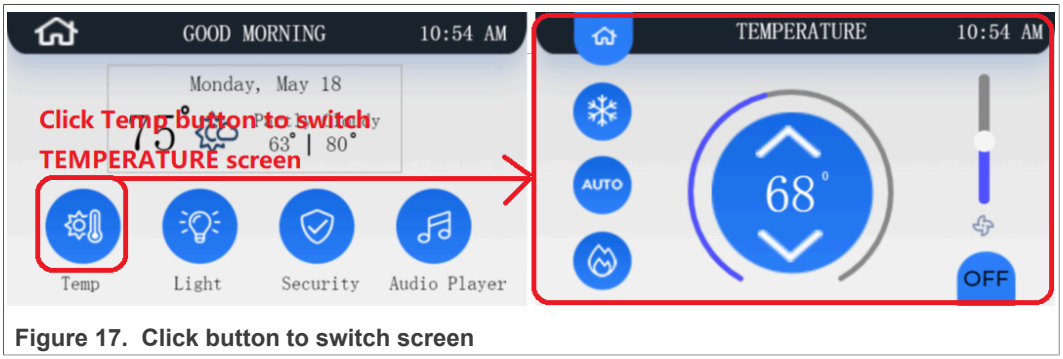

Figure 17. Click button to switch screen

To implement this function, it is necessary to add event to the Temp button, as shown in Figure 18.

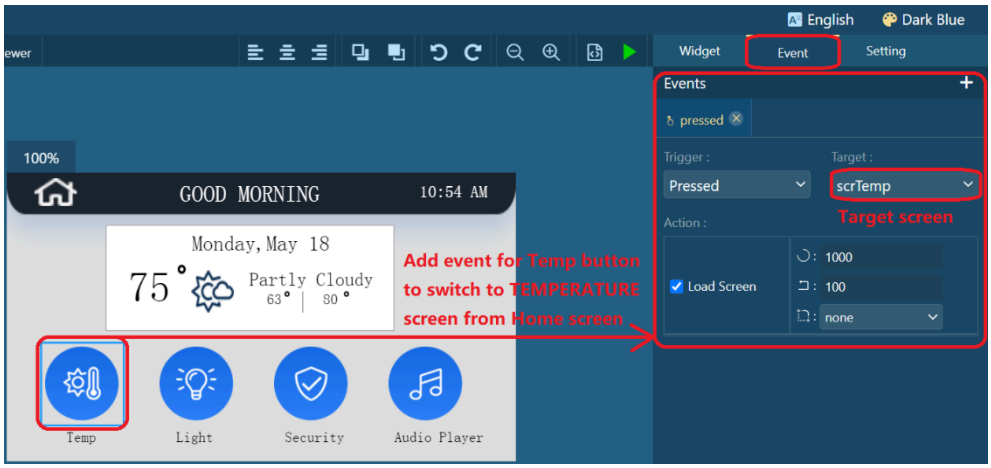

Figure 18. Add event to button widget

5.4 Simulation

After the design is complete, the simulator can be launched to check whether the simulation results match the expected functionality or not. The launching of the simulator is shown in Figure 19.

