# FAQ Application Note for TLE984xQX

# Frequently asked questions and application hints

## Application Note

## About this document

### Scope and purpose

This Application Note is intended to provide helpful suggestions and hints how to set up and handle specific modules and functionalities, which are not subject of the User's Manual or Data Sheet and might be interesting for end users. It is organized in a frequently asked question style and doesn't follow any specific order.

*Note:* *The following information is given as a hint for the implementation of the device only and shall not be regarded as a description or warranty of a certain functionality, condition or quality of the device.*

### Intended audience

This document is intended for Customer and Field Application Engineers to answer frequently asked questions for the embedded Power IC, TLE984xQX device family.

## Table of contents

# 1 Introduction

This Application Note lists topics, which emerged from frequently asked questions of users or from changed user requirements. Each topic is organized in three sections:

- **Topic:**
  - Short description of the issue.
- **Description:**
  - More details about the topic.
- **Implementation hint (optional):**
  - Instruction how to handle this topic.

# 2 Collection of questions and topics

This chapter gives an overview of the collected Questions and Topics.

**Table 1 Table of questions**

# 3 GPIO port map and alternate functions

*Which Pin is connected to which peripheral?*

The TLE984xQX has 18 port pins organized in three parallel ports: Port 0 (P0), Port 1 (P1) and Port 2 (P2). Each port pin has a pair of internal pull-up and pull-down current sources that can be individually enabled or disabled. Either pull-up or pull-down devices can be enabled at a time, for a single port pin. P0 and P1 are bidirectional and can be used as general-purpose input/output (GPIO) or to perform alternate input/output functions for the on-chip peripherals. When configured as an output, open drain mode can be selected. On Port 2 (P2) analog inputs are shared with general purpose input.

## 3.1 Description: GPIO register description

Each port consists of 8-bit control and data registers. The registers are defined in **Table 2**.

**Table 2 Port registers**

| Register Short Name | Register Long Name | Description |
|---|---|---|
| Px_DATA | Port x Data Register | x = {0, 1, 2} |
| Px_DIR | Port x Direction Register | x = {0, 1, 2} |
| Px_OD | Port x Open Drain Control Register | x = {0, 1, 2} |
| Px_PUDSEL | Port x Pull-Up/Pull-Down Select Register | x = {0, 1, 2} |
| Px_PUDEN | Port x Pull-Up/Pull-Down Enable Register | x = {0, 1, 2} |
| Px_ALTSEL0 | Port x Alternate Select Register 0 | x = {0, 1, 2} |
| Px_ALTSEL1 | Port x Alternate Select Register 1 | x = {0, 1, 2} |

## 3.2 Implementation: alternate function configuration example

The ports P0 and P1 can be configured to four different output functions. The default configuration is the GPIO function. The three remaining functions are alternate output functions.

The alternate output function selection is split in two bitfields (e.g. **P1_ALTSEL0** and **P1_ALTSEL1**). **ALTSEL1** contains the most significant bit. **ALTSEL0** contains the least significant bit. The example code below shows how to configure these bitfields to connect UART2 module (TXD, RXD) with the GPIOs (P1.0, P1.1).

```
/* connect UART2 to GPIO */

/* set P1.0 to UART2_TXD: */

PORT->P1_DIR.bit.PP0 = 1u; /* PORT P1.0 output configuration */

PORT->P1_ALTSEL0.bit.PP0 = 1u; /* UART2_TXD alternate function 3 */

PORT->P1_ALTSEL1.bit.PP0 = 1u; /* UART2_TXD alternate function 3*/

/* Set P1.1 to UART2_RXD: */

PORT->P1_DIR.bit.PP1 = 0u; /* PORT P1.1 input configuration */
```

## 3.3 Implementation: port map of alternate functions

Each pin is able to handle multiple purposes. **Figure 1** shows the internal signals mapped to GPIOs. The arrow boxes contain the signal names and indicate the data flow direction.

