AN12326

Secure General-Purpose Input/Output (GPIO) and Usage

Rev. 2 — 24 July 2023 Application note

Document Information

Information	Content
Keywords	LPC55Sxx, LPC55S69
Abstract	LPC55Sxx (with TrustZone) has secure GPIO module whose usage is closely related to normal GPIO, TrustZone, and secure AHB controller. This document briefly introduces these functions.

1 Background

LPC55Sxx (with TrustZone) has secure GPIO module whose usage is closely related to normal GPIO, TrustZone, and secure AHB controller. This document briefly introduces these functions. For more information, see the user manual.

1.1 TrustZone and secure AHB controller

1.1.1 TrustZone

TrustZone for Armv8-M are available to protect secure resources from malicious code. Such secure resources may include secure memory blocks (code/data), and secure peripherals. It is achieved by segmentation of address space into either Secure (S) or Non-secure (NS). TrustZone can filter address access from CPU0 based on specific security attribute (S, NS) assigned to that address space.

As shown in Figure 1, CM33 CPU in Secure state (CPU-S) can execute instructions from Secure memory (S-memory), but not allowed to execute instructions directly from Non-secure memory (NS-memory). However, CPU- S can access data in both S-memory and NS-memory. CPU-NS can execute instructions only from NS-memory, and not allowed to execute instructions from S-memory. CPU-NS can access data only in NS-memory, but not allowed to access data from S-memory.

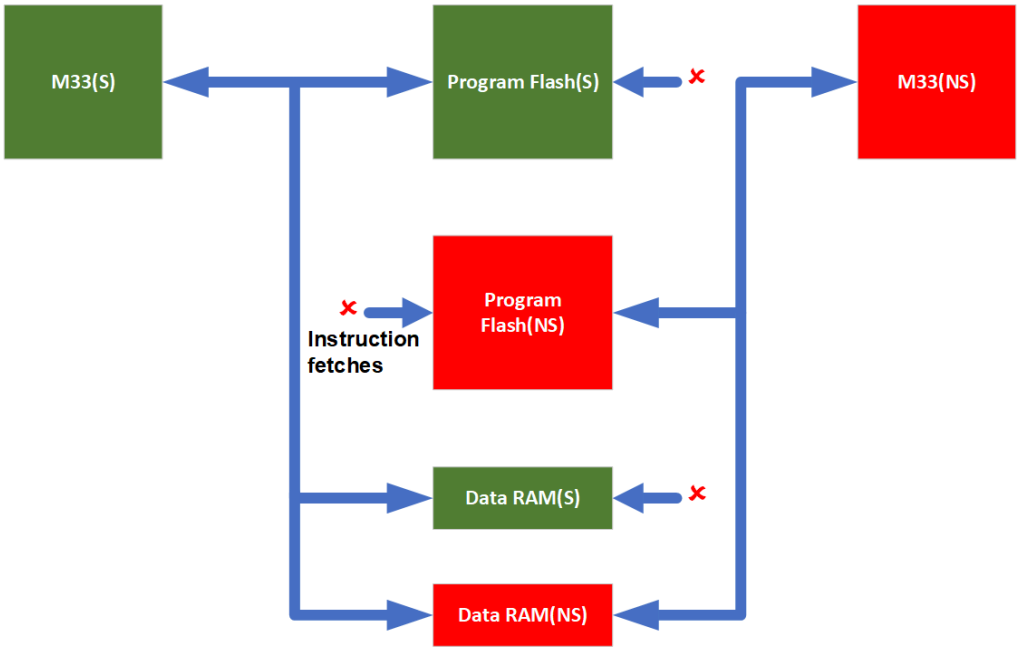

Figure 1. Secure state and Non-secure state view for TrustZone

In summary:

- NS application code “trust” that secure code does not corrupt/modify NS code or data inadvertently or on purpose to create malfunction or hazard.
- S application code does not “trust” NS application code and disallows access from a CPU-NS.

1.1.2 Secure AHB controller

The LPC55Sxx (with TrustZone) implements second layer of protection with secure AHB controller to provide secure trusted execution at system-level.

With secure AHB controller, you can configure security access rules for each peripheral.