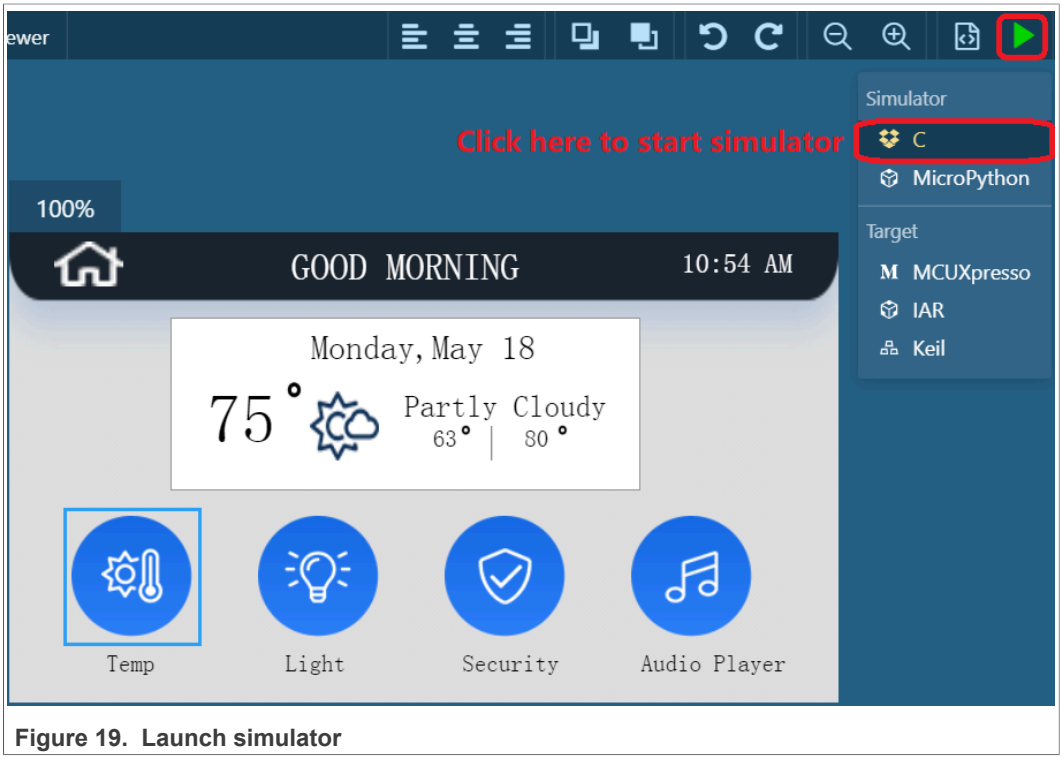

Figure 19. Launch simulator

The running result of smart home demo in the simulator is shown in Figure 20.

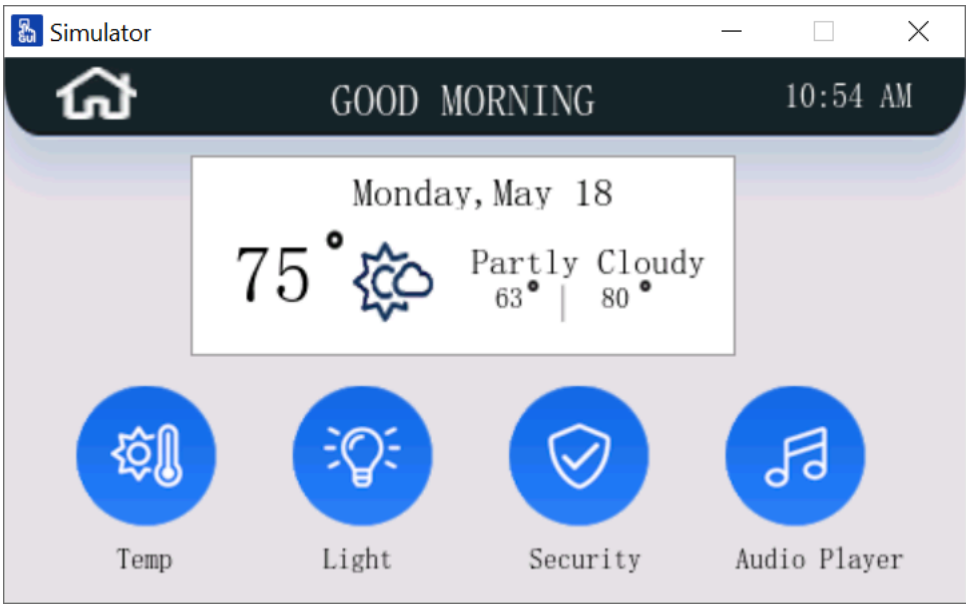

Figure 20. First GUI Screen

5.5 Export Code Project

After the simulation is successfully executed, use the code generation function of GUI Guider to export code based on Keil, IAR, and MCUXpresso IDEs for smart home demo.

As shown in Figure 21, click **Run** > **Target Keil** to export the code. Certainly, it is also possible to export IAR-based or MCUXpresso-based projects.