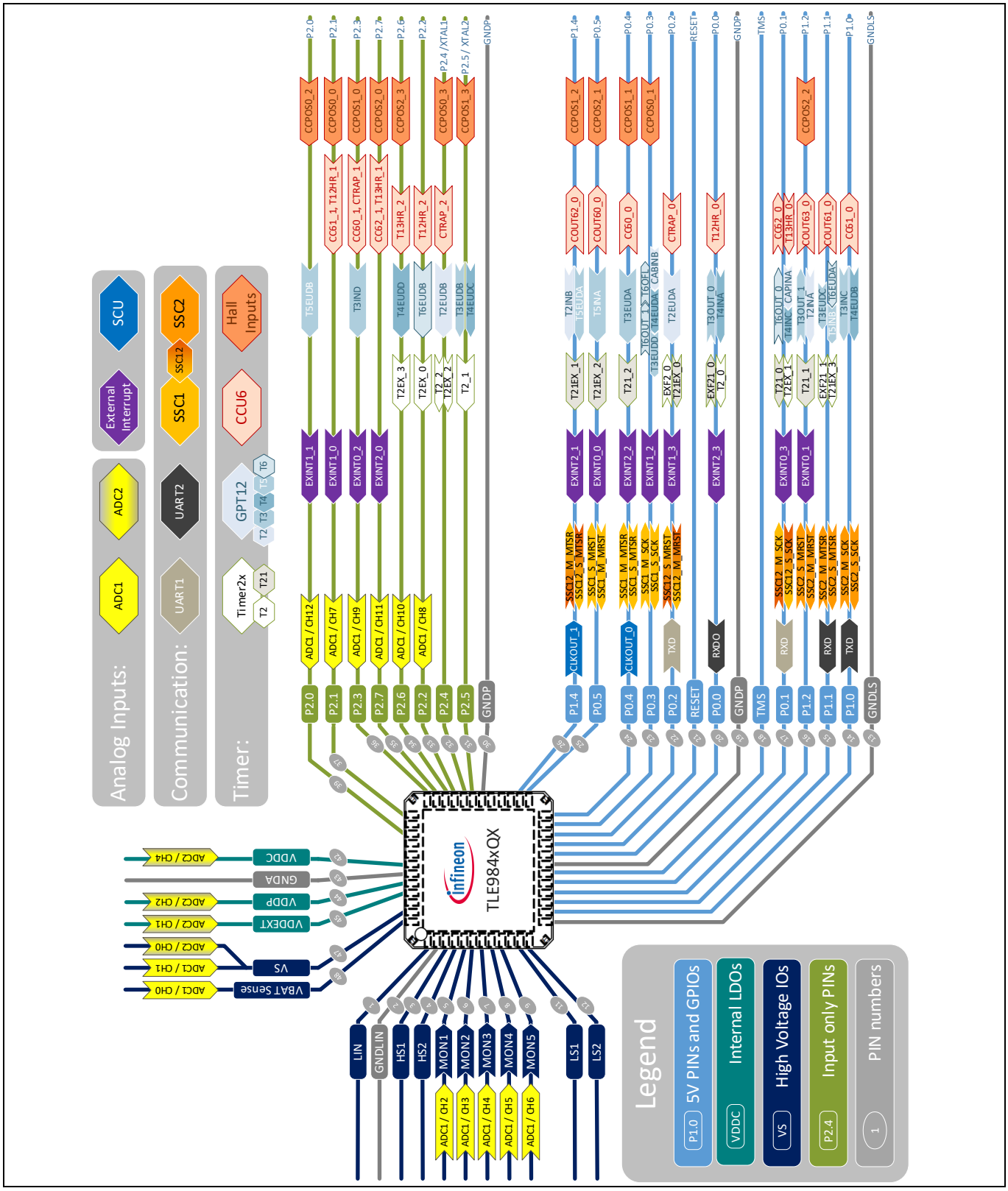


*Figure 1 Port map of alternate function*

# 4 Flash / NVM hints

*What has to be considered when using the NVM?*

## 4.1 Implementation: NVM write-cycles

*What has to be considered during NWM write cycles?*

During NVM-write cycle, the Systick is not running. It is not necessary to explicitly disable it.

It is recommended to trigger a short open window (SOW) before writing the NVM.

## 4.2 Implementation: Address of data flash (emulated EEPROM)

*What are the addresses of the Data Flash (emulated EEPROM)?*

The emulated EEPROM has a variant dependent size as well as a dedicated start address offset

**Table 3** shows the data flash start address offsets and the available size.

**Table 3 Data flash start address**

| Product name | Flash size [kB] | Data flash start address offset |
|---|---|---|
| TLE9842QX | 36 | 0x8000 |
| TLE9842-2QX | 40 | 0x9000 |
| TLE9843QX<br>TLE9845QX | 48 | 0xB000 |
| TLE9843-2QX | 52 | 0xC000 |
| TLE9844QX<br>TLE9844-2QX | 64 | 0xF000 |

## 4.3 Implementation: Content of erased NVM

*What values do erased NVM sectors have?*

When the algorithm reads from an erased or empty NVM page, there are specified values to expect.

The values differ between Code and Data Flash and are listed in **Table 4**.

**Table 4 Data flash start address**

| NVM | Value |
|---|---|
| Data Flash | 0x00 |
| Code Flash | 0xFF |

# 5 Layout and design hints

*Are there any hardware design guidelines that should be followed when designing a board with the TLE984x?*

There are several things that have to be considered when designing a board with the TLE984x.

## 5.1 Description: VDDC and VDDP regulator routing

VDDC is the core voltage regulator. If the voltage is influenced, for example by GND spikes, a hard fault may happen. It is mandatory to pay special attention to the layout of the VDDC capacitors.

VDDP is the peripheral voltage regulator. The VDDC and VDDP GND connections should be separated and not connected to the global ground with a shared via. This prevents the coupling of high-frequency noise coming from VDDC into VDDP and therefore into the GPIOs.

