AN13450

Smart Home LVGL GUI Demo on GUI Guider

Rev. 0 — 15 December 2021 Application Note

Contents

1 Introduction

The smart home has had a rapid and significant development in recent years. It aims at making our life more convenient, comfortable, efficient, and secure. The smart home is a platform system that can monitor and control varied devices at home. The embedded system plays an important role in the smart home application and the GUI is necessary as well. LVGL (Light and Versatile Graphics Library) is a good GUI library used for embedded system. NXP® has launched and been improving an SW tool called GUI Guider to support strongly LVGL GUI development.

This application note is introducing how to implement a typical smart home GUI demo with a simulator on GUI guider. The reader can learn some skills on GUI Guider for GUI development. And the implemented smart home GUI may be a good reference and a base for a smart home application.

1.1 LVGL

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

Key features:

- Powerful building blocks such as buttons, charts, lists, sliders, images
- Advanced graphics with animations, anti-aliasing, opacity, smooth scrolling
- Various input devices such as touchpad, mouse, keyboard, encoder. and so on
- Multi-language support with UTF-8 encoding
- Multi-display support, for example, connect more TFT, monochrome displays at the same time
- Fully customizable graphic elements
- Hardware independent to use with any microcontroller or display
- Scalable to operate with little memory (64 kB flash, 16 kB RAM)
- OS, external memory, and GPU supported but not required
- Single frame buffer operation even with advanced graphical effects
- Written in C language for maximal compatibility (C++ compatible)
- Simulator to start embedded GUI design on a PC without embedded hardware
- Binding to MicroPython
- Tutorials, examples, themes for rapid GUI design
- Free and open-source under MIT License

1.2 GUI Guider

GUI Guider is a user-friendly graphical user interface development tool from NXP that enables the rapid development of high-quality displays with the LVGL Open-Source Graphics Library. GUI Guider's drag-and-drop editor makes it easy to utilize the many features of LVGL such as 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 your project, accelerating the development process and allowing you to seamlessly add an embedded user interface to your application.

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

2 Introductions of smart home GUI demo

The smart home GUI demo is developed using a simulator on GUI Guider. Therefore the GUI demo is independent of MCU hardware. However, the generated GUI related code can be applied directly to the embedded system's project based on different MCU parts. With GUI Guider, the simulated demo implements not only the layout containing the widgets but also man-machine interactive on GUI. Four functions of the smart home are presented in this GUI demo. They are temperature control, light control, security control, and audio player. The display of the home screen shows the four functions and the basic layout design of the GUI, see Figure 1.

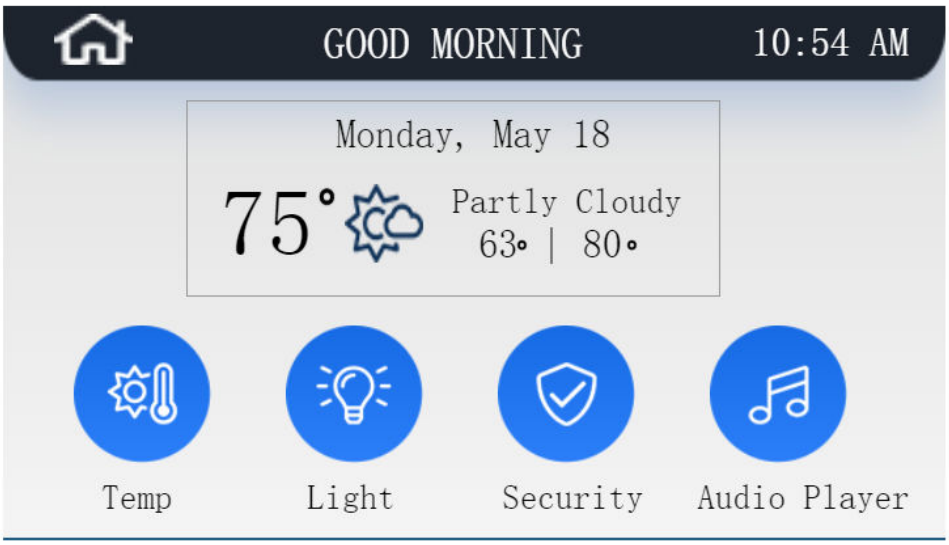
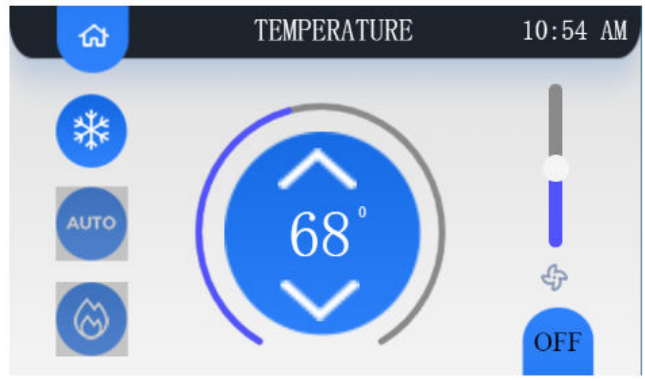