By default, CM33 CPU in Secure state (CPU-S) can access the peripherals in both S-state and NS-state. CM33 CPU in Non-secure state (CPU-NS) can only access the peripherals in NS-state, as shown in Figure 2.

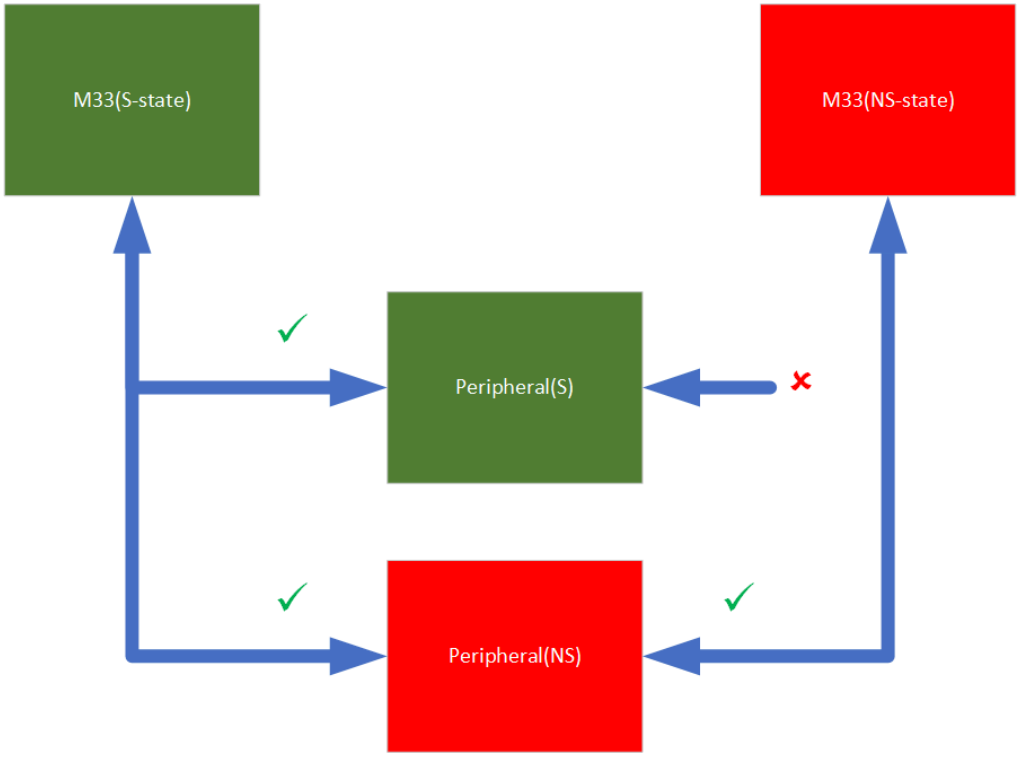

Figure 2. Secure-state and Non-secure state view for TrustZone and secure AHB controller

1.2 Normal GPIO

Normal GPIO is the most common digital peripheral in a microcontroller. Normal GPIO of LPC MCU is very flexible and powerful. Like SPI, UART, and so on, a normal GPIO is also a digital peripheral in the MCU. Figure 3 shows a simple block diagram of the normal GPIO. The normal GPIO can read a pin state regardless of pin function configured. For example, if this pin is configured as UART, then the pin state can be read via normal GPIO read.