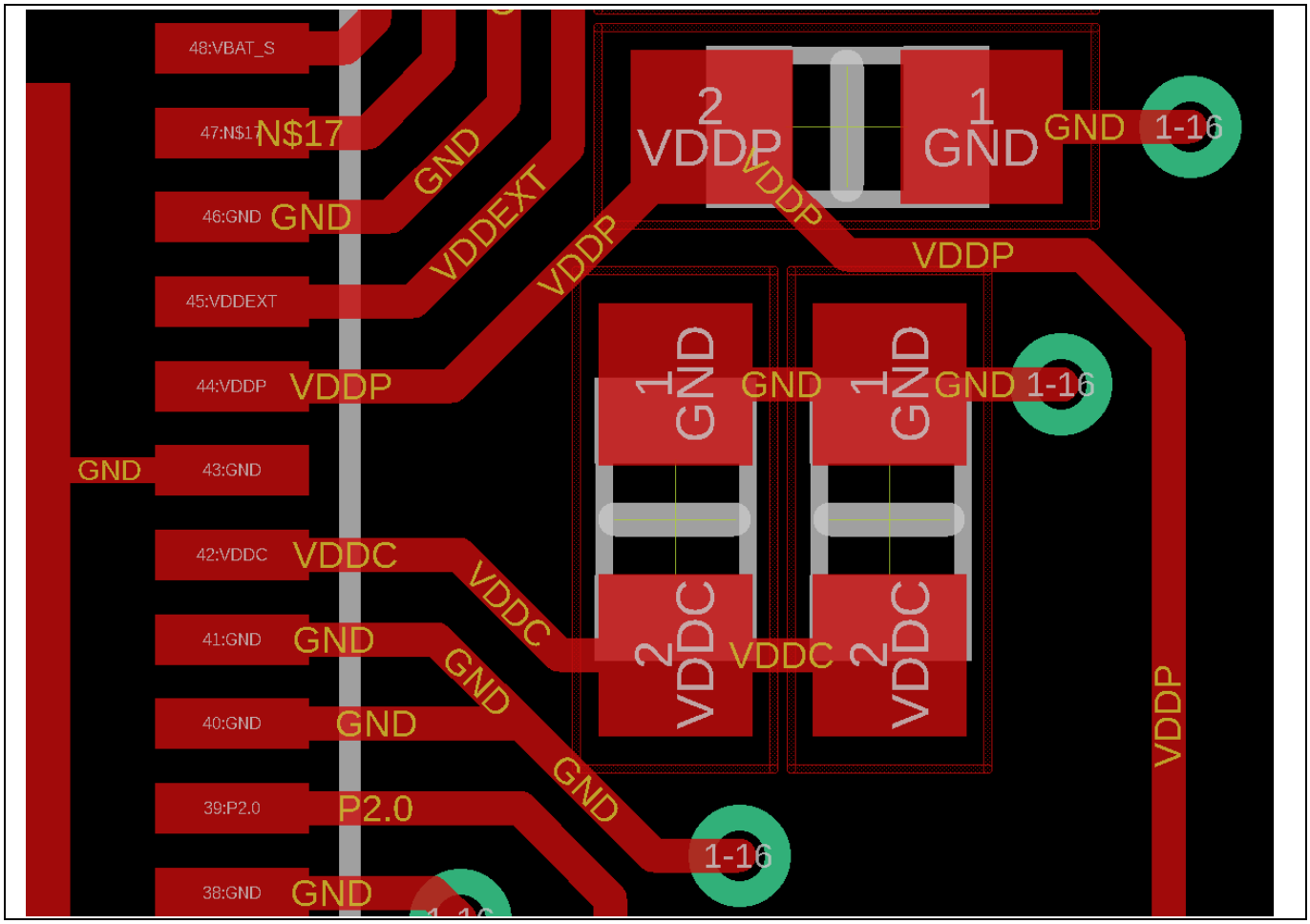


*Figure 2 Layout proposal: VDDC and VDDP capacitors*

**Figure 2** shows a PCB layout proposal for VDDC and VDDP capacitors. It is recommended to place the VDDC capacitors as close as possible to pin 42 and connect them to the global ground with a via. The VDDP capacitor shall also be placed close to pin 44 and connected to the global ground with a separate via.

## 5.2 Description: EMC improvement of Low Side Switch

It is recommended to place a 2.2 nF capacitor at the Low Side Switches (LS1/LS2).

## 5.3 Description: GPIO input filters for hall sensors

In case of long GPIO wiring for external sensors, it is recommended to place a filter to improve HF immunity.

**Figure 3** shows a simplified filter circuitry. For open-drain sensor outputs, it is recommended to place an external pull-up resistor close to the sensor instead of using the internal pull-up sources.

