

AN14140

Keep GPIO State in Low Power – i.MX 8ULP

Rev. 1 — 4 December 2023

Application note

Document information

| Information | Content |
|-------------|---|
| Keywords | AN14140, i.MX 8ULP, FSGPIO, RGPIO, GPIO state keep, uPower, Linux, SDK, low power modes, PAD isolation, CM33, A35 |
| Abstract | This application note describes how to keep GPIO state in low power modes including the transition. |



1 Overview

i.MX 8ULP I/O design provides the feature of keeping the GPIO state (output HIGH or LOW) in low-power modes and even during the power mode transition. This makes hardware board design easier when users want to use GPIO to control the external peripherals state. For example, using GPIO to control the Wi-Fi module power supply, keep its power during system suspend to enable Wake On WLAN to wakeup system. In this case, always power on the Wi-Fi module.

This documentation describes how to keep GPIO state in low-power modes including the transition.

2 i.MX 8ULP I/O design

Figure 1 shows the big picture of the i.MX8ULP I/O Design. The IOMUX cell is required whenever two or more functional modes are required for a specific pad. For example, the PTA0 port PAD can be used by General Purpose I/O (GPIO, PTA0), LPUART (LPUART0_CTS_b), or I²C (LPI2C0_SCL). Through IOMUXC PCR register MUX bits, you can configure by which module the PAD is used. The PAD attributes, such as drive strength, slew rate, and pull up/down, are also configurable through PCR registers.