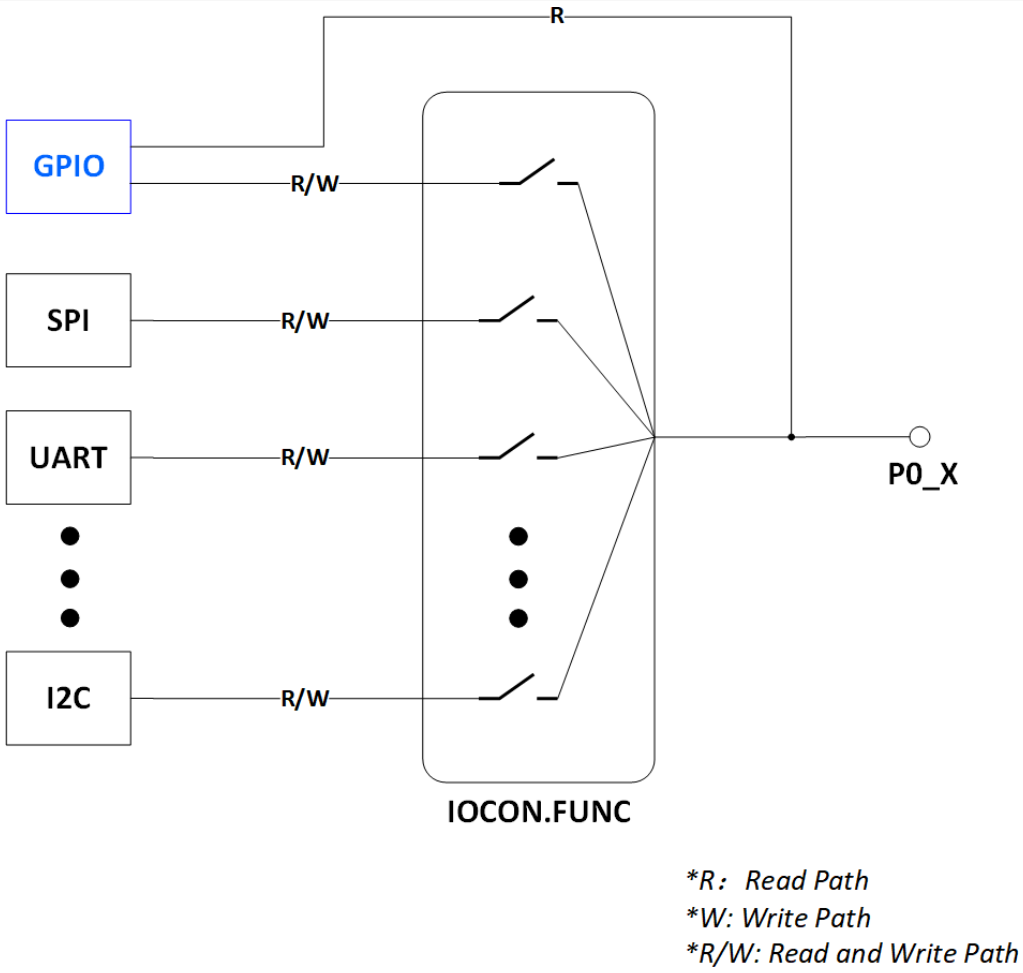

Figure 3. Normal GPIO

2 Secure GPIO, secure GPIO mask, and Secure PINT

Due to the architecture of normal GPIO, all digital I/O pins states are readable through normal GPIO module from the GPIO read path, independent of which function is chosen for this pin as aforementioned. As a result, there is a possibility of leaking information from secure resource(S).

For example, when a UART is configured as a secure peripheral, it means that only the secure-world (such as, code) can access this UART, not the Non-secure world.

However, in this case, non-secure world can monitor the UART pin states through normal GPIO read path, as shown in Figure 3. Therefore, the non-secure world can get all the information of secure UART.

To solve this issue and safeguard incoming data on secure peripherals, secure GPIO Mask is implemented on LPC55Sxx (with TrustZone).

In addition, if secure-world need operate GPIO, it cannot use normal GPIO as normal GPIO is masked. In this case, a new module, named secure GPIO is introduced on LPC55Sxx (with TrustZone). Unlike normal GPIO, this secure GPIO functionality is available only if FUNC=10 in IOCON. It can be used to generate certain input pattern from external device for secure signaling.

For the same reason, secure-world needs secure pin interrupt/Pattern Match Engine (PINT), so another module named secure PINT is implemented.

Figure 4 shows a simple block diagram of the secure GPIO and secure GPIO mask.