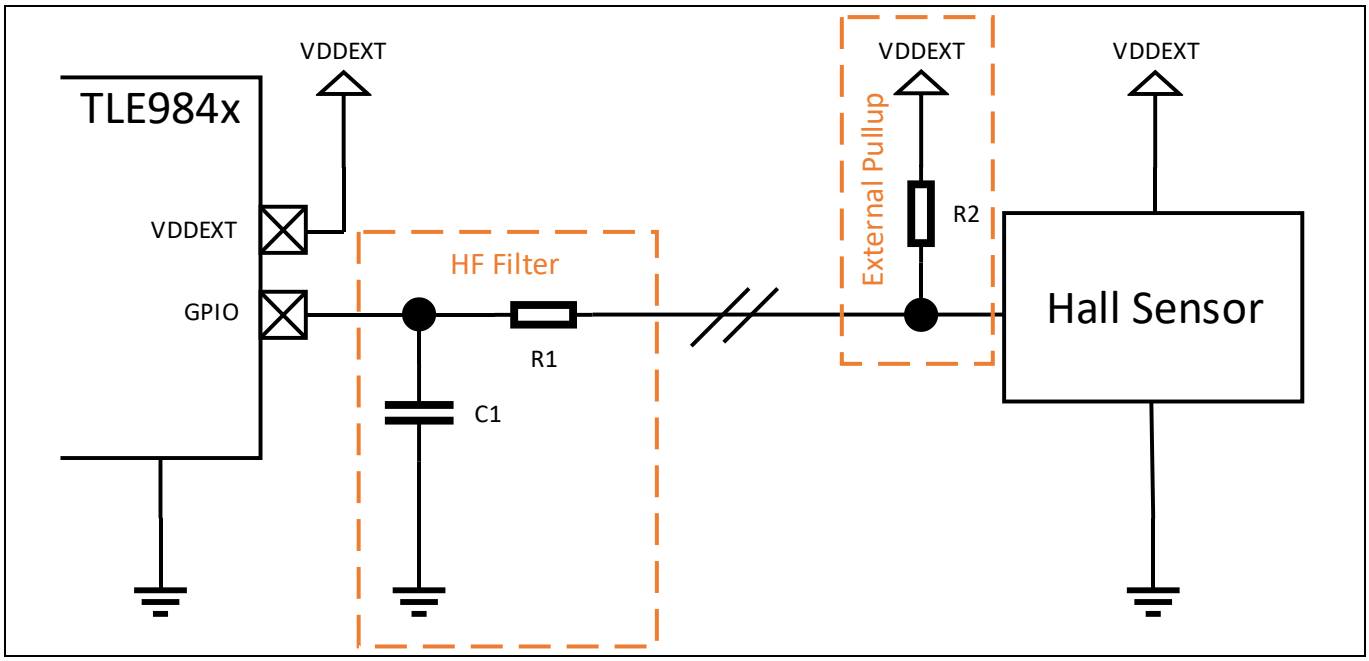


*Figure 3 hall sensor filter*

The additional RC-filter protects the GPIO from any damage based on HF coupling. Example filter values are listed in **Table 5**.

**Table 5 Filter values**

| Part | Value |
|---|---|
| R1 | 1 kOhm |
| C1 | 180 pF |

## 5.4 Description: Ground concept pin 19, 30 and 43

The GND pins 19, 30 and 43 shall directly be connected to the exposed pad, in order to avoid ground shift caused by noise coupling.

## 5.5 Description: Ground concept LIN GND

The LIN GND pin 2 should be connected to the global GND with a dedicated via as close as possible to the device to avoid high-frequency coupling from the LIN connection that could also be present at LIN GND.

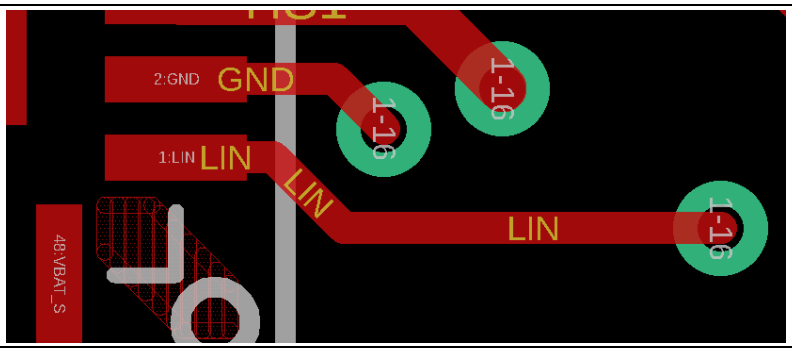


*Figure 4 LIN GND concept*

## 5.6 Description: Ground concept LS GND

The LS GND pin 10 shall be connected to the global GND with a dedicated via as close as possible to the device, in order to avoid high-frequency coupling from the relay (LS1/2) connection that could also be present at LS GND.