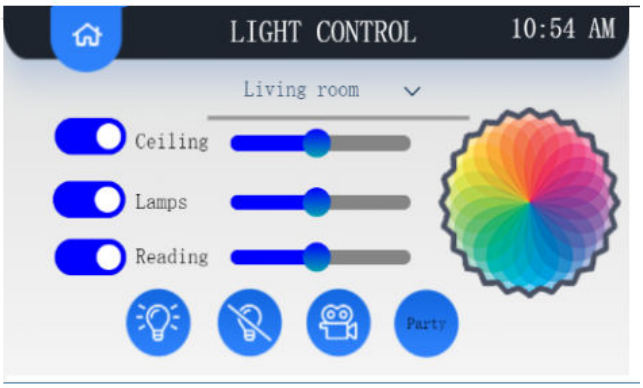
Figure 1. Home screen of smart home GUI demo on GUI Guider

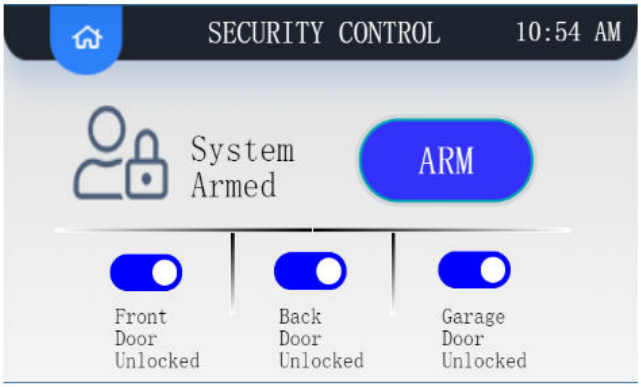
NOTE

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

For the other main screens which show the four functions, see Figure 2.







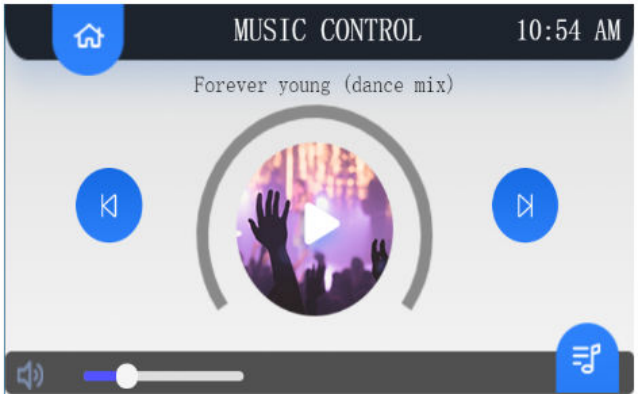


Figure 2. Main screens for temperature, light, security, and audio player

3 Implementation of smart home GUI demo

The GUI layout implementation on GUI Guider is not so difficult. This application note introduces the basic process of GUI development and implementations of several advanced animation functions on GUI Guider. Before this, the development environment should be set up.