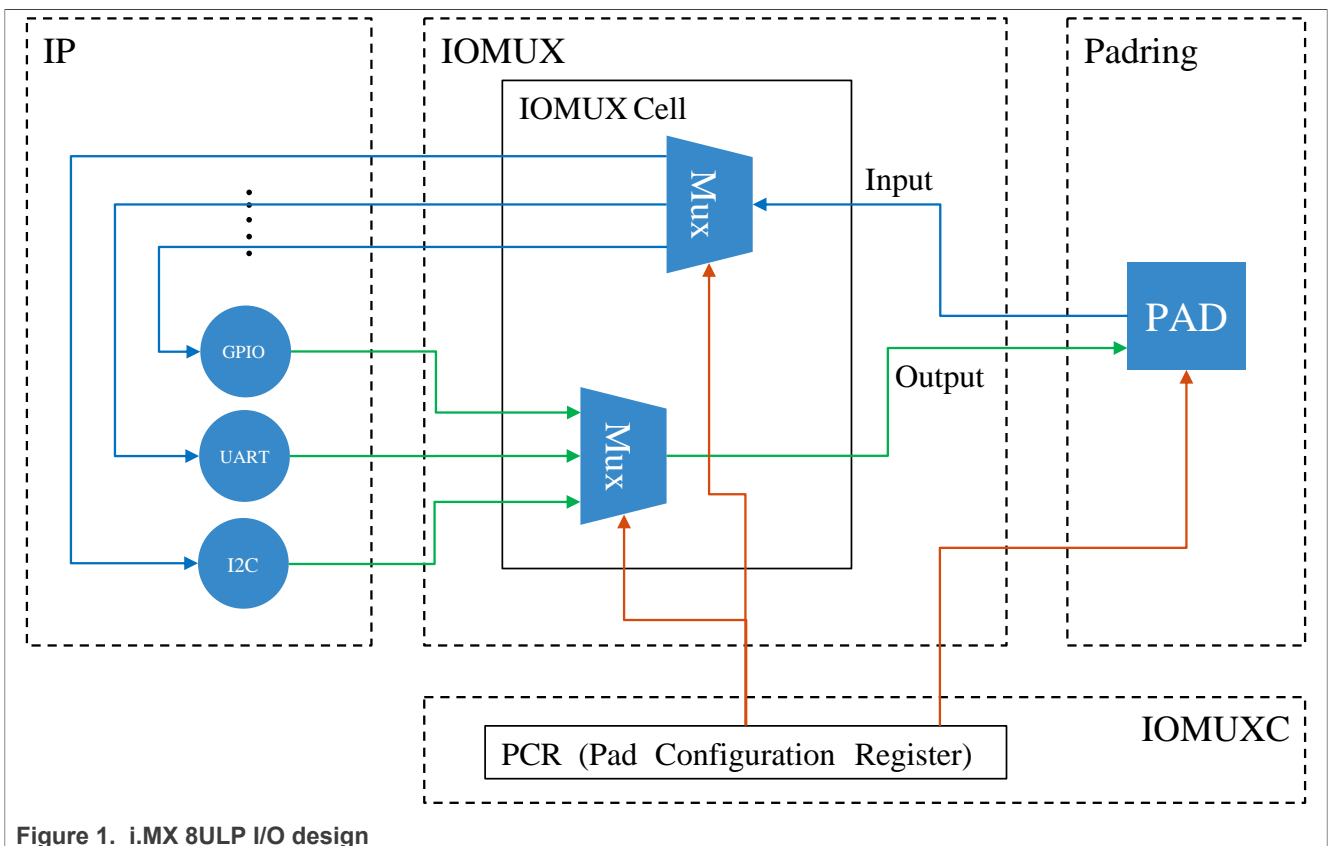


Figure 1. i.MX 8ULP I/O design

2.1 IOMUX design

The IOMUX Controller (IOMUXC), working with the IOMUX, enables the chip to share one pad for multiple signals from different peripheral interfaces. To perform this pad sharing mechanism, multiplex the input and output signals of the pad. Every peripheral signal requires a specific pad setting parameter, such as weak pull-up or drive-strength. The IOMUXC controls the pads setting parameters, digital filter functions, and peripherals multiplexing of the pad. For each pad, there are up to 16 peripherals multiplexing options, called ALT modes. The input signal of a peripheral may come from multiple pads. To avoid an input signal enabled at multiple

locations, the IOMUXC also controls input multiplexing logic. The IOMUX consists of combinatorial logic combined from several basic IOMUX cells. Each basic IOMUX cell handles the multiplexing of one pad signal as defined on IOMUXC registers.

2.2 PAD design

i.MX 8ULP has two pad types:

- Failsafe GPIO (FSGPIO)
 - Operating voltage range: 1.8 V ~3.3V. The range is configurable.
 - Isolation: close - keeps the state of the pad before turning off the domain.
 - Isolation: reset - resets level shift and weak drive.
 - Used by PTA/B/E/F.
- High Speed GPIO (HSGPIO)
 - Operation voltage range: 1.8 V ~3.3V. The range is configurable.
 - Automatic compensation: adjusts drive strength dynamically; freeze stops compensation uses last value (lowest power).
 - When the power supply is off, the pads go to the high Z state. The state is not retained.
 - Used by PTC/D.

In this case, FSGPIO PAD is recommended, because it can do isolation to keep its state retention even when IOMUX and GPIO power gated (IP not functional). It means that for this purpose, use PTA/B (RTD domain) or PTE/F (APD domain).

3 Keep GPIO state in power down

To keep GPIO state in power down modes, enable the FSGPIO PAD isolation function.

The i.MX 8ULP IP modules have their own statuses in different power modules. [Table 1](#) describes the example for IOMUX and GPIO modules in real-time domain.

Table 1. IOMUX and GPIO modules in real-time domain

| Modules | Power modes | Active | Sleep | Deep sleep | Power down |
|--------------|--------------------------|--|---|---|--|
| | Power state power domain | Core supply = ON, Bias = AFBB and DVS, System/Bus clocks = ON, I/O supply = ON | Core Supply = ON, Bias = AFBB or ARBB, Voltage = Fixed, System/Bus clock = ON (optional) IO supply = ON | Core supply = ON, Bias = RBB Voltage/Bias = prog, System/Bus clock = OFF, I/O supply = ON | Core supply = ON (Mem only), Bias = RBB, Voltage/Bias = prog, System/Bus clock = OFF, I/O supply = ON (optional) |
| IOMUX [0] | RTD | Functional | CG or Static (optional) ^[1] | CG | PG |
| RGPIOA, B, C | RTD | Functional | CG (optional) ^[2] | CG | PG |

[1] If the system/bus clock is not available (OFF), the module is clock gated. Otherwise, it remains static.

[2] If the system clock is OFF in the sleep mode, the module is clock gated. Otherwise, it is fully functional.

The RTD IOMUX and GPIO can be Clock Gated (CG) optionally under the sleep mode, CG under the Deep Sleep mode, and Power Gated (PG) under the power-down mode. To keep the external I/O pin state in the sleep or deep sleep mode, keep the IOMUX and GPIO configurations of the pin unchanged, because GPIO state can be kept even when the clock is gated. But under the power down mode, user can only use the PAD isolation feature.

