*Errata*

# *MSP432P401R Microcontroller*

## ABSTRACT

This document describes the known exceptions to the functional specifications (advisories).

## Table of Contents

## 1 Functional Advisories

Advisories that affect the device's operation, function, or parametrics.

✓ The check mark indicates that the issue is present in the specified revision.

| Errata Number | Rev D | Rev C |
|---|---|---|
| BSL16 | | ✓ |
| COMP11 | ✓ | ✓ |
| PORT31 | ✓ | ✓ |
| PORT32 | ✓ | ✓ |
| PORT34 | ✓ | ✓ |
| PSS4 | ✓ | ✓ |
| RTC14 | ✓ | ✓ |
| SYS26 | | ✓ |
| TA23 | ✓ | ✓ |
| USCI42 | ✓ | ✓ |
| USCI43 | ✓ | ✓ |
| USCI44 | ✓ | ✓ |
| USCI45 | ✓ | ✓ |
| USCI46 | ✓ | ✓ |
| USCI47 | ✓ | ✓ |
| USCI48 | | ✓ |
| USCI50 | ✓ | ✓ |
| USCI51 | ✓ | ✓ |

## 2 Preprogrammed Software Advisories

Advisories that affect factory-programmed software.

✓ The check mark indicates that the issue is present in the specified revision.

| Errata Number | Rev D | Rev C |
|---|---|---|
| UDMA2 | ✓ | ✓ |

## 3 Debug Only Advisories

Advisories that affect only debug operation.

✓ The check mark indicates that the issue is present in the specified revision.

The device does not have any errata for this category.

## 4 Fixed by Compiler Advisories

Advisories that are resolved by compiler workaround. Refer to each advisory for the IDE and compiler versions with a workaround.

✓ The check mark indicates that the issue is present in the specified revision.

The device does not have any errata for this category.

Refer to the following MSP430 compiler documentation for more details about the CPU bugs workarounds.

**TI MSP430 Compiler Tools (Code Composer Studio IDE)**

- MSP430 Optimizing C/C++ Compiler: Check the --silicon_errata option
- MSP430 Assembly Language Tools

**MSP430 GNU Compiler (MSP430-GCC)**

- MSP430 GCC Options: Check -msilicon-errata= and -msilicon-errata-warn= options
- MSP430 GCC User's Guide

**IAR Embedded Workbench**

- IAR workarounds for msp430 hardware issues

# 5 Nomenclature, Package Symbolization, and Revision Identification

The revision of the device can be identified by the revision letter on the Package Markings or by the HW_ID located inside the TLV structure of the device.

## 5.1 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all MSP MCU devices. Each MSP MCU commercial family member has one of two prefixes: MSP or XMS. These prefixes represent evolutionary stages of product development from engineering prototypes (XMS) through fully qualified production devices (MSP).

**XMS** – Experimental device that is not necessarily representative of the final device's electrical specifications

**MSP** – Fully qualified production device

Support tool naming prefixes:

**X**: Development-support product that has not yet completed Texas Instruments internal qualification testing.

**null**: Fully-qualified development-support product.

XMS devices and X development-support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

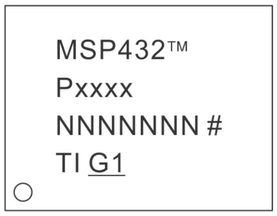
MSP devices have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (XMS) have a greater failure rate than the standard production devices. TI recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the temperature range, package type, and distribution format.

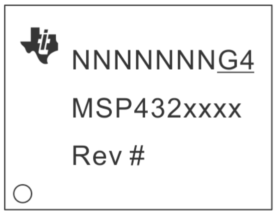
## 5.2 Package Markings

**ZXH80** ***NFBGA, 80 Pin***



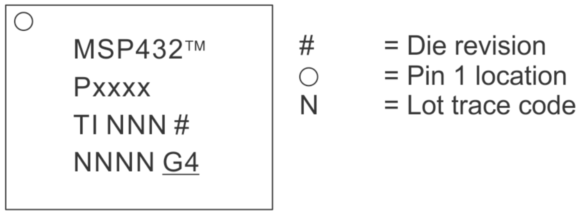
| | |
|---|---|
| # | = Die revision |
| ○ | = Pin 1 location |
| N | = Lot trace code |

**RGC64** ***QFN (RGC), 64 pin***



| | |
|---|---|
| # | = Die revision |
| ○ | = Pin 1 location |
| N | = Lot trace code |

**PZ100** ***LQFP (PZ) 100 Pin***



## 5.3 Memory-Mapped Hardware Revision (TLV Structure)

This device does not support reading the hardware revision from memory.