3.1 Development environment

The development environment is simple since using a simulator on GUI Guider.

3.1.1 Hardware environment

This action needs a PC to set the hardware environment.

3.1.2 Software environment

This action required the GUI Guider V1.2.0 to set the software environment.

Remark:

- The resources about GUI Guider, including an overview, downloading, documents, training, for example, can be seen on NXP official website at GUI Guider.
- Read the user manual of GUI Guider for your installation and know about the tool at GUI Guider. After installation, the user manual can be also seen under the path: `GUI Guider\DOCUMENTATION\`.
- It is easy to use GUI Guider. This application note does not introduce the usage.

NOTE

Users can read the user manual or training materials at GUI Guider.

3.2 Basic process of development on GUI Guider

The basic process of the development is to create a GUI project, develop and run a simulator (including building, running, and generating code).

Create GUI project:

1. Defining the project name and directory.
2. Create the project with the default settings.
    a. Select a simulator with empty application template.
    b. For the LCD panel.
        i. Select the color depth of 16bit.
        ii. Select the resolutions of 480*272.

For details, see Figure 3.

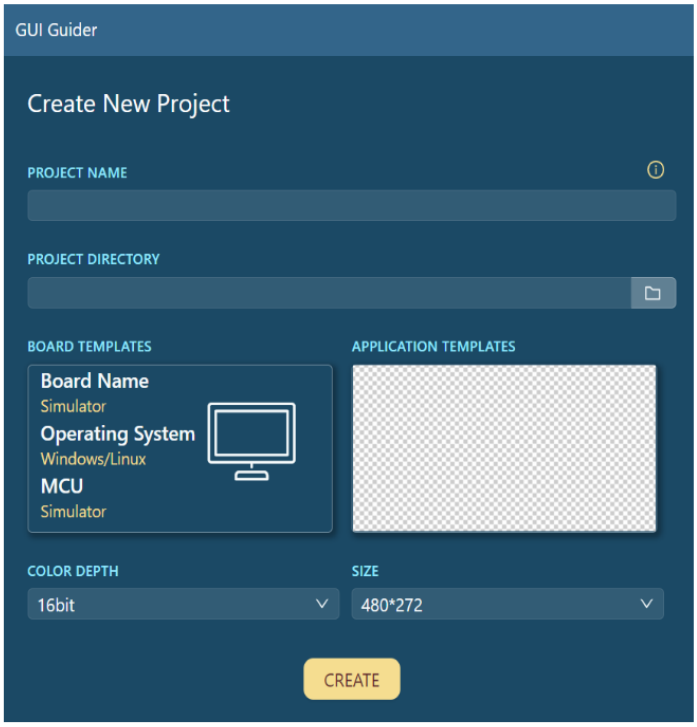


Figure 3. GUI project setup on GUI Guider

Generate GUI code:

1. Develop the GUI on GUI Guider.
2. Run simulator for building code and doing a test.
3. Once the simulator runs successfully, the verified code gets generated.

NOTE

Clicking the **Generate Code** icon can also generate code without being built successfully and there might be some error.

Both “custom” and “generated” subfolders in the package of the GUI Guider project are what the developers must add to their application project. The code created by the developer is stored in the folder named `custom` where there are initially the template files `custom.c`, `custom.h`, and the code generated by GUI Guider are stored in the folder named “generated” where the image and font resources are stored.

NOTE

Do not edit the files in the “generated” folder as anything edited manually gets rewritten automatically by GUI Guider when performing the code generation. The user code should be added to the “custom” folder.

3.3 Implementation of advanced animation functions

As we know, plenty of widgets with varied styles are necessary to make a beautiful visual effect in GUI. Animation is a very important and helpful way to get a better visual effect.

GUI Guider supports making an animation of a widget by simple configurations in “MOVE ANIMATION” item of widget settings. Sometimes, it needs coding to implement some advanced animation functions. There is a benefit of a feature on GUI Guider that it supports the event settings with coding function on a widget so that it allows the developer to handle event on GUI Guider. This brings a significant convenience and chance to the developers to implement and observe the more visual effect on GUI Guider.