To use PAD isolation to keep GPIO state, a dedicated programming flow is needed. In brief, make sure to enable the PAD isolation before IOMUX/GPIO clock or power gated. And disable the isolation after IOMUX/GPIO registers are restored:

1. Configure IOMUX and GPIO to make the pin output to the required level.
2. Keep and save IOMUX and GPIO configuration for the pin before entering the low-power mode.
3. Enable FSGPIO PAD isolation by uPower before entering the low-power mode.
4. Keep FSGPIO PAD isolation enabled even when exiting from the low-power mode by uPower.
5. Restore IOMUX and GPIO configuration for the pins after exiting from the low-power mode.
6. Disable FSGPIO PAD isolation.

The programming flows for APD and RTD domains are a little bit different. They need two sections to cover. Meanwhile, PTE or PTF is used in APD in this case and PTA or PTB is used in RTD for this case.

3.1 Application Domain (APD)

This section uses the Linux BSP as an example to describe how to configure IOMUX/GPIO/PAD for APD.

3.1.1 Pin configuration and usage

[Table 2](#) describes two ways to configure and use GPIO in Linux.

Table 2. Configure and use GPIO

| | General GPIO device driver | Dedicated device driver |
|-----------|--|--|
| Configure | GPIO <code>pinctrl</code> hogging in device tree | Device driver <code>pinctrl</code> in device tree. |
| Usage | <code>gpiod</code> utility in user space | Handle in driver, like toggling GPIO during driver probe, to reset or power up the device. |

- **General GPIO device driver.**
User can use GPIO hogging in the device tree to configure a PAD pin to GPIO mux and its PAD attributes. GPIO hogging is a mechanism providing automatic GPIO request and configuration as part of the driver probe function of the `gpio-controller`.

```
&iomuxc {
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_hog>;
    pinctrl_hog: hoggrp {
        fsl,pins = <
            MX8ULP_PAD_PTF26__PTF26 0x3 // 0x3 is PAD attributes
        >;
    };
    Pull-up selected
    .....
};
```

After boot, users can use `gpiod` utility to toggle GPIO. Taking PTF26 as an example:
Set the PTF26 output LOW:

```
$gpioset -c 5 26=0
```

Set the PTF26 output HIGH:

```
$gpioset -c 5 26=1
```

- **Dedicated driver**
User can use `pinctrl` nodes in the device tree to configure a PAD pin to GPIO mux and its PAD attributes. Taking SDIO driver (connect external WiFi module) as an example, `mmc-pwrseq-simple` driver takes one

property `reset-gpios` to use this pin as power switch for connected Wi-Fi module power. To initialize this pin setting during device driver probe, assign `pinctrl_usdhc2_ptf` to `pinctrl` of the `usdhc2` device node. And assign PTF26 to `reset-gpios` in `usdhc2_pwrseq` node as below:

```
&iomux1 {
    pinctrl_usdhc2_ptf: usdhc2ptfgrp {
        fsl,pins = <
            MX8ULP_PAD_PTF26__PTF26          0x3
        >;
    };
    ...
};

usdhc2_pwrseq: usdhc2_pwrseq {
    compatible = "mmc-pwrseq-simple";
    reset-gpios = <&gpiof 26 GPIO_ACTIVE_LOW>;
};

&usdhc2 {
    pinctrl-names = "default", "state_100mhz", "state_200mhz", "sleep";
    pinctrl-0 = <&pinctrl_usdhc2_pte>, <&pinctrl_usdhc2_ptf>;
    pinctrl-1 = <&pinctrl_usdhc2_pte>, <&pinctrl_usdhc2_ptf>;
    pinctrl-2 = <&pinctrl_usdhc2_pte>, <&pinctrl_usdhc2_ptf>;
    pinctrl-3 = <&pinctrl_usdhc2_pte>, <&pinctrl_usdhc2_ptf>;
    mmc-pwrseq = <&usdhc2_pwrseq>;
    ...
};
```

3.1.2 Program flow

Arm Trusted Firmware (ATF) deals with the program flow for keeping the GPIO state in low-power modes. To modify the ATF source code:

1. Do not reset dedicated IOMUX/GPIO registers. Take PTF26 as an example (*plat/imx/imx8ulp/apd_context.c*):

```
void apd_io_pad_off(void) {
    .....
    /* off the PTD/E/F, need to be customized based on actual user case
    */
    for (i = 0; i < 3; i++) {
        for (j = 0; j < iomuxc_sections[i].reg_num; j++) {
            mmio_write_32(iomuxc_sections[i].offset + j * 4, 0);
            if (!(i == 2 && j == 26))
                mmio_write_32(iomuxc_sections[i].offset + j *
                    4, 0);
        }
    }
    .....
}
```

