AN13334

LPC5460x LVGL GUI Demo Development with GUI Guider

Rev. 0 — 16 August 2021 Application Note

Contents

1 Introduction

As well-known, GUI is widely used in an embedded application. A good GUI helps a product more competitive. However, it is not easy to do it. To make the development of GUI more easy, much GUI mid-wares come forth. NXP has introduced the emWin GUI on NXP MCUs for some years and many customers have used it successfully in their products. Recently, NXP has promoted one more GUI mid-ware called Light and Versatile Graphics Library (LVGL) for more choices to the users. As you know, developing GUI application should still be a heavy work even with GUI mid-ware. NXP launched an SW tool called GUI Guider to support strongly LVGL GUI development.

So how to implement an LVGL GUI application with GUI Guider's assistance?

This application note is trying to show it by introducing how to implement a typical GUI demo on LVGL with GUI Guider based on LPC54608.

2 Overview

This section gives a brief overview of the LPC546xx family, Light and Versatile Graphics Library (LVGL), and GUI Guider.

2.1 LPC546xx

The LPC546xx family includes up to 512 KB of flash, 200 KB of on-chip SRAM, up to 16 kB of EEPROM memory, a quad SPI Flash Interface (SPIFI) for expanding program memory, one high-speed and one full-speed USB host and device controller, Ethernet AVB, LCD controller, smart card interfaces, SD / MMC, CAN FD, an External Memory Controller (EMC), a DMIC subsystem with PDM microphone interface and I^2S, five general-purpose timers, SCTimer/PWM, RTC / alarm timer, Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), ten flexible serial communication peripherals (USART, SPI, I^2S, I^2C interface), Secure Hash Algorithm (SHA), 12-bit 5.0Msamples/sec ADC, and a temperature sensor.

Features relevant to GUI display:

- The LPC5460x / 61x devices operate at CPU frequencies of up to 180 MHz. The LPC54628 device operates at CPU frequencies of up to 220 MHz.
- LCD Controller supporting both Super-Twisted Nematic (STN) and Thin-Film Transistor (TFT) displays. It has a dedicated DMA controller, selectable display resolution (up to 1024 x 768 pixels), and supports up to 24-bit true-color mode.
- External Memory Controller (EMC) provides support for asynchronous static memory devices such as RAM, ROM, and flash, in addition to dynamic memories such as single data rate SDRAM with an SDRAM clock of up to 100 MHz.

2.2 Light and Versatile Graphics Library (LVGL)

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

Key features:

- Powerful building blocks, for example, buttons, charts, lists, sliders and, images.

- Advanced graphics with animations, anti-aliasing, opacity and, smooth scrolling.
- Various input devices, for example, touchpad, mouse, keyboard and, encoder.
- Multi-language support with UTF-8 encoding.
- Multi-display support, for example, uses more TFT, monochrome displays simultaneously.
- 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 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.

2.3 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.

3 GUI demo design

GUI is a graphical user interface and a GUI application generally presents user a graphical interface to operate the device. A user-friendly GUI should be easy-to-use and have a beautiful visual effect. A typical GUI is to present icons for user to select different functions in an embedded product, for example, home appliance. Different configurations are also usually presented by icons in a GUI. The GUI demo introduced in this application note is designed referring to the typical GUI application. This GUI demo presents some icons for user to select one by moving and focusing a square on it with a key input after loading a startup interface where a logo is displayed with animation effect. And it provides a measure for refresh rate as well since the users usually concern it. The demo is implemented without OS since it is typically non-OS in the embedded devices based on MCU.

This application note is introducing how to implement the typical GUI demo without OS on LVGL with GUI Guider on LPCXpresso54608 EVB. Hope it could be a reference and helpful for the engineers to develop their GUI application.

4 GUI demo implementation

This section describes the development environment for the hardware and software (to run them as well) also, they are easy to be set up.