Two advanced animations in the smart home GUI demo are introduced in the following section.

3.3.1 Animation button

All the buttons in the smart home GUI demo are designed to move down a short distance and up to the original position to show an animation effect when they are released. This is done by event settings and adding the code. For the typical settings for the **Events** button, see Figure 4,

1. Select the current button in the **Source Widget**.
2. Select **Released** from the **Trigger** list.
3. Select **Null** from the **target widget** list.
4. Select **C code...** from the **Action** list, because the animation action of the button is customized to apply for all buttons.
5. Add the included "C" coding file in the **Include** field.
6. Type the user’s code in **Code** field.

As a typical example, Figure 4 shows the event settings of the button labeled **Temp** in the home screen, see Figure 1. The function is that the button moves down for 10 pixels and up to the original position along the Y coordinate when the clicked button is released and then the screen for temperature control is loaded by calling the function `loadTempScreen()`.

NOTE

It is recommended to the added code are brief as possible, for example, using API call like `btnAnimCustom()`, see Figure 4.

The code gets generated automatically in the file `events_init.c` in the folder `generated` when running the simulator. See Figure 5 for the code generated from the settings in Figure 4.

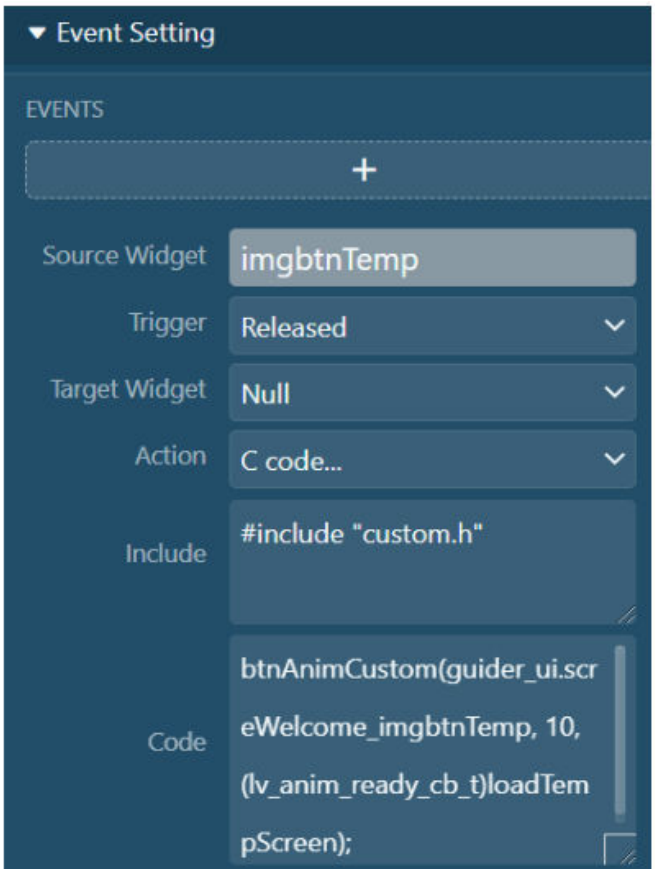


Figure 4. Event settings for "Animation" button

```
static void screWelcome_imgbtnTempevent_handler(lv_obj_t * obj, lv_event_t event)
{
  switch (event)
  {
  case LV_EVENT_RELEASED:
  {
    btnAnimCustom(guider_ui.screWelcome_imgbtnTemp, 10, (lv_anim_ready_cb_t)loadTempScreen);
  }
    break;
  default:
    break;
  }
}
```

Figure 5. Generated code for "Event" button

The function `btnAnimCustom()` is implemented in the file `custom.c`. The function is described as below:

Prototype: `void btnAnimCustom(lv_obj_t *obj, int delta, lv_anim_ready_cb_t animCb)`

Function: Move a widget object down for a distance and up to the original position along with "Y" coordinate. Then call a callback function. This is used for **Animation** button and **Event** button handling.