2. Configure `PAD_CFG` to **CLOSED** for PTE/F for APD Power Down. Configure (keep) `PAD_CFG` (in `uPower` structure array by `ADMA_PWR_MODE` index) as **CLOSED** after resume to **ACTIVE** (*plat/imx/imx8ulp/imx8ulp_psci.c*).

```
ps_apd_pwr_mode_cfgs_t apd_pwr_mode_cfgs = {
    .....
    /* PD */
    [PD_PWR_MODE] = {
        .swt_board_offs = 0x170,
```

```

        .swt_mem_offs = 0x178,
        .pmic_cfg = PMIC_CFG(0x23, 0x0, 0x2),
        .pad_cfg = PAD_CFG(0xC, 0x0, 0x01e80a00),
        .bias_cfg = BIAS_CFG(0x0, 0x2, 0x2, 0x0),
    },

    [ADMA_PWR_MODE] = {
        .swt_board_offs = 0x120,
        .swt_mem_offs = 0x128,
        .pmic_cfg = PMIC_CFG(0x23, 0x0, 0x2),
        .pad_cfg = PAD_CFG(0xC, 0x0, 0x0deb7a00),
        .bias_cfg = BIAS_CFG(0x2, 0x2, 0x2, 0x0),
    },
.....
};

```

The `pad_close` member in the `pad_cfg` highlight in GREEN color here is to tell uPower to close which FSGPIO PAD: Bit 0: PTA, Bit 1: PTB, Bit 2: PTE, Bit 3: PTF. So, here `0xC` means to enable isolation for PTE and PTF.

- Restore IOMUX/GPIO registers after resume to ACTIVE. By default, ATF handles the restore, so no change is required.
- Clear the bits of `PMC_SYS_CTRL_PAD[PADCLOSE]` to remove isolation of PADS after IOMUX/GPIO register is restored after resume (*plat/imx/imx8ulp/imx8ulp_psci.c*).

```

#define DRAM_LPM_STATUS                U(0x2802b004)
+#define PMC_SYS_CTRL_PAD              U(0x283590BC)
extern void imx8ulp_init_scmi_server(void);
void imx_domain_suspend_finish(const psci_power_state_t *target_state)
    /* restore the ap domain context */
    imx_apd_ctx_restore(cpu);
+    /* restore the PAD status */
+    mmio_write_32(PMC_SYS_CTRL_PAD, 0x0);
.....
}

```

3.2 Real-Time Domain (RTD)

This section uses MCUXpresso SDK as an example to describe how to configure IOMUX/GPIO/PAD for RTD.

3.2.1 Pin configuration and usage

In SDK, to configure IOMUX, refer to `pin_mux.c` in the demo applications. For example:

```

void BOARD_InitXXPins(void) {
    IOMUXC_SetPinMux(IOMUXC_PTA4_PTA4, 0U); // Configure PTA4 MUX to GPIOA pin
    4.
    IOMUXC_SetPinConfig(IOMUXC_PTA4_PTA4, // Configure it's PAD configure to
    Pull-up
    IOMUXC_PCR_PE_MASK | IOMUXC_PCR_PS_MASK);
}

```

GPIO output can be configured in application. For example:

```

rgpio_pin_config_t gpioConfig = {
    kRGPIO_DigitalOutput,
    0U,
};

```

```
RGPIO_PinInit(GPIOA, 4, &gpioConfig); // Configure PTA4: GPIOA pin4 as output
RGPIO_PinWrite(GPIOA, 4, 1); // Output HIGH
```

3.2.2 Program flow

In the SDK (taking SDK v2.14.0 as example), the low power configuration flow is done in the application: take `power_mode_switch` demo as example to keep PTA4 state during the low-power mode:

1. Do not reset dedicated IOMUX/GPIO registers (`power_mode_switch.c`).

```
static void APP_Suspend(void)
{
    .....
    /* Backup PTA IOMUXC and GPIOA ICR registers then disable */
    for (i = 0; i <= 24; i++)
    {
        iomuxBackup[backupIndex] = IOMUXC0->PCR0_IOMUXCARRAY0[i];
        gpioICRBackup[backupIndex] = GPIOA->ICR[i];
        GPIOA->ICR[i] = 0; /* disable interrupts */
        if (i != 4) // PTA4
            IOMUXC0->PCR0_IOMUXCARRAY0[i] = 0;
        backupIndex++;
    }
}
```