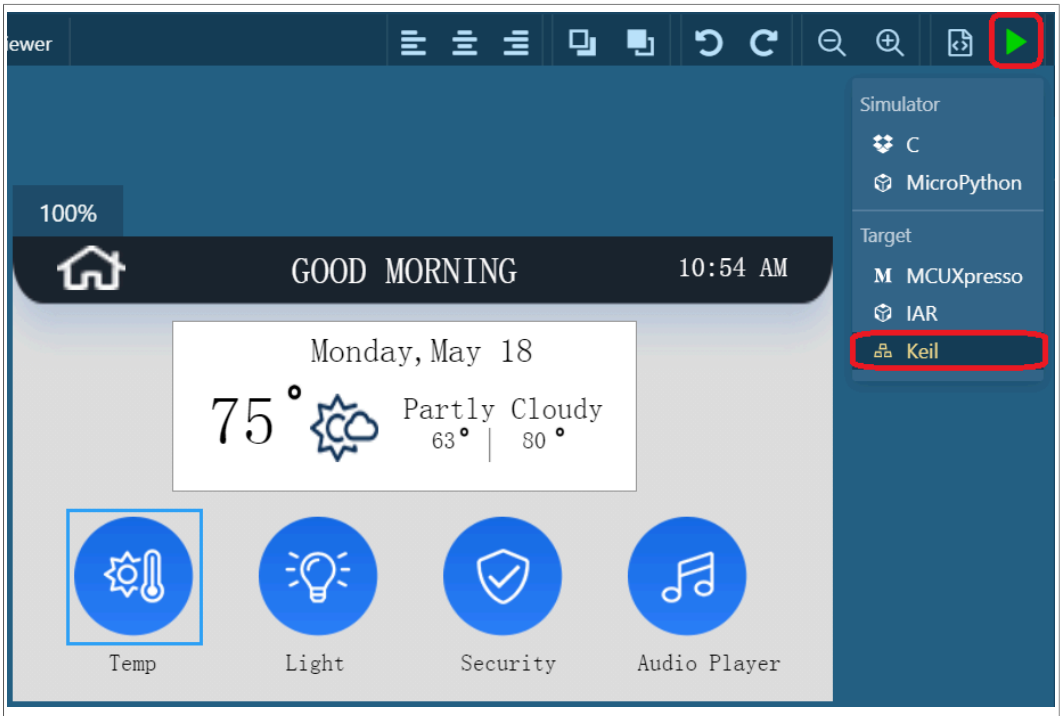

Figure 21. Export code project

5.6 External SPI Flash Support

After the code of smart home demo is exported, compile it. Figure 22 shows compilation errors in the compilation output window.

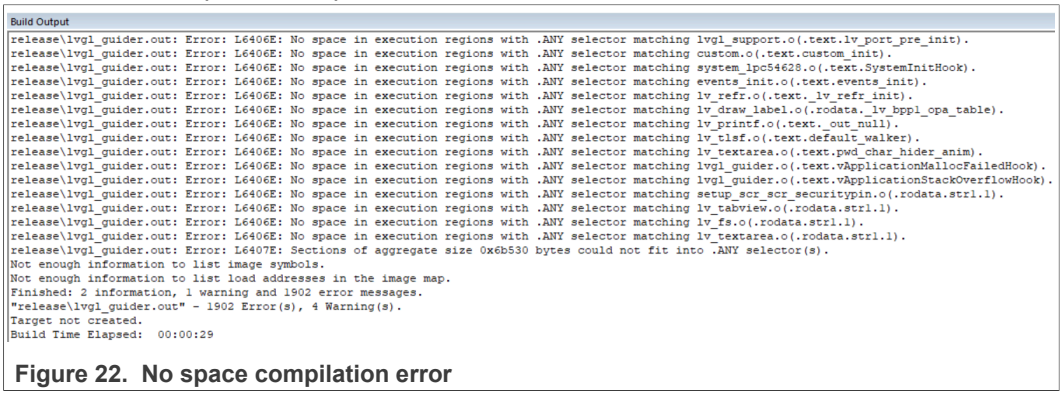

Figure 22. No space compilation error

When GUI Guider with version 1.3.1 exports code, the **Code** section and the **RO** section of the application, that is, the picture data, are stored in the on-chip flash by default. All image data cannot be stored in the on-chip flash since LPC54628 only has 512 kB on-chip flash, whereas all picture data is around 800 kB. As a result, several no-space compilation errors are generated when the code is compiled.