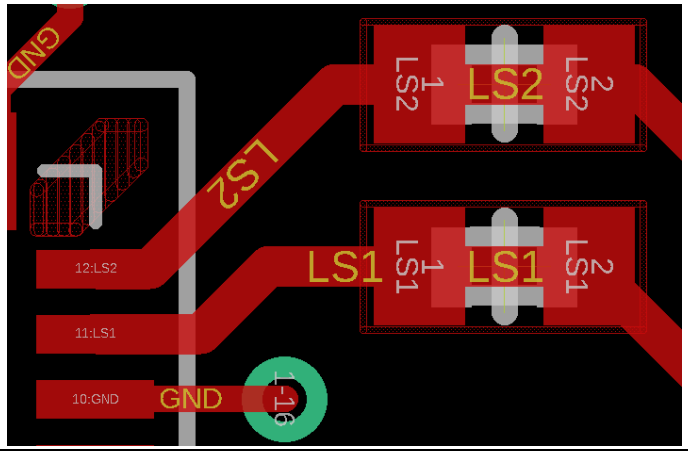


*Figure 5 LS GND concept*

# 6 General Questions

This chapter covers a list of smaller FAQs.

## 6.1 Description: NMI/Interrupt

*How is an NMI enabled?*

NMIs have to be enabled separately via the NMICON register

The default state is ”off”.

## 6.2 Description: Hard fault

*What can cause a hard fault?*

Writing to a protected register will cause a hard fault. Also writing to or reading from an address which has no register assigned to it will also cause a hard fault. A hard fault will call the HardFault_Handler. The HardFault_Handler will be called again and again as long as the source of the hard fault is still present. The user can enable the hard fault interrupt to handle the hard fault.

## 6.3 Description: High side switches HS1/HS2

*What has to be considered for single high side switch devices?*

TLE984x family include devices with one or two high side switches. The software handling has to ensure that the second high-side switch stays disabled for single high side switch devices.

## 6.4 Description: Debug entry behavior

*What has to be considered during debug entry?*

When the Chip enters debug mode, the core will not stop immediately. The user code will run for 85 ms before the debug interface stops the core. If the user code contains data-flash access during this timeframe, the debug connection cannot be established.

# 7 Boot up configuration

*Why does the device not start-up after a reset?*

Using a new device can cause issues, since the device is not executing user code. This chapter explains how to configure the chip to boot up and enter user code as expected.

## 7.1 Description: BSL connection window

After the reset pin is released and the initialization of the device is finished, the BSL connection window starts. The duration of this timing window is defined by the None-Activity-Counter (**NAC**). In this time period, the device can be programmed via LIN. After the NAC expires, the device will enter the user mode as shown in **Figure 6**.The following chapter provides more information about the NAC value.

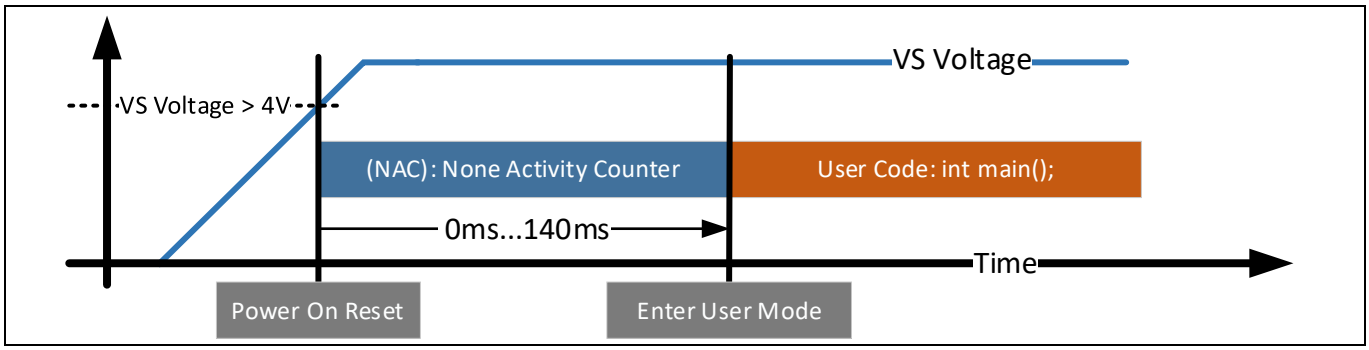


*Figure 6 Start-up procedure*

### 7.1.1 Description: None-Activity-Counter (NAC)

The NAC timer is started during start-up before the BSL mode starts. Once the NAC timer has expired; the boot up process will be continue and the control of the device will be given to the user application. The NAC value is defined by the user. Usually the NAC value is part of the user application that was downloaded before.

The NAC value is stored inside one byte. Only six bits of the NAC byte are defining the timeout value of the NAC. In case of an invalid value, i.e. for an erased flash where, the **NAC timer never expires**, so that the device will stay and wait in BSL mode. This behavior is especially useful for unprogrammed devices, where no user application has been downloaded to the device yet. Here the firmware will not branch into user mode, but instead it will stay in BSL mode and keep waiting for any BSL communication to download any user application into the flash.