Parameters:

- `obj` – a widget object
- `delta` – a distance

- `animCb` – callback function called when the animation is ready. It is used for event handling after the animation

See Figure 6 for the implementation of the function.

```
void btnAnimCustom(lv_obj_t *obj, int delta, lv_anim_ready_cb_t animCb)
{
    lv_anim_t a;

    //Initialize an animation variable
    lv_anim_init(&a);
    //Set the function "lv_obj_set_y()" to set the y coordinate in animation
    lv_anim_set_exec_cb(&a, (lv_anim_exec_xcb_t)lv_obj_set_y);
    //Set the object for animation
    lv_anim_set_var(&a, obj);
    //Set the duration of animation to 100ms
    lv_anim_set_time(&a, 100);
    //Get the current y coordinate
    int y = lv_obj_get_y(obj);
    //Set the start and end value of the animation on y coordinate
    lv_anim_set_values(&a, y, y+delta);
    //Set the duration of animation to play back to 100ms
    lv_anim_set_playback_time(&a, 100);
    lv_anim_set_delay(&a, 0);
    if(animCb)
    {//Set the function call when the animation is ready
        lv_anim_set_ready_cb(&a, animCb);
    }
    //Create the animation
    lv_anim_start(&a);
}
```

Figure 6. Customizing code for "Animation" button

This API function is used for all of the **Animations** button in this demo.

3.3.2 Image rotation

In this smart home GUI demo, there is a function of audio player. In the GUI window, there is a function that the album image rotates around its center with music playing once clicked **Play** button, see Figure 7.

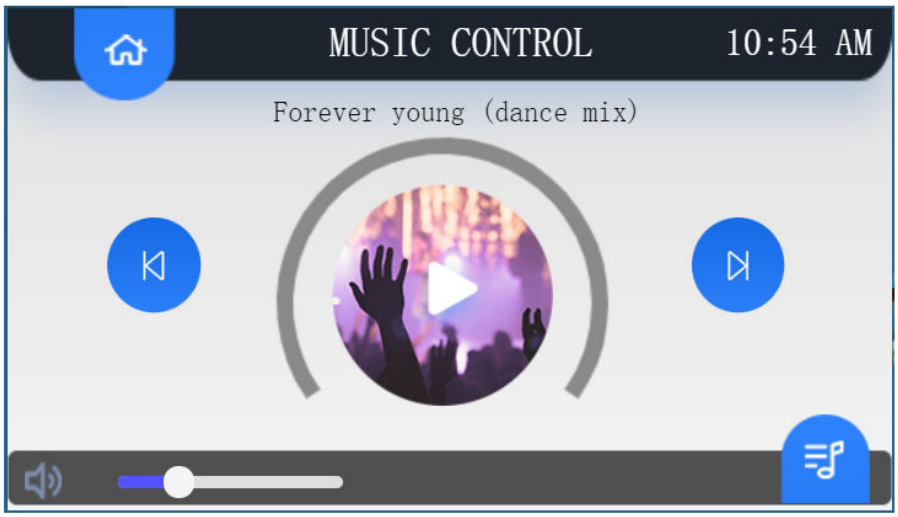


Figure 7. Screen capture from GUI, audio player

How to make the image rotation: It is required to do the event settings for the audio **Play** button since the rotation is configured through the **Event** button setting. See Figure 8 for the event settings which is very similar to the Figure 4.