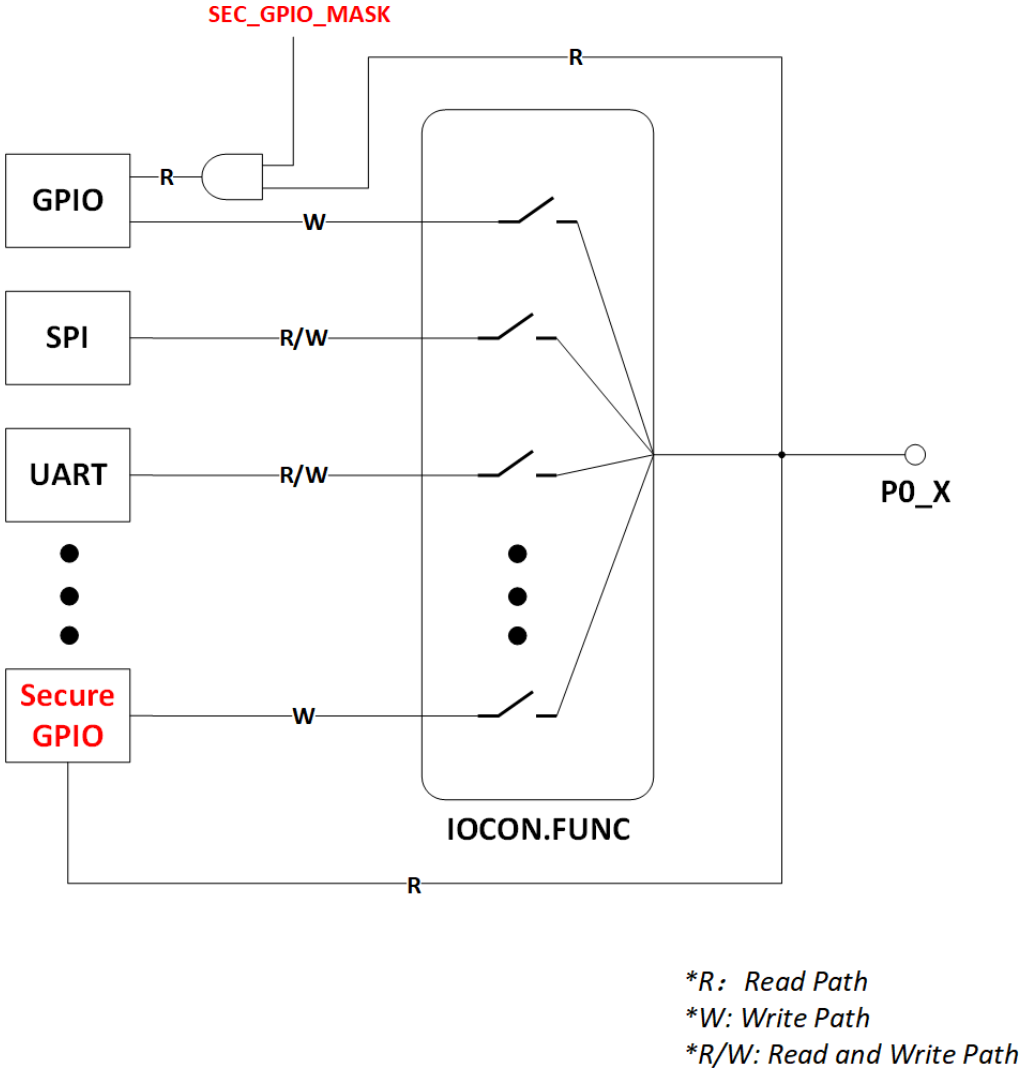

Figure 4. Secure GPIO and secure GPIO mask

2.1 Secure GPIO mask

Each GPIO has a secure GPIO MASK. As shown in Figure 4, we can think of the Secure GPIO Mask as one input of the AND gate. Its default value is **1**. Through Secure GPIO Mask, we can control the on/off state of the normal GPIO read path.

2.2 Secure GPIO

As shown in Figure 4, secure GPIO has the same functions as normal GPIO. However, the access rules to this secure GPIO for different secure levels are configured through the secure AHB controller which can only be accessed in secure state.

2.3 Secure PINT

The main difference between secure PINT and PINT is that the secure PINT only supports up to two pins on Port 0. Similar as secure GPIO, the access rules to this module are configured through the secure AHB controller.