Further guidance on how to locate the TLV structure and read out the HW_ID can be found in the device User's Guide.

# 6 Advisory Descriptions

| | |
|---|---|
| **BSL16** | ***BSL Module*** |
| **Category** | Functional |
| **Function** | On blank devices, debugger access over JTAG is disabled after the timeout window |
| **Description** | On blank devices, the default device initialization sequence that resides in the boot strap loader (BSL) disables the JTAG pins after a 10 second timeout window if no activity has occurred. This causes debug and program access via JTAG to be disabled. JTAG access can be re-enabled via a device reset (either Class 0 or Class 1). |
| **Workaround** | 1. Use Serial Wire Debug (SWD) as the debug/program interface instead of JTAG.<br><br>OR<br><br>2. If timeout occurs, reset device (either Class 0 or Class 1) and ensure JTAG is accessed within a 10 second window |

| | |
|---|---|
| **COMP11** | ***COMP Module*** |
| **Category** | Functional |
| **Function** | Polling comparator interrupts may result in a missed interrupt |
| **Description** | Polling the comparator output interrupt and the output inverted interrupt flags (CEIFG.CExINT and CEIIFG.CExINT) may result in a missed interrupt if the flags are modified while being read. |
| **Workaround** | Using an interrupt service routine to service CEIFG and CEIIFG interrupts significantly reduces the probability of the issue occurring. |

| | |
|---|---|
| **PORT31** | ***PORT Module*** |
| **Category** | Functional |
| **Function** | Fast transient noise on GPIOs may result in a constant high current |
| **Description** | GPIOs subject to fast transient noise (e.g. electromagnetic noise) may see a high constant current. The constant high current is a result of the fast transient noise triggering the internal ESD protection structure and it persists as long as the internal or external current driver sustains the current.<br><br>1. When using in Input mode (PxDIR = 0 ), all GPIOs are impacted by this issue. A fast transient noise on the pin configured as an input or on an adjacent pin could cause a constant high current that is sustained as long as the external driver is present.<br><br>2. When using in Output mode (PxDIR = 1), only the high drive GPIOs (P2.0,P2.1,P2.2 and P2.3) are impacted by this issue. If the affected GPIOs are configured in high drive mode (PxDS register) and they (or adjacent pins) are subject to fast transient noise, it could cause a constant high current. Note that this issue is not seen in GPIOs configured in the output direction with the regular drive strength setting since the high drive mode is required to sustain the high current.<br><br>If the GPIO configuration is reset by a power cycle, the constant high current is no longer sustained. |

**Workaround**

1. For GPIOs configured as input, ensure that they are driven by a current-limited source < 30mA (up to Tj=85C) or 20mA (up to Tj=125C, if allowed for device. See device specific datasheet for maximum allowed operating junction temperature) OR use a series resistance >100 ohm (up to Tj=85C) or 150 ohm (up to Tj=125C, if allowed for device. See device specific datasheet for maximum allowed operating junction temperature) to limit the current.

2. For high drive GPIOs configured as output, ensure adequate protection from fast transients is provided to both the high drive IOs and any adjacent pins.

3. In general, it is recommended to terminate any unused GPIOs in the output direction, driving low to minimize the occurrence of this issue.

4. For guidelines on ESD considerations refer to the document: MSP430 System-level ESD Considerations SLAA530

## PORT32

***PORT Module***

**Category**

Functional

**Function**

Sucessive writes to port registers may cause interrupt to pend incorrectly

**Description**

Writing to the PxIES register sets the corresponding PxIFG bit. The PxIFG bit can be cleared by writing '0' to it (clearing the register).
However if the PxIFG bit is cleared immediately (next instruction cycle) after writing to the PxIES register, the interrupt flag does not clear and stays pending.