The LPCXpresso54628 evaluation board has an onboard SPI flash, therefore we can map the picture data to the SPI flash.

To support the SPI flash, perform the following steps:

1. Set the pin multiplexing function for pins connected to the SPI flash. The corresponding code is located in `pin_mux.c`.
2. Add SPIFI driver including `fsl_spifi.c` and `fsl_spifi.h` to the code project.

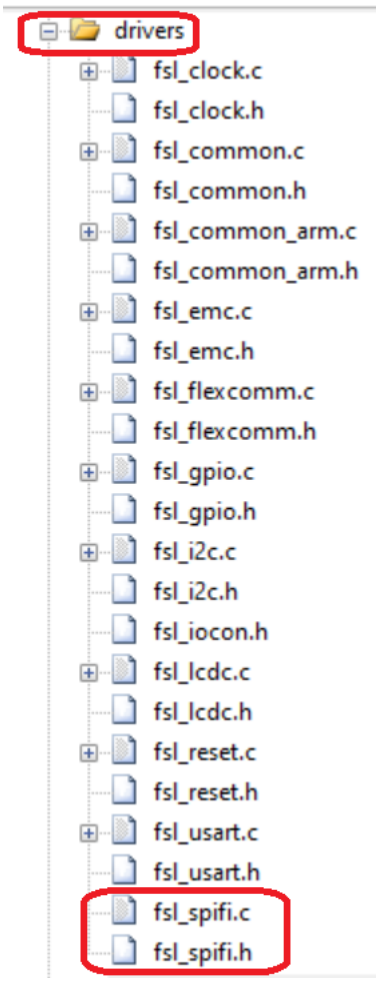

Figure 23. Add SPIFI driver files

3. Add the SPI flash driver to the code.

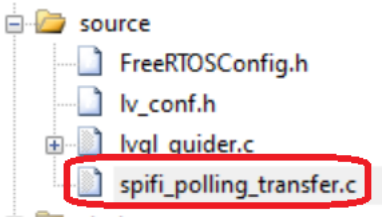

Figure 24. Add SPIFI flash driver file

4. The main function in `lvgl_guider.c` calls the SPI flash initialization function, which is called `spifi_flash_init`, as shown in Figure 25.

```
int main(void)
{
    BaseType_t stat;

    /* Init board hardware. */
    /* attach 12 MHz clock to FLEXCOMM0 (debug console) */
    CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);
    /* Enable the asynchronous bridge */
    SYSCON->ASYNCAPBCTRL = 1;
    /* Use 12 MHz clock for some of the Ctimers */
    CLOCK_AttachClk(kFRO12M_to_ASYNC_APB);

    BOARD_InitPins();
    BOARD_BootClockPLL220M();
    BOARD_InitDebugConsole();
    BOARD_InitSDRAM();
    spifi_flash_init();

    PRINTF("This is Smart home demo for LPC54608!\r\n");
    stat = xTaskCreate(AppTask, "lvgl", configMINIMAL_STACK

    if (pdPASS != stat)
    {
        PRINTF("Failed to create lvgl task");
        while (1)
            ;
    }
```

Figure 25. Call SPI Flash initialization function

5. Modify the scatter file. Provide the address space for SPI flash and map the picture files to the address space corresponding to the SPI flash. The scatter file is `LPC54628J512_flash.scf` and the storage path is deploy\Keil\sdk \LPC54628\arm.