The secure pin interrupt generator and the secure pattern match engine are available on all LPC55Sxx (with TrustZone) devices. Similar as normal PINT, the secure pin interrupt generator, and the secure pattern match engine are mutually exclusive.

2.3.1 Secure pin interrupts

- For secure PINT block, up to two pins can be selected from all pins on port 0, as edge-sensitive or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
- Edge-sensitive interrupt pins can interrupt on rising or falling edges or both.
- Level-sensitive interrupt pins can be HIGH-active or LOW-active.

2.3.2 Secure pattern match engine

- Up to two pins can be selected from all digital pins on port 0 to contribute to a boolean expression. The boolean expression consists of specified levels and/or transitions on various combinations of these pins.
- Each bit slice minterm (product term) comprising the specified boolean expression can generate its own, dedicated interrupt request.
- Any occurrence of a pattern match can be programmed to generate an RXEV notification to the CPU.
- Pattern match can be used with software, to create complex state machines based on pin inputs.

3 Usage

3.1 Use secure GPIO mask to protect secure digital peripherals which need I/O

`SEC_GPIO_MASK` register is used for controlling secure GPIO mask. Default register value is all 1, which means NS code can still read secure peripheral states by reading its pin states, as shown in left side of Figure 5.

To prevent this risk of secure information leakage, to mask the normal GPIO, set the corresponding bits in `SEC_GPIO_MASK` to **0**, as shown in the right side of Figure 5.

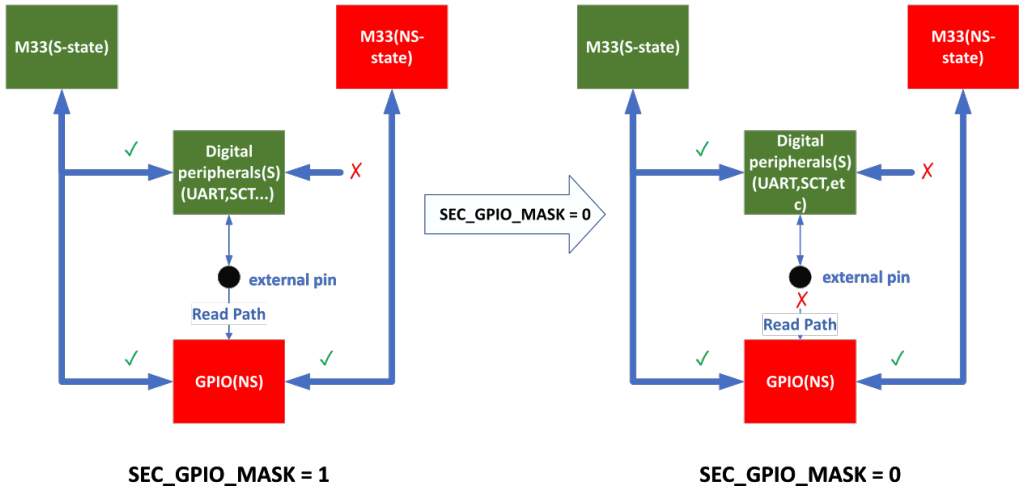

Figure 5. Usage of SEC_GPIO_MASK

Figure 6 shows how to mask P0_5 pin by using secure GPIO MASK:

```
AHB_SECURE_CTRL->SEC_GPIO_MASK0 = AHB_SECURE_CTRL->SEC_GPIO_MASK0 &
                                  ~AHB_SECURE_CTRL_SEC_GPIO_MASK0_PIO0_PIN5_SEC_MASK(0x1U);
```

Figure 6. Set the SEC_GPIO_MASK of P0_5 to 0

3.2 Set one I/O to secure GPIO

Following are the steps to configure an I/O pin to Secure pin:

- Configure the corresponding bit of SEC_GPIO_MASK to **0**.
- Configure the secure GPIO module to **Secure** through secure AHB controller, It prevents non-secure world from accessing the secure GPIO.
- Configure the IOCON block to **Secure** through secure AHB controller. It prevents non-secure world from accessing the IOCON.
- Configure the corresponding pin function to secure GPIO (FUNC=10) through secure IOCON block.
- Enable secure GPIO clock.