**Workaround**

Insert a NOP or __no_operation(); instruction after writing to the PxIES register and before clearing the PxIFG register.

## PORT34

***PORT Module***

**Category**

Functional

**Function**

Timer_A1 & A2 outputs and EUSCIB3 pins should not be used simultaneously

**Description**

The following pin functions cannot be simultaneously active on the device at any pin:

1) TA1.0 and UCB3STE

2) TA2.0 and UCB3CLK

3) TA2.3 and UCB3SIMO

4) TA2.4 and UCB3SOMI

Attempting to use both pin's secondary functions will cause functional conflicts and abnormal behavior of the peripherals. For example, using Timer_A2.3 output will prevent proper operation of EUSCIB3SIMO.

**Workaround**

None.

## PSS4

***PSS Module***

**Category**

Functional

**Function**

SVSMHLP has no impact in LPM4.5 mode

|  |  |
|---|---|
| **Description** | Upon entry into LPM4.5, the SVSMH is set to low power normal performance mode automatically regardless of the SVSMHLP setting.<br>Clearing the SVSMHLP bit (PSSCTL0.SVSMHLP = 0) has no impact in LPM4.5.<br>This bit works as expected in all other low power modes. |
| **Workaround** | None |

## RTC14 — *RTC Module*

|  |  |
|---|---|
| **Category** | Functional |
| **Function** | Polling RTC interrupts may result in a lost interrupt flag |
| **Description** | Polling any RTC interrupt flag mapped to the RTCIV register may result in a missed interrupt if the flags are modified while being read. |
| **Workaround** | Using an interrupt service routine to service flags mapped to the RTCIV register significantly reduces the probability of the issue occurring. |

## SYS26 — *SYS Module*

|  |  |
|---|---|
| **Category** | Functional |
| **Function** | IP Protection Feature not available |
| **Description** | The Regional IP Protection feature is not available up to Revision C for the MSP432P401R/M devices.<br>If your application only requires MSP432 full-chip security to protect your software IP from JTAG read-out, this limitation does not apply to you.<br>Regional IP Protection is used in (i) protecting a specific block of code/data from readout by the rest of the application code and (ii) multi-party firmware development, where each software IP owner delivers an executable IP that is protected from read-out by code from other IP owners using the device.<br><br>For more information on benefits and how to use Regional IP Protection, refer to the document: Software IP Protection on MSP432P4xx Microcontrollers |
| **Workaround** | None |

## TA23 — *TA Module*

|  |  |
|---|---|
| **Category** | Functional |
| **Function** | Polling timer interrupts may result in a lost interrupt flag |
| **Description** | Polling any Timer interrupt flag may result in a missed interrupt if the flags are modified while being read. |
| **Workaround** | Using an interrupt service routine to service timer interrupt flags significantly reduces the probability of the issue occurring. |

## UDMA2 — *UDMA Module*

|  |  |
|---|---|
| **Category** | Software in ROM |
| **Function** | UDMA driver library versions < 4.20.00.03 (SimpleLink MSP432P4 SDK versions < 2.10.00.14) do not support use case where increment size and transfer size are different. |

**Description** UDMA driver library APIs with versions < 4.20.00.03 (SimpleLink MSP432P4 SDK versions < 2.10.00.14) do not support use cases where increment size and transfer sizes are different for either the source or destination. The addresses computation are performed wrongly and this result in erroneous data transfers.

**Workaround** This is fixed in driver library API versions >= 4.20.00.03 (SimpleLink MSP432P4 SDK versions >= 2.10.00.14).

- For TI Drivers users, update to SDK version 2.10.00.14 and TI Drivers based applications automatically leverage the bug fix.

- For Driver Library users, update to SDK version 2.10.00.14 and ensure that the application code uses MAP_DMA_ function calls for Driverlib DMA [instead of direct DMA_ API calls].

## USCI42 *USCI Module*

**Category** Functional

**Function** UART asserts UCTXCPTIFG after each byte in multi-byte transmission

**Description** UCTXCPTIFG flag is triggered at the last stop bit of every UART byte transmission, independently of an empty buffer, when transmitting multiple byte sequences via UART. The erroneous UART behavior occurs with and without DMA transfer.