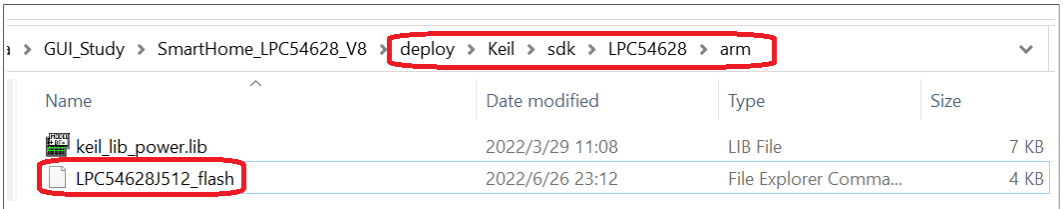

Figure 26. Location of scatter file

```
#define m_interrupts_start          0x00000000
#define m_interrupts_size           0x00000400

#define m_text_start                0x00000400
#define m_text_size                 0x0007FC00

#define m_text_spifi_start          0x10000000
#define m_text_spifi_size           0x1000000

#define m_data_start                0x20000000
#define m_data_size                 0x00028000

#define m_usb_sram_start            0x40100000
#define m_usb_sram_size             0x00002000
```

Figure 27. SPI Flash address space

Figure 28 shows files with the .o extension that are mapped into the SPI flash address space.

```
LR_m_spifi_text m_text_spifi_start m_text_spifi_size {
ER_m_spifi_text m_text_spifi_start m_text_spifi_size {
    _btn_big_bg_130x130.o
    _btn_big_bg_audio_player_alpha_70x70.o
    _btn_big_bg_backward_alpha_51x52.o
    _btn_big_bg_forward_alpha_51x52.o
    _btn_big_bg_light_alpha_70x70.o
    _btn_big_bg_secure_alpha_70x70.o
    _btn_big_bg_temp_alpha_70x70.o
    _btn_long_button_alpha_55x50.o
    _btn_long_button_list_alpha_49x48.o
    _btn_long_button_list_close_alpha_49x48.o
    _btn_long_top_house_alpha_55x50.o
    _btn_no_alpha_80x80.o
    _btn_ok_alpha_80x80.o
    _btn_play_alpha_48x48.o
    _btn_pause_alpha_48x48.o
    _btn_small_bg_allLightOff_alpha_50x50.o
    _btn_small_bg_allLightOn_alpha_50x50.o
    _btn_small_bg_alpha_50x50.o
    _btn_small_bg_auto_alpha_55x55.o
    _btn_small_bg_cinema_alpha_50x50.o
    _btn_small_bg_fire_alpha_55x55.o
    _btn_small_bg_snow_alpha_55x55.o
    _color_table_alpha_140x140
    _DegreeTrans_6x6.o
    _DegreeTrans_10x10.o
    _icn_arm_72x68.o
    _icn_arrow_down_alpha_60x30.o
    _icn_arrow_up_alpha_60x30.o
    _icn_fan_20x20.o
    _icn_house_small_40x26.o
    _icn_p_cloud_40x40.o
    _img_album_1_big_100x100.o
    _img_album_1_small_alpha_40x40.o
    _img_album_2_big_100x100.o
    _img_album_2_small_alpha_40x40.o
    _img_album_3_big_100x100.o
    _img_album_3_small_alpha_40x40.o
    _img_header_480x80.o
    _inc_sound_20x20.o
    _nxp_smart_home_10_480x272.o
  }
}
```

Figure 28. Map picture object files to address space of SPI Flash

These files are objective files compiled from the C files for pictures located in deploy\Keil \sdk\generated\images, as shown in Figure 29.