Afterward, you can use it like a normal GPIO pin.

The following code snippets take P0_5 pin as an example.

- Configure the SEC_GPIO_MASK of P0_5 to **0**:

```
AHB_SECURE_CTRL->SEC_GPIO_MASK0 = AHB_SECURE_CTRL->SEC_GPIO_MASK0 &
                                  ~AHB_SECURE_CTRL_SEC_GPIO_MASK0_PIO0_PIN5_SEC_MASK(0x1U);
```

Figure 7. Set the SEC_GPIO_MASK of P0_5 to 0

- Make the secure GPIO IP **Secure**:

```
AHB_SECURE_CTRL->SEC_CTRL_AHB2[0].SEC_CTRL_AHB2_0_SLAVE_RULE = (uint32_t)(0x3U);
```

Figure 8. Make the secure GPIO IP Secure

- Make the IOCON block **Secure**:

```
AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE0_MEM_CTRL0 =
        AHB_SECURE_CTRL_SEC_CTRL_APB_BRIDGE_SEC_CTRL_APB_BRIDGE0_MEM_CTRL0_IOCON_RULE(0x3U);
```

Figure 9. Make the IOCON block Secure

- Configure P0_5 pin function to secure GPIO (FUNC=10):

```
const uint32_t port0_pin5_config = (/* Pin is configured as Secure GPIO */
                                    IOCON_PIO_FUNC10 |
                                    /* No addition pin function */
                                    IOCON_PIO_MODE_INACT |
                                    /* Input function is not inverted */
                                    IOCON_PIO_INV_DI |
                                    /* Enables digital function */
                                    IOCON_PIO_DIGITAL_EN |
                                    /* Standard mode, output slew rate control is enabled */
                                    IOCON_PIO_SLEW_STANDARD |
                                    /* Open drain is disabled */
                                    IOCON_PIO_OPENDRAIN_DI);
/* PORT0 PIN30 (coords: A2) is configured as FC0_TXD_SCL_MISO */
IOCON_PinMuxSet(IOCON, 0U, 5U, port0_pin5_config);
```

Figure 10. Configure P0_5 pin function to secure GPIO (FUNC=0xA)

- Enable secure GPIO clock:

```
CLOCK_EnableClock(kCLOCK_Gpio_Sec);
```

Figure 11. Enable secure GPIO clock

3.3 Usage of secure PINT

From application perspective, the method of using secure PINT is same as of normal PINT. There is one thing that needs extra attention:

- To disable the non-secure world from accessing the secure PINT register, set the secure PINT to **Secure** through secure AHB controller.
- Then you can use it like normal PINT and use the same APIs as normal PINT.

The code snippets for above settings are as shown in Figure 12.

- Make the Secure PINT register **Secure**:

```
/* Set Secure PINT register as secure */
AHB_SECURE_CTRL->SEC_CTRL_APB_BRIDGE[0].SEC_CTRL_APB_BRIDGE0_MEM_CTRL0 =
    AHB_SECURE_CTRL_SEC_CTRL_APB_BRIDGE_SEC_CTRL_APB_BRIDGE0_MEM_CTRL0_SEC_PINT_RULE(0x3U);
```

Figure 12. Make the secure PINT register Secure

4 Example

This chapter uses LPC55S69 as an example, and the operation of other device is similar.

4.1 Environment

4.1.1 Hardware environment

- Board
  - LPCXpresso55S69
- Debugger
  - Integrated CMSIS-DAP debugger on the board
- Miscellaneous
  - One micro-USB cable
  - PC
- Board setup
  - Connect the micro-USB cable between PC and P6 link on the board for loading and running a demo.