**Workaround** None.

## USCI43 *USCI Module*

**Category** Functional

**Function** I2C communication stalled when polling UCBxRXIFG

**Description** When using the USCI_B I2C module as a receiver, if an asynchronous event occurs during the read of the UCBxRXIFG interrupt flag, the flag could be unintentionally cleared. This may result in the I2C communication being stalled.

**Workaround** Workaround

1. If the device functions as an I2C master receiver, use synchronous clock sources for operation.

OR

2. Avoid polling UCBxRXIFG. Using the standard interrupt service routine to service the UCBxRXIFG interrupt flag significantly reduces the probability of this errata occurring. Avoid register access to UCBxCTLW0, UCBxSTATW, UCBxRXBUF, UCBxTXBUF, UCBxIFG, & UCBxIV while transmit or receive operation is ongoing and UCBxRXIFG or UCBxTXIFG is expected to be set.

OR

3. Use the clock low time-out select feature (UCCTLO.UCBxCTLW1) to enable a timeout window. In the event that the I2C communication is stalled, use the clock low time-out interrupt to reset the eUSCI module and re-initiate communication.

| | |
|---|---|
| **USCI44** | ***USCI Module*** |
| **Category** | Functional |
| **Function** | Differing clock sources may cause UART communication failure |
| **Description** | When using the USCI_A UART module with differing clock sources for the system clock (MCLK) and the UART source clock (BRCLK), any read or write of the UCAxIFG, UCAxCTLW0, UCAxSTATW, UCAxRXBUF, UCAxTXBUF, UCAxABCTL & UCAxIV registers, while the UCATXIFG or UCARXIFG flag is being set by a UART event could unintentionally clear the UCATXIFG or UCARXIFG. This may result in the UART communication being stalled. |
| **Workaround** | Workaround 1: Use synchronous clocks to source BRCLK and MCLK. Note that the clock frequencies need not be identical and dividers may be used as long as they are using the same clock source.<br><br>Workaround 2: Avoid polling UCAxTXIFG and UCAxRXIFG. Using the standard interrupt service routine to service the interrupt flag significantly reduces access to the USCI registers & hence reduces the probability of this errata occurring.<br>Also, limit all accesses of the UCAxCTLW0, UCAxSTATW, UCAxRXBUF, UCAxTXBUF, UCAxABCTL, UCAxIFG, & UCAxIV registers while transmit or receive operation is ongoing (and UCAxRXIFG or UCAxTXIFG is expected to be set) as this can further reduce the probability of this errata occurring. |

| | |
|---|---|
| **USCI45** | ***USCI Module*** |
| **Category** | Functional |
| **Function** | Unexpected SPI clock stretching possible when UCxCLK is asynchronous to MCLK |
| **Description** | In rare cases, during SPI communication, the clock high phase of the first data bit may be stretched significantly. The SPI operation completes as expected with no data loss. This issue only occurs when the USCI SPI module clock (UCxCLK) is asynchronous to the system clock (MCLK). |
| **Workaround** | Ensure that the USCI SPI module clock (UCxCLK) and the CPU clock (MCLK) are synchronous to each other. |

| | |
|---|---|
| **USCI46** | ***USCI Module*** |
| **Category** | Functional |
| **Function** | UART may receive the first byte incorrectly |
| **Description** | The USCI UART may receive the first byte incorrectly under the following conditions:<br><br>1. If the USCI UART is setup to source BRCLK from ACLK or SMCLK and the clock source is turned off, for example when the module is in idle condition and no other peripheral is requesting the same clock source<br><br>AND<br><br>2. The UART is configured for a baud rate of 9600 and the BRCLK is setup to source a 32kHZ clock (REFO or LFXT)<br><br>If the above two conditions are satisfied, the UART start byte received interrupt flag |

(UCSTTIFG) may not be set and the UART may miss the first bit resulting in an incorrect data byte. Subsequent data bytes are not impacted.

**Workaround**

1. When setting up the UART at 9600 baud, use a source clock >32kHz or if a 32kHz clock source must be used, lower the baud rate to 4800 baud

OR

2. Ensure any other peripheral (for example a timer sourced from ACLK) is active and