4.1 Hardware environment

- LPCXpresso54608 EVB Rev B with LCD module
- One micro USB cable

- PC

4.2 Software environment

- SDK_2.9.0_LPCXpresso54628
- KEIL MDK V5.33.0.0
- GUI Guider V1.0.0.

Remark:

- The resources about GUI Guider, including overview, downloading, documents, training, and so on, can be seen on GUI Guider .
- Refer to the *GUI Guider User's Guide* (document GUIGUIDERUG) for your installation and also to know about the tool. After installation, the user manual can be also seen under the path: GUI-Guider\resources\.
- It is easy to use GUI Guider. This application note does not introduce the usage. Users can read the *GUI Guider User's Guide* (document GUIGUIDERUG) or training materials at GUI Guider for the usage.

4.3 Develop LVGL non-OS GUI demo in KEIL with GUI Guider

The GUI demo is implemented with KEIL IDE based on the non-OS LVGL SDK example `littlevgl_demo_widgets_bm` under the path: `SDK_2.9.0_LPCXpresso54628\boards\lpcxpresso54628\littlevgl_examples\`.

The basic process to develop the GUI demo with GUI Guider is:

1. Create GUI project: Create a GUI project with LPC54628 board template and empty application template in GUI Guider. For the basic setup of the GUI project in GUI Guider, see Figure 1.

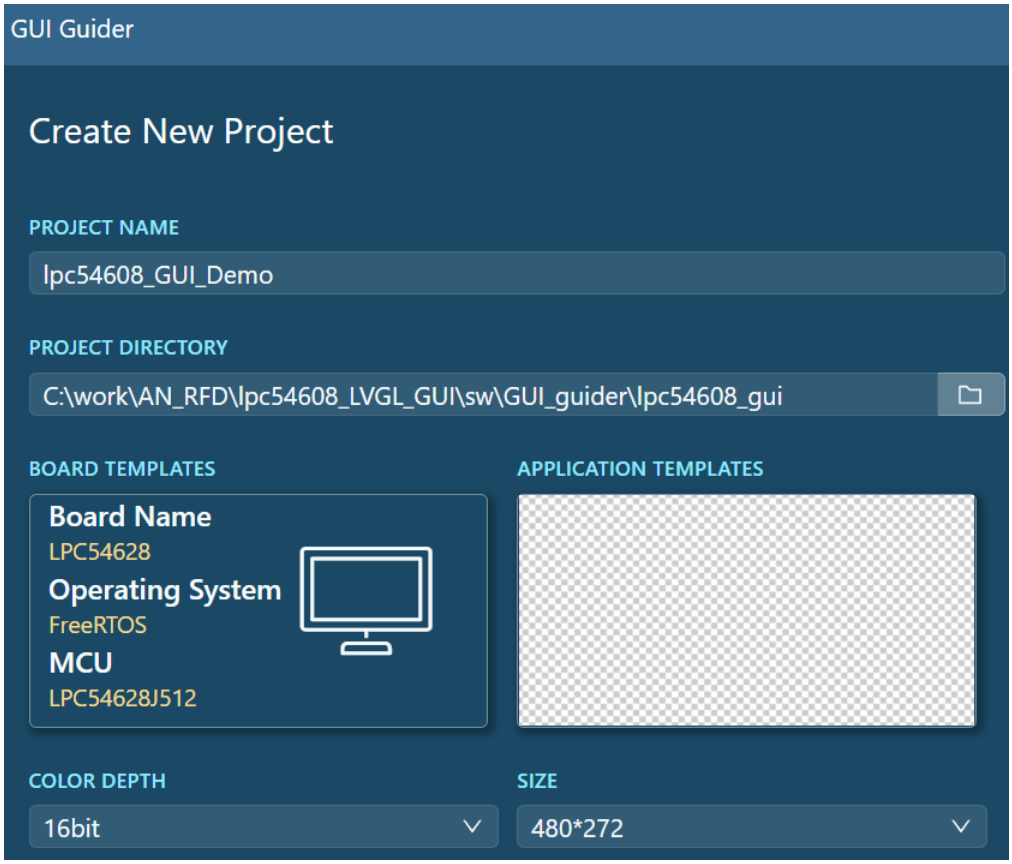

Figure 1. GUI project setup in GUI Guider

2. Generate GUI code: Develop the GUI in GUI Guider. The GUI code is provided in three subfolders under `export` folder when they are generated and exported, see Figure 2. The code files in the 3 folders except for the file `littlevgl_demo_widgets` in `source` folder must be added into the KEIL project.