4.1.2 Software environment

- Tool chain
  - IAR embedded workbench
- Software package
  - AN12326SW.zip

4.2 Steps and result

This example demonstrates how to use secure GPIO. The basic steps are as follows:

1. Configuration
   - Open the `secure_gpio_s` project located in the path, as shown in Figure 13.

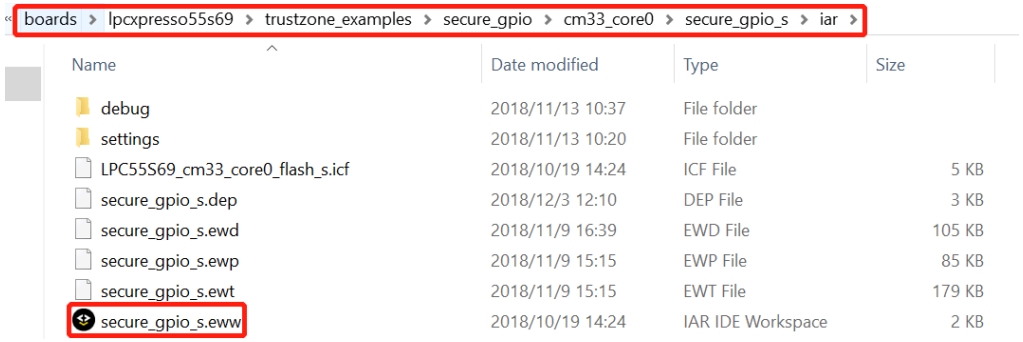

Figure 13. Location of the demo project

There are two projects in the workspace.

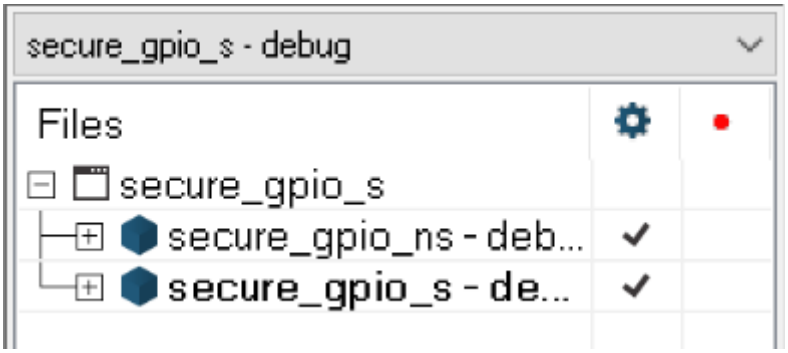

Figure 14. Demo projects

- Configure `secure_gpio_s` and `secure_gpio_ns` projects, as shown in Figure 14.