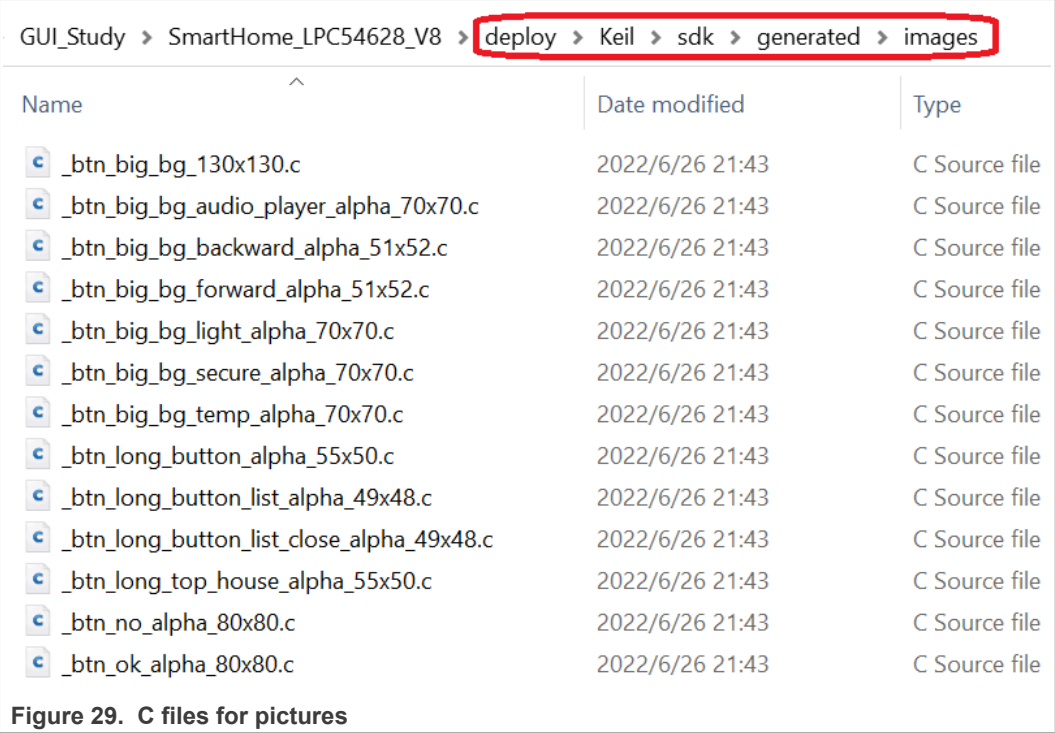

Figure 29. C files for pictures

5.7 Set Flash Programming Algorithm

Since SPI flash is used to store pictures, the programming algorithm of SPI flash must be added to Keil IDE. Thus after the download process starts, the SPI flash can be erased and the picture data can be downloaded to the SPI flash. Figure 30 shows the method of adding flash programming algorithm to Keil.

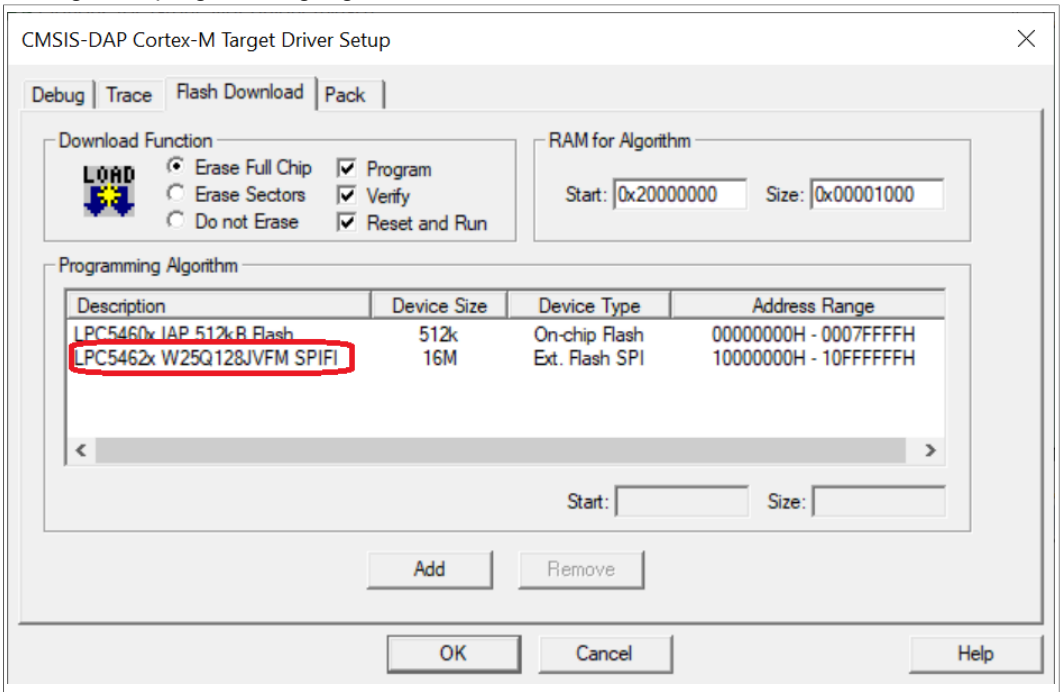

Figure 30. Add SPI Flash program algorithm

5.8 Storage Optimization

To improve the efficiency of memory use, it is necessary to take measures to optimize the use of memory. The commonly used optimization measures are described as follows:

- Set IDE optimization level for Keil IDE as shown in Figure 31.