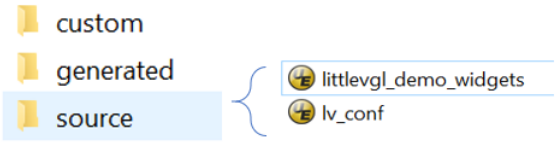

Figure 2. Folders containing GUI code from GUI Guider

3. Create KEIL IDE project: Clone the existing SDK LVGL example folder `littlevgl_demo_widgets_bm` to create a new folder `littlevgl_guider_demo_bm` at the same path and create our GUI demo project with KEIL IDE.
4. Add GUI code: Copy folders `custom` and `generated` and the file `lv_conf.h` in `source` folder into the folder `littlevgl_guider_demo_bm`. Let `lv_conf.h` replace the old one.
5. Configure GUI demo project: Configure the GUI demo project for building it successfully. It mainly contains the below sides:
    a. Add the new groups and files relevant to the GUI code to the KEIL project and remove the unwanted ones from the cloned SDK example project.
    b. Add compiler include paths relevant to the GUI code files.
6. Modify the file containing the main(): In this GUI demo, the file is `littlevgl_guider_demo_bm.c` cloned from the one `littlevgl_demo_widgets_bm.c` in the SDK example. In this case, it is necessary to do modifications as below:
    a. Add and remove related include files in main `()` file, as in following code:

```
#include "gui_guider.h"
#include "events_init.h"
#include "custom.h"
//#include "lv_examples/src/lv_demo_widgets/lv_demo_widgets.h"
```

    b. Change the call `BOARD_BootClockPLL220M()` to `BOARD_BootClockPLL180M()` since the lpc54608 part on the development board has up to 180 MHz of main clock.
    c. Change the main call for GUI setup: In this demo, delete the old call `lv_demo_widgets()` and add the one `setup_ui(&guider_ui)`.

    More coding could be needed in the file depending on the application.

The above is the basic process of the GUI demo development. More details of the GUI demo implementation are introduced in following section of this document.

4.4 Startup GUI

Typically, a GUI application shows a company's logo with some animation effect on system startup. In this GUI demo, it moves the "NXP" logo to the center of screen with bounce effect meanwhile moving a label `LPC546xx GUI Demo`. It is easy to be implemented with GUI Guider. Dragging and dropping an image widget for adding the logo image of NXP and a label for editing its text to `LPC546xx GUI Demo`. Configure the basic attributions, for example, size, position, text color, depending on the requirements. The animations of the image and label can be easily configurated on GUI Guider.

Figure 3 shows the configurations of the animation of the logo NXP: Enabling the animation by checking `MOVE ANIMATION`; set the destination positions of X and Y coordinates to "140" and "86"; set the animation path to `bounce` (it has a bounce effect when moving to the destination); set the duration to "1000" that means the duration of moving is for 1 second.

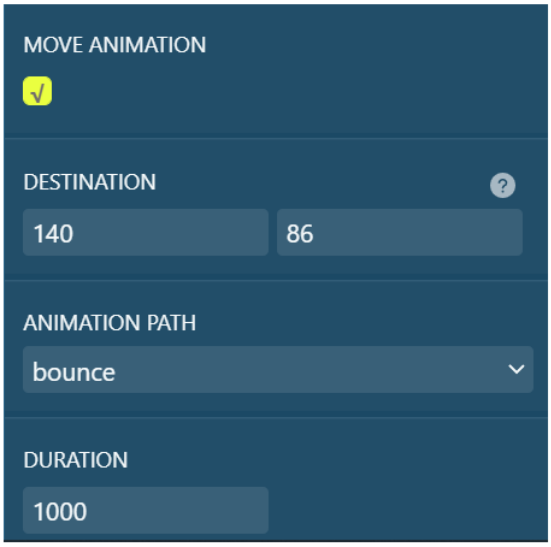

Figure 3. Animation configurations for startup GUI

When running simulator, the startup GUI is displayed as in Figure 4, after moving.