2. Configure `PAD_CFG` to **CLOSED** for PTA/B when entering RTD Power Down. Configure (keep) `PAD_CFG` (in `uPower` structure array by `ACT_RTD_PWR_MODE` index) as **CLOSED** after resume to **ACTIVE** (`lpm.c`):

```
static ps_rtd_pwr_mode_cfgs_t rtd_pwr_mode_cfgs = {
    /* PD */
    [PD_RTD_PWR_MODE] =
    {
        .in_reg_cfg      = IN_REG_CFG(0x00000000, 0x00000000),
        .pmic_cfg        = PMIC_CFG(0x00000023, 0x00000000),
        .pad_cfg         = PAD_CFG(0x3, 0x00000000, 0x00000000),
        .mon_cfg         = MON_CFG(0x00000000, 0x0, 0x0),
        .bias_cfg        = BIAS_CFG(0x00010001, 0x0001001a, 0x0001001a,
0x00000001),
        .pwrsys_lpm_cfg = PWRSYS_LPM_CFG(0),
    },
    .....
    /* ACT */
    [ACT_RTD_PWR_MODE] =
    {
        .in_reg_cfg      = IN_REG_CFG(0x0000001c, 0x3),
        .pmic_cfg        = PMIC_CFG(0x23, 0x00000000),
        .pad_cfg         = PAD_CFG(0x3, 0x0, 0x0),
        .mon_cfg         = MON_CFG(0x0, 0x0, 0x0),
        .bias_cfg        = BIAS_CFG(0x00020002, 0x2, 0x2, 0x0),
        .pwrsys_lpm_cfg = PWRSYS_LPM_CFG(0),
    },
};
```

The `pad_close` member in the `pad_cfg` highlight in GREEN color here is to tell `uPower` to close which FSGPIO PAD: Bit 0: PTA, Bit 1: PTB, Bit 2: PTE, Bit 3: PTF. So, here `0x3` means to enable isolation for PTA and PTB.

3. Restore IOMUX/GPIO registers after resumed to **ACTIVE**. By default, it is restored in the `APP_Resume()`: `power_mode_switch.c`.
4. Clear the bits of `PMC_SYS_CTRL_PAD[PADCLOSE]` to remove isolation of PADs.

```
static void APP_Resume(bool resume)
```

```

{
.....
    /* Restore PTC IOMUXC and GPIOC ICR registers */
    for (i = 0; i <= 23; i++)
    {
        IOMUXC0->PCR0_IOMUXCARRAY2[i] = iomuxBackup[backupIndex];
        GPIOC->ICR[i] = gpioICRBackup[backupIndex];
        backupIndex++;
    }
    *(volatile uint32_t *)0x283590BC = 0x0;
.....
}

```

4 Reference

- *MX8ULP Reference Manual*
- *uPower User Guide* (document [UPOWERFWUG](#))
- *i.MX Linux Reference Manual* (document [IMXLXRM](#))
- [SDK demo](#)

5 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.

6 Revision history

[Table 3](#) summarizes the revisions to this document.

Table 3. Revision history

| Revision number | Release date | Description |
|-----------------|------------------|------------------------|
| 1 | 04 December 2023 | Initial public release |

Legal information

Definitions

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

Disclaimers

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

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

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

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

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

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

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

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

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

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

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

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

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

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

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

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

Trademarks

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

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

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, μ Vision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) 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.

i.MX — is a trademark of NXP B.V.

Contents

| | | |
|----------|---|----------|
| 1 | Overview | 2 |
| 2 | i.MX 8ULP I/O design | 2 |
| 2.1 | IOMUX design | 2 |
| 2.2 | PAD design | 3 |
| 3 | Keep GPIO state in power down | 3 |
| 3.1 | Application Domain (APD) | 4 |
| 3.1.1 | Pin configuration and usage | 4 |
| 3.1.2 | Program flow | 5 |
| 3.2 | Real-Time Domain (RTD) | 6 |
| 3.2.1 | Pin configuration and usage | 6 |
| 3.2.2 | Program flow | 7 |
| 4 | Reference | 8 |
| 5 | Note about the source code in the document | 8 |
| 6 | Revision history | 8 |
| | Legal information | 9 |

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.