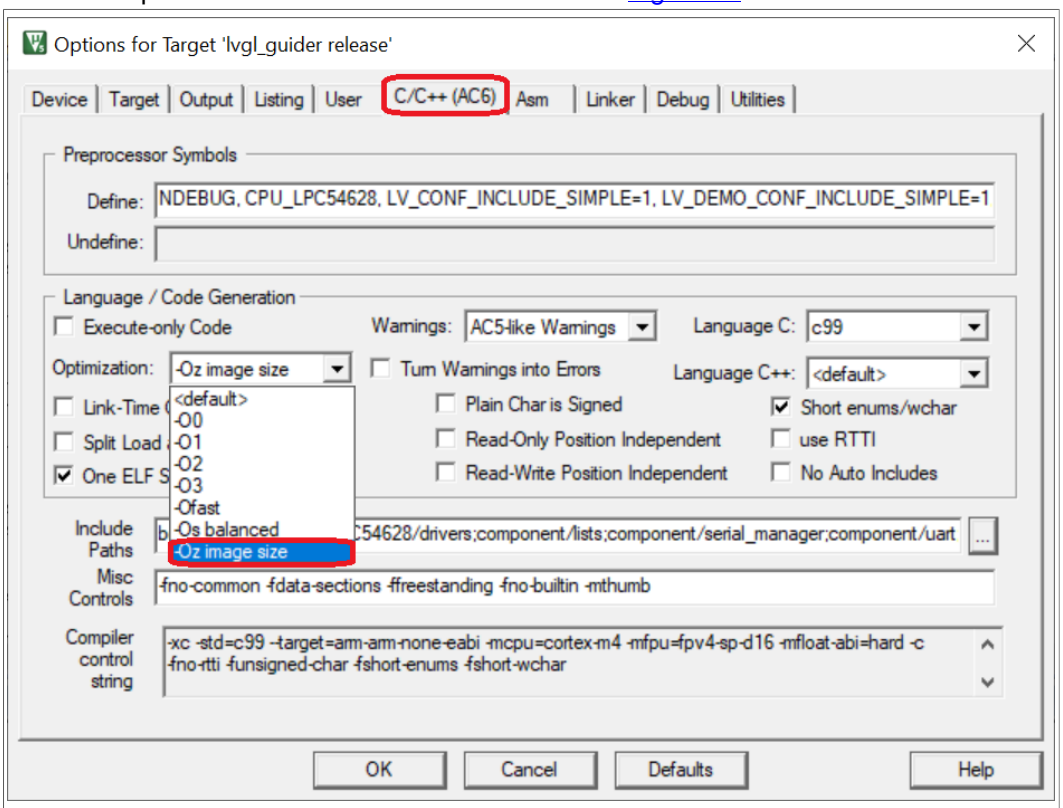

Figure 31. Optimization level setting

- Compile the code with the release version. Figure 32 shows the selection of compiled version in Keil IDE.