Figure 4. Startup GUI

In the actual GUI demo, the background color of the screen is designed to be black. May modify it in KEIL project as in the following code (bold font):

```
//Write codes screen
ui->screen = lv_obj_create(NULL, NULL);

//added to change the bg color of screen to black
lv_obj_set_style_local_bg_color(ui->screen, LV_OBJ_PART_MAIN, LV_STATE_DEFAULT, LV_COLOR_BLACK);
```

4.5 Main GUI

As mentioned, the main GUI is implemented referring to a typical GUI application in consumer and industry. Basically, the main GUI is implemented as follows:

The main GUI presents six icons with a square focusing on one as the selected in Figure 5. With a key input, the square is moved left/right to focus on an icon which is presented in the screen. When the square has reached to the most left / right icon in the screen, it stops moving while the icons are moved left / right one by one so that a new icon is presented with an old one invisible in the screen until there is no more new icon to be shown.

There is a label on the left top of the screen, see Figure 5. It shows the name of the current focused icon with the movement.

Key features:

- To make the moving look more smooth for a better visual effect, each movement from one icon to the next is broken into four frames to process automatically with a key input.
- A total of 12 icons are used for the demo. Six icons are displayed in a screen, see Figure 5. The size of each icon is "48*48" (so they can be stored in SRAM and so it does in the demo).
- LCD module on the *LPCXpresso54608 EVB* has: 480*272 resolution, 16bit color depth. The frame buffer is located at SDRAM and double frame buffer is used.
- There is a special handling for the keypad input in the demo as only one key (SW5) is available in the five keys on the EVB board. In the remainder, one is reset key and the IOs connected to other keys are multiplexed with the bus IOs of SDRAM which is used for LCD frame buffer. So a switch widget is taken to configure the only available key(SW5) as left or right key for controlling the moving direction (left / right).

For display effect of the main GUI, see Figure 5.

Figure 5. Main GUI

Development in GUI Guider:

To develop the main GUI with GUI Guider, we can take advantage of the image widget of GUI Guider to convert the icons to the C code conforming to the requests of LVGL. Additionally, the icons on the screen, see Figure 5 can be created with GUI Guider at first. Later, we can do some modifications in KEIL project based on the generated code by GUI Guider. Thus, it saves much workload.

It is not difficult to implement the main GUI like Figure 5 with GUI Guider. With generating the code for the main GUI from GUI Guider, add them to the KEIL non-OS LVGL GUI project and modify the code as below sides:

- Change the background color of the screen to black by modifying it in KEIL project as in the following code (bold font):

```
//Write codes screen
ui->screen1 = lv_obj_create(NULL, NULL);

//added to change the bg color of screen to black
lv_obj_set_style_local_bg_color(ui->screen1, LV_OBJ_PART_MAIN, LV_STATE_DEFAULT, LV_COLOR_BLACK);
```

And some widgets' color might be required to update accordingly with the background's change.