The NAC value is stored at the following address:

**Table 6 Address of the NAC value**

| Address | Value |
|---|---|
| 10FFFFFC | NAC |

**Table 7 NAC bit meaning**

| NAC bit | Value |
|---|---|
| 5…0 | NAC expire value n, n * 5ms, 0ms…140ms<br>0x00 BSL mode is skipped, BSL mode deactivated<br>0x20…0x3F: NAC never expires, device stays in BSL Mode, BSL mode is active<br>0x01…0x1C: NAC timeouts between 5ms…140ms, BSL mode is active |
| 6 | Reserved |
| 7 | 0b0: BSL interface selection Fast-LIN<br>0b1: BSL interface selection UART |

An unprogrammed or completely erased device always selects Fast-LIN as BSL interface and stays in BSL mode during start-up.

## 7.2 Description: Node Address - NAD

The NAD is used for the BSL communication to select an individual node. The NAD is an 8 bit value, where only the values 0x80 to 0xFF are valid, 0x00 to 0x7F are invalid values. The value 0xFF acts as a broadcast, this means all devices connected to the LIN line are addressed no matter which NAD value is programmed inside the device. The broadcast can be used to establish a BSL connection to devices where the programmed NAD value is unknown.

The NAD value is stored inside one byte at the following address:

**Table 8 Addresses of NAD value**

| Address | Value |
|---|---|
| 10FFFFFE | NAD |

**Table 9 NAD bit meaning**

| NAC bit | Value |
|---|---|
| 0x00...0x7F | Invalid NAD value |
| 0x80…0xFE | Valid NAD value |
| 0xFF | Broadcast NAD, all devices will react to this value |

# 8 Bit error mechanisms in RAM/NVM

*How does the device react to single/double bit errors in the RAM or NVM?*

The device does support an Error Correction Code (ECC) to detect single bit and double bit errors and dynamic correction of single bit errors in a data word.

*What happens when more than two bits are faulty within one data word?*

If more than two errors are clustered within one data block, the ECC is exhausted and falsely (or wrongly corrected) data might be read without any indication.

## 8.1 Description: Error Correction Code in RAM

The RAM module supports single bit error correction and single/double bit error detection for each 32 bit data-word. When an ECC error occurs, the corresponding status flag in the register SCU_EDCSTAT will be set. A double bit error can trigger a non-maskable interrupt if configured in the SCU_EDCCON register.

The 7 ECC bits for every block are automatically generated when the page in the RAM is written. During every read access the 7 ECC bits are read together with the data-word bits. The validity of the 32 bit data-word + 7 bit ECC is checked by hardware

## 8.2 Description: Error Correction Code in the flash

The flash has an independent ECC error correction/detection mechanisms for the mapblock and the datablock of one page and for the mapRAM. The assembly buffer represents the RAM that holds the mapblock and data blocks for writing one page.

### 8.2.1 Description: Error Correction Code in datablock of flash

The NVM module supports single bit error correction and single/double bit error detection for each 64 bit data-word in the datablock. When an ECC error occurs, the corresponding status flag in the register SCU_EDCSTAT will be set. A double bit error can trigger a non-maskable interrupt if configured in the SCU_EDCCON register.

The 8 ECC bits for every data-word are automatically generated when the assembly buffer is written. During every read access the 8 ECC bits are read together with the data-word bits. The validity of the 64 bit data-word + 8 bit ECC is checked by hardware.

### 8.2.2 Description: Error Correction Code in mapblock of flash

The mapblock of a flash-page consist of a 14 bit data-word that hold the necessary information to map a physical page to a logical page.

The 6 ECC bits for the mapblock are automatically generated when the assembly buffer is written. During every read access the 6 ECC bits are read together with the data-word bits. The validity of the 14 bit data-word + 6 bit ECC is checked by hardware.

### 8.2.3 Description: Error Correction Code in mapRAM of flash

The mapRAM of the last flash-sector has 6bits that hold the look-up table for address translation of the EEPROM emulation.

The 5 ECC bits for the mapRAM are automatically generated when the assembly buffer is written. During every read access the 5 ECC bits are read together with the data block bits. The validity of the 6 bit data-word + 5 bit ECC is checked by hardware.

## 8.3 Implementation: How are bit errors detected and corrected?

When the data-word and the ECC bits are read from a page, the resulting codeword vector is multiplied with a specific hemming generator matrix (done in hardware). Depending on the resulting vector **s,** different scenarios can be detected:

**Table 10 ECC calculation results**

| Resulting vector s: | Scenarios |
|---|---|
| \|s\|=0 | No failure has occurred |
| \|s\|≠ 0 **and** s == column content of generator matrix | Single bit error has occurred at the position of the matching generator matrix column<br>The erroneous bit is flipped and thereby corrected |
| \|s\|≠ 0 **and** s ≠ column content of generator matrix | Double bit error has occurred |

# 9 Register access hints

*What has to be considered when reading and writing into/from a MI_CLK clocked register?*

Some registers are clocked by the measurement interface clock ($f_{MI\_CLK}$). Since the MI_CLK is usually running at a lower rate than the core clock, registers are not always written immediately.