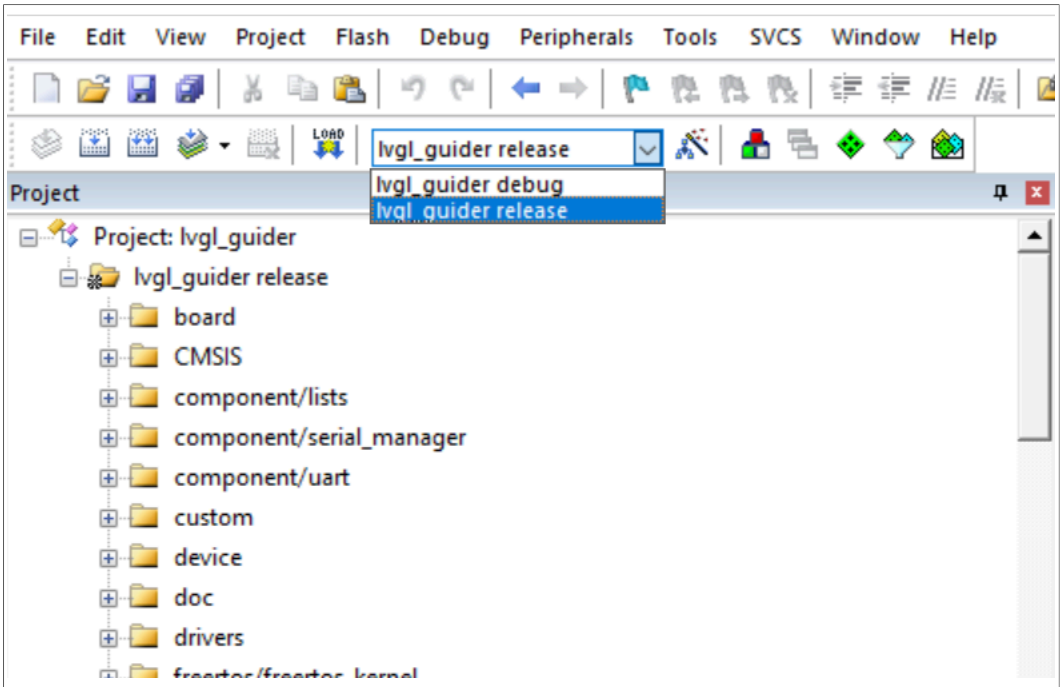

Figure 32. Compile version selection

- Crop the LVGL configuration file called `lv_conf.h` and do not compile the code of the widget that is not used. For example, smart home does not use the Spinner widget. Configure the macro, `LV_USE_SPINNER`, to 0, as shown in Figure 33.

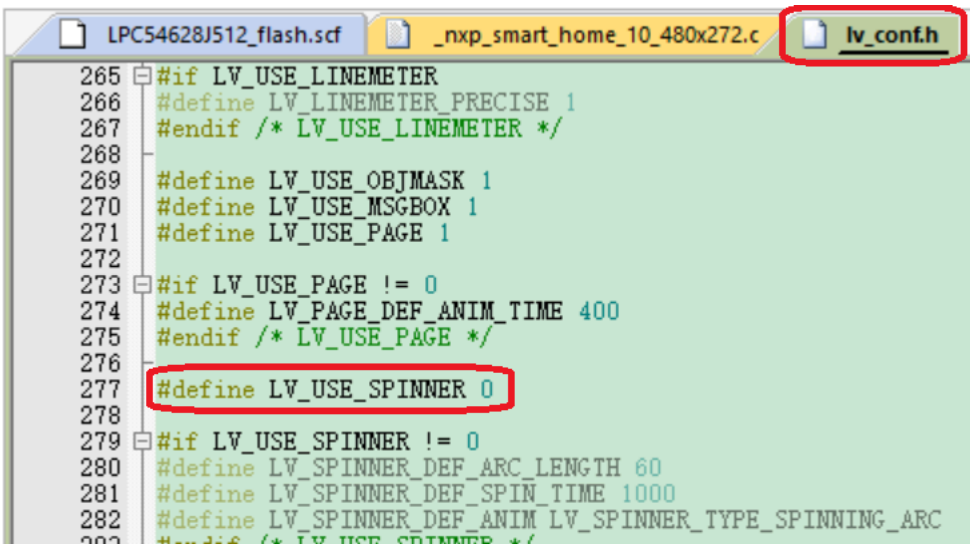

Figure 33. Crop LVGL function

6 Demonstration

This section includes steps for demonstration purpose as follows:

- Download the code project attached to this application note.
- Compile the code project and download the executable binary file to LPC54628 EVK board.

- To start up the application, re-power or reset the EVK board.

7 Revision history

Table 2. Revision history

Rev.	Date	Description
0	20 July 2022	Initial release

8 Legal information

8.1 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.

8.2 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 http://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 data sheet 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.

8.3 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: http://www.nxp.com

Date of release: 20 July 2022
Document identifier: AN13694