- Rewrite the code relevant to the creation of the icons based on the code from GUI Guider which is repetitive as each icon is initialized by the similar code. We can optimize it to initialize the icons in a loop with less code as follows:

```
//update to create all icons on screen1 in a loop
for(int i = 0; i < SCREEN1_IMG_NUM; i++)
{//12 icons
   //create image objs for icons
   ui->screen1_img[i] =     lv_img_create(ui->screen1, NULL);

  lv_obj_add_style(ui->screen1_img[i], LV_IMG_PART_MAIN, &style_screen1_img_main);
  if(i < SCREEN1_IMG_SHOW_NUM)
  {//show 6 icons
    lv_obj_set_pos(ui->screen_img[i],       SCREEN_IMG_START_X+i*SCREEN1_IMG_DISTANCE,
SCREEN1_IMG_START_Y);
  }
  else
  {//hide else icons by laying them    outside of screen
   lv_obj_set_pos(ui->screen1_img[i], LV_HOR_RES_MAX, SCREEN1_IMG_START_Y);
  }
 //set the size of icons to w=48, h=48
  lv_obj_set_size(ui->screen1_img[i],  icon_img[i]->header.w,  icon_img[i]->header.h);
  lv_obj_set_click(ui->screen1_img[i], true);
 lv_img_set_src(ui->screen1_img[i], icon_img[i]);
 lv_img_set_pivot(ui->screen_img[i], 0,0);
 lv_img_set_angle(ui->screen_img[i],  0);
}
```

Key functions developed in KEIL IDE:

Some functions are required to be modified or developed in KEIL IDE. For more details, please read the source code referring to the list in Table 1 about the key functions developed in KEIL IDE.

Table 1. Key functions developed in KEIL IDE for reference

Function API	Descriptions	File
`screen1_keypad_events_animove()`	Move a square or icons by four frames with a key input.	`setup_scr_screen1.c`
`screen1_keypad_event_handler()`	Keypad event callback handler to set the control information for movement.	`events_init.c`
`DEMO_ReadKey()`	Keypad input device driver to read keypad value for triggering a keypad event	`Littlevgl_support.c`

5 Refresh rate measurement

Usually, users concern about the refresh performance for a GUI application. The refresh rate is the most important specification to show the performance. This demo provides a way to show it by measuring the refresh rate of the movement in main GUI. This feature can be enabled or disabled by setting a macro definition to 1 or 0 in the file `littlevgl_support.c` as follows.

Enabled the default definition:

```
#define REF_SPEED_MEASURE_EN 1
```

If enabled, the measure function is shown in the left bottom of area of the screen. The function includes two features:

- Real-time data display:

The real-time refresh rate shown in the screen every time the square or icon moves with pressing the key once. It shows the values for four frames and their average value as the movement with each keypad input contains actually four frames of animation (it is, the screen is refreshed four times). The first one in four values is calculated by measuring the time between getting the keypad input event to completing refreshing the first frame. And the remainder is between the completion of the previous frame refreshing to the completion of the next one refreshing.

- Statistics data display:

  The recent measured average data can be shown in a statistic chart when users check a checkbox for data statistics. Up to 10 sets of recent data can be collected and showed in the statistics chart (the chart cannot be shown if no data measured).

  For the display effect of the real-time and statistics data of the refresh rate, see Figure 6.