## 9.1 Description: Register write

A register write is requested by the core, clocked with $f_{CCLK}$. The internal write command is latched with the rising edge of $f_{CCLK}$. The command is received and performed by the target register with the next rising edge of the register clock. The register acknowledges the write to the core and the internal write command is cleared with the next rising edge of the core clock.

If the register is clocked by the MI_CLK, the delay between the latching of the internal write command and the actual write can be either long as one full $f_{MI\_CLK}$ period, as shown in **Figure 7**, or as short as a full $f_{CCLK}$ period, as shown in **Figure 8**.

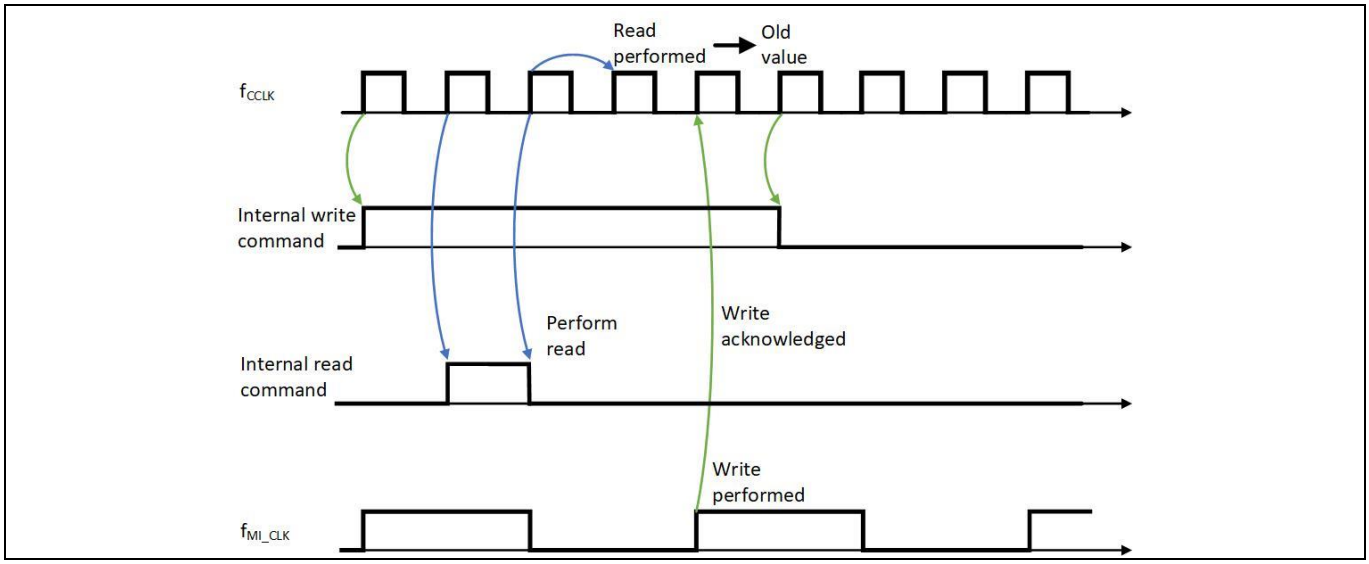


*Figure 7 Long write time and false read*

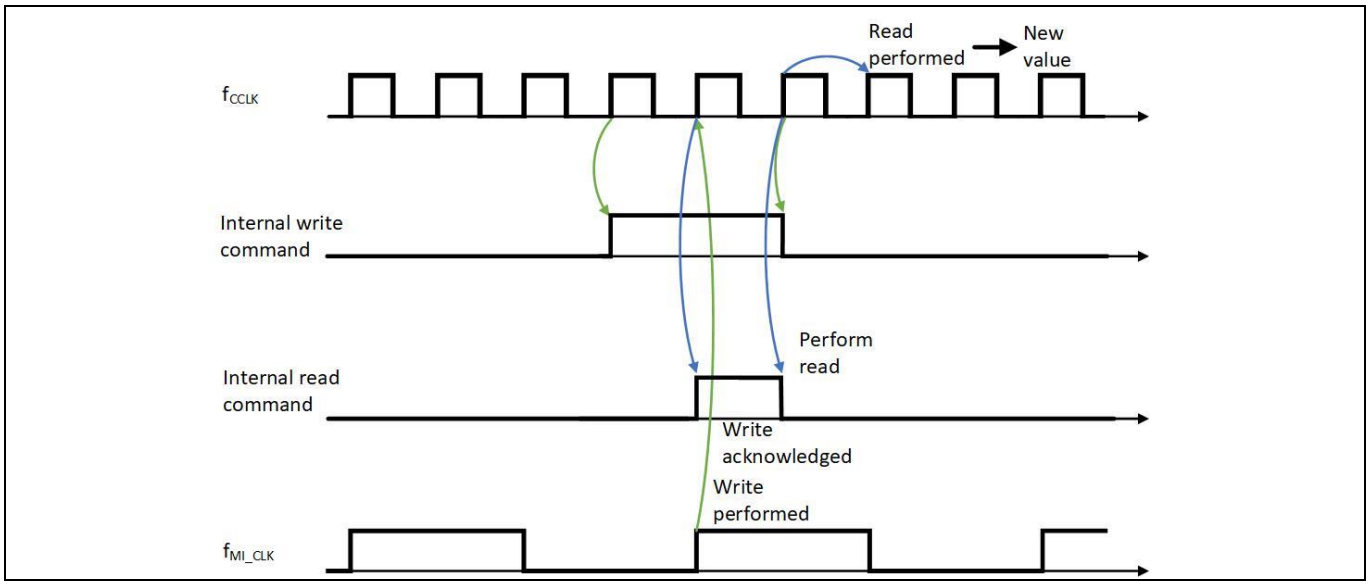


*Figure 8 Short write time and correct read*

## 9.2 Description: Register read

A register read is requested by the core which is clocked by $f_{CCLK}$. The internal read command is latched with the rising edge of $f_{CCLK}$ and cleared again with the next rising edge. The read command is performed by the core when the internal read signal is cleared. The read register content is available with the next rising edge of the $f_{CCLK}$. The read scheme is shown in **Figure 7** and **Figure 8**.

## 9.3 Description: Read timing

**Figure 7** shows, what can happen if a read command is requested one $f_{CCLK}$ tick after the write command is requested. The read request will read back the old value, since the write command is still waiting for the rising edge of the MI_CLK. **Figure 8** shows an example where the same timing between write and read will read back the new value.

Because of this uncertainty, a minimum of one MI_CLK tick must elapse between sending a write command and sending a read command to a register clocked by the MI_CLK.

## 9.4 Implementation: Set correct wait time between read/write

There are serval software use cases to avoid read-back of yet unwritten values.

**Use case 1:** Some C-code instructions are placed between write and read command:

```
/* clearing of VDDEXT supply undervoltage interrupt status flag */
PMU->PMU_VDDEXT_CTRL.bit.VDDEXT_UV_ISC = 1;
/* some other code is placed here */
...
/* reading of VDDEXT supply undervoltage interrupt status flag */
if( PMU-> PMU_VDDEXT_CTRL.bit.VDDEXT_UV_STS == 1 )
{
        /* some user code here */
}
```

In this example, between the interrupt clear write and the corresponding status read, sufficient time is ensured, when at least one C-code instruction is placed in between.