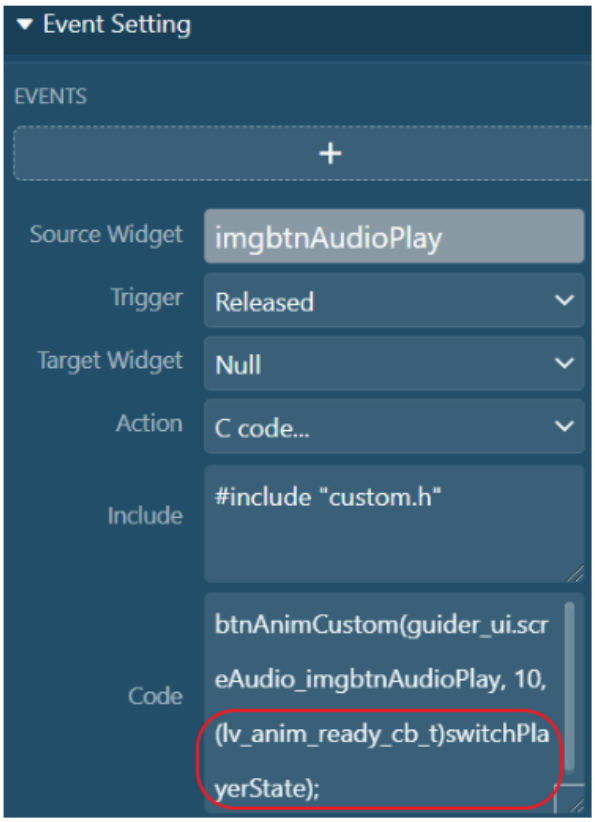


Figure 8. Event settings for audio "Play" button

The code function `btnAnimCustom()` in the **Code** field is called for the audio play button animation and the callback code function switchPlayerState() is called for the image rotation. The callback function is customized code that is implemented in `custom.c`.

The album image rotation is essentially an animation. The implementation is same as the button animation in Figure 6. There are two major differences:

1. The code function set in `lv_anim_set_exec_cb()` for animation is `lv_img_set_angle()` since the animation is to rotate an image. The code function `lv_img_set_angle()` is to set the rotation angle of the image. In animation, the function code changes the angle of image rotation.
2. The start and end values of the animation set in `lv_anim_set_values()` are individually 0 degrees and 3600 degrees (the solution is 0.1 degrees) for rotation.

Read the functions `switchPlayerState()` and `initAudioAlbumImgAnim()` in the file `custom.c` for the details of the implementation.

Rotation center: The rotation center is the center of the album image widget in this demo as mentioned in the above section. The coordinate of the center is relative to the image and can be configured in the settings of the widget. For the related settings of the album image, see Figure 9. The "X" and "Y" coordinate for the ROTATE CENTER are set to 60 and 60 relative to the image size (120 x 120).

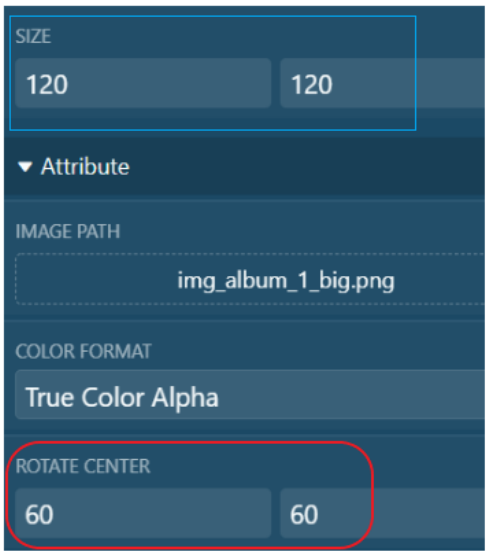


Figure 9. ROTATE CENTER settings for album image

4 Demonstration

Follow the steps for demonstration purpose:

1. Unzip the attached software package.
2. Open GUI Guider and import the project "`SmartHome_GARC1.guiguider`" under the root path of the SW package.
3. Click "Run Simulator" to run the demo.

   With the demo running, the user can try to experience the GUI by clicking the buttons on GUI window, the operation is very simple.

NOTE

In the **SECURITY** dialog box, enter the password "1234".

5 Revision history

Table 1 summarizes the changes done to this document since the initial release.

Table 1. Revision history

Revision number	Date	Substantive changes
0	15 December 2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

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.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. 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.

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

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 15 December 2021

Document identifier: AN13450

arm