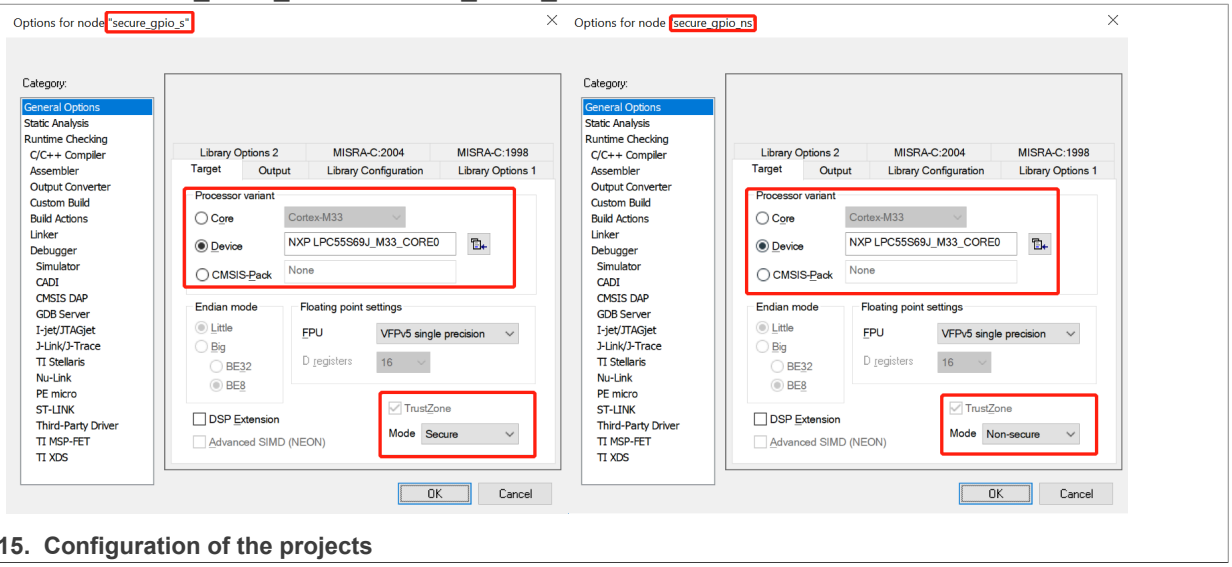

Figure 15. Configuration of the projects

2. Compile and download
   - Compile `secure_gpio_s` project first, then compile `secure_gpio_ns` project.
   - Connect the micro-USB cable between PC and P6 link on the board while pressing and holding down ISP button.
   - Download compiled executable file.
   - Release ISP button after the download is successful.
3. Run
   Reset the board to run by pressing the **Reset** (S4) button on the board.

4. Result
   Two LEDs are used in this example. Blue LED indicates that normal GPIO reads the pin state, whereas green LED indicates that the secure GPIO reads the pin state. After reset, code is running in secure world, and it initializes the system including above two LEDs, and then it jumps to Non-secure world. In non-secure world, it reads `P0_5` pin (ISP button/S1 on EVK) via both normal GPIO and Secure GPIO and the pin state it reads is 1 since this pin is pulled up externally by default. `P0_5` is read as **0** when ISP button is pressed and hold down, If `P0_5` is 0, it turns on the blue LED and green LED as now both normal GPIO and secure GPIO reads 0 from this pin.
   Press USER button (S3), it jumps to secure world, toggle secure GPIO mask, and then jump back to non-secure world. Press the **WAKEUP** button (S2), and it jumps to secure world. Make secure GPIO **Secure**, and it then jump back to non-secure world. At last, it tries to access the secure GPIO from non-secure world, because of secure access violation, it enters **Hard Fault**. Figure 16 depicts its control flow.

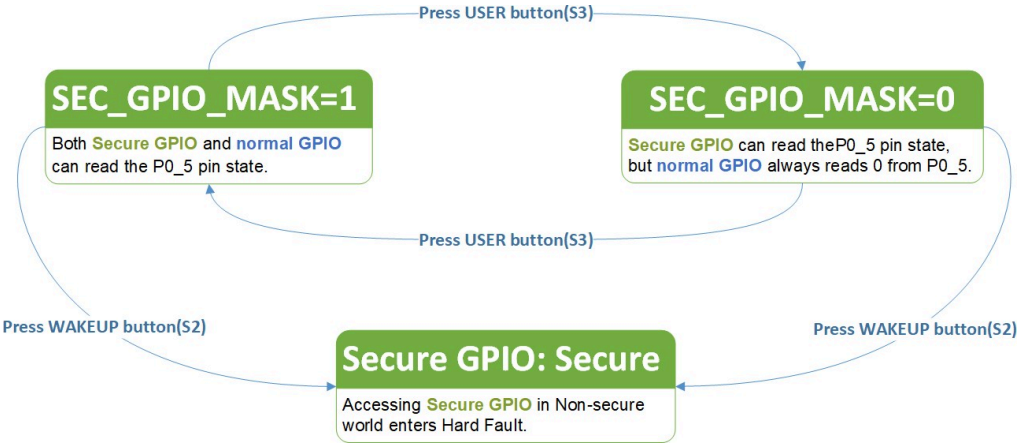

Figure 16. Flow of Secure GPIO example

5 Conclusion

The example shows that non-secure world can access a peripheral pin state regardless of the pin function and whether the peripheral function is secure or non-secure. It results in secure information leakage. To prevent, a secure GPIO must be used and configured and used in secure world. Whereas, the normal GPIO is used in non-secure world. Same rules apply to secure PINT and normal PINT.

6 Note About the Source Code in the Document

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

Copyright 2023 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 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.

7 Revision history

Table 1 summarizes the revisions to this document.

Table 1. Revision history

Revision number	Date	Substantive changes
2	24 July 2023	General updates
1	26 February 2020	General updates
0	15 January 2019	initial version

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.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell 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: 24 July 2023
Document identifier: AN12326