**Use case 2:** Performe an extra write attempt to the interrupt clear register:

```
/* clearing of VDDEXT supply undervoltage interrupt status flag */
PMU->PMU_VDDEXT_CTRL.bit.VDDEXT_UV_ISC = 1;
 PMU->PMU_VDDEXT_CTRL.bit.VDDEXT_UV_ISC = 0;
...
/* reading of VDDEXT supply undervoltage interrupt status flag */
if( PMU-> PMU_VDDEXT_CTRL.bit.VDDEXT_UV_STS == 1 )
{
        /* some user code here */
}
```

In general, any C-code instruction is fine, but it is recommended to use **case 2**, since it will not be eliminated by the compiler optimization and writing a ‘0’ to the clear field will have no functional effect to the register.

## Table of contents

Revision history

| Document version | Date of release | Description of changes |
|---|---|---|
| 1.0 | 2016-09-29 | Release state |
| 1.1 | 2018-08-30 | New Revision rev1.1<br>Changed Document Title<br>Updated: **Figure 1**<br>Added Topics:<br>**Flash / NVM hints**<br>**NMI/Interrupt hints**<br>**Hard Fault hint**<br>**High Side Switches HS1/HS2 hint**<br>**Debug Entry Behavior**<br>**Layout and Design Hints** |
| 2.0 | 2021-06-14 | Collection of all questions added: **chapter 2.0**<br>Updated: **Figure 1**<br>Updated topics:<br>**GPIO port map and alternate functions**<br>**Flash / NVM hints**<br>**Layout and design hints**<br>**NMI/Interrupt hints**<br>**Hard fault hint**<br>**High side switches HS1/HS2 hint**<br>**Debug entry behavior**<br>Added Topics:<br>**Bootup configuration**<br>**Bit error mechanisms in RAM/NVM**<br>**Register access hints** |

**Trademarks**
All referenced product or service names and trademarks are the property of their respective owners.

**Edition 2021-06-14**

**Published by**

**Infineon Technologies AG**

**81726 Munich, Germany**



**Do you have a question about this document?**

**Email:** erratum@infineon.com

**Document reference**
**Z8F55920139**

**IMPORTANT NOTICE**

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology delivery terms and conditions and prices pleas contact your nearest Infineon Technologies offic (**www.infineon.com**).

**WARNINGS**

Due to technical requirements products may contai dangerous substances. For information on the type in question please contact your nearest Infineo Technologies office.

Except as otherwise explicitly approved by Infineo Technologies in a written document signed b authorized representatives of Infineo Technologies, Infineon Technologies' products ma not be used in any applications where a failure of th product or any consequences of the use thereof ca reasonably be expected to result in personal injury.