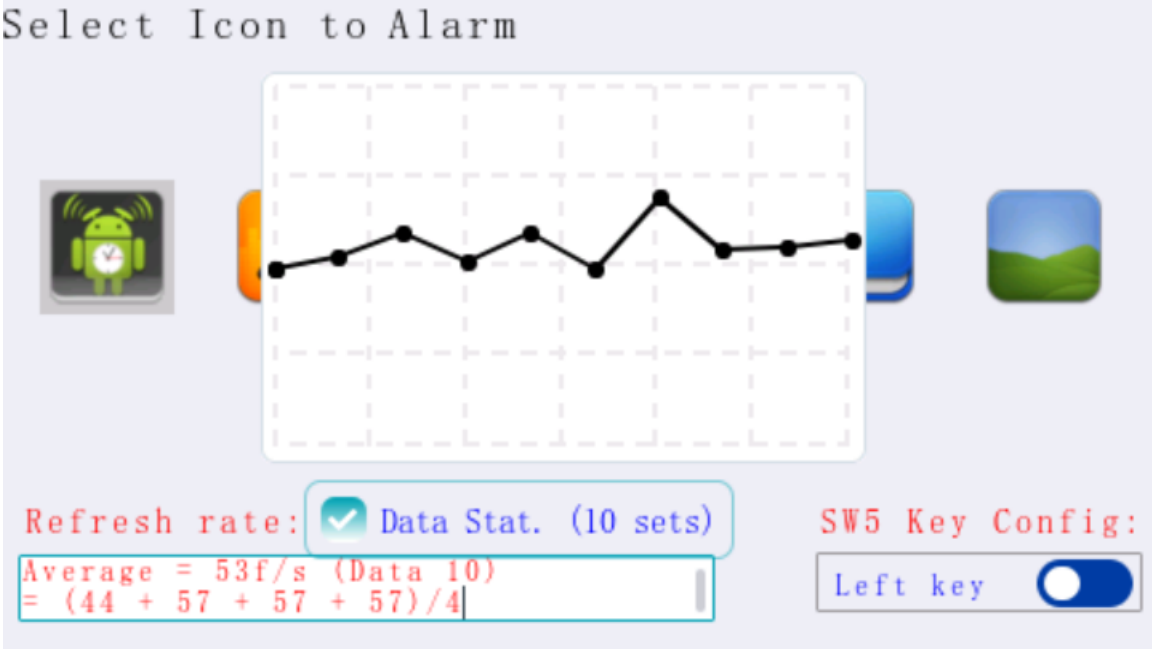

Figure 6. Refresh rate data display

6 Demonstration

6.1 Hardware setup

It is only required to connect PC to USB debug-link port on *LPCXpresso54608 EVB* via the micro USB cable.

6.2 Software setup

Once the attached software package unzips, copy the folder `littlevgl_guider_demo_bm` under the path: `SDK_2.9.0_LPCXpresso54628\boards\lpcxpresso54628\littlevgl_examples\`

6.3 Running and results

Open the KEIL MDK project `littlevgl_guider_demo_bm.uvprojx` under the path of `\ littlevgl_guider_demo_bm\MDK\`, build and download. After running on board, it displays the startup GUI with animation. After a moment when the logo moves to the center of the screen, the main GUI displays automatically, see Figure 4.

At this initial state of the main GUI, the square is focused on the most left icon and the SW5 key is set as left key. So no movement happens when pressing the SW5 key on the board at this moment. It must touch the switch to set the SW5 key as right key.

Then the square moves to the right next icon with pressing SW5 key. Meanwhile the refresh rate is measured and real-time result displayed.

Make sure that the SW5 key has been set to left or right key before want to get left or right movement. If the last icon (12 icons totally) is focused with the square, no movement happens with SW5 key being pressed unless it is switched as left key.

As mentioned, the refresh rate data is displayed in a statistic chart with checking the checkbox *Data Stat*. See Figure 6 when refresh rate data is available.

7 Summary

NXP is promoting a GUI mid-ware – LVGL and released a tool called GUI Guider for LVGL GUI development. This application note is introducing how to implement an LVGL non-OS GUI demo running on LPCXpresso54608 EVB in KEIL IDE with GUI Guider. First, it introduces the basic process of the demo development to show how to create an LVGL non-OS GUI project in KEIL with GUI Guider. Next, it elaborates the implementation of the startup GUI and main GUI. Finally, it tells about showing the refresh performance by measuring and displaying the real-time and statistics data of the refresh rate.

The reference code of the demo is wrapped and attached with this application note. Hope this application note with the reference code could be helpful for users to develop their GUI application on LVGL with GUI Guider.

8 Revision history

Table 2 summarizes the changes done to this document since the initial release.

Table 2. Revision history

Revision number	Date	Substantive changes
0	16 August 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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX,EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. M, M Mobileye and other Mobileye trademarks or logos appearing herein are trademarks of Mobileye Vision Technologies Ltd. in the United States, the EU and/or other jurisdictions.

© NXP B.V. 2021. **All rights reserved.**

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

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

Date of release: 16 August 2021

Document identifier: AN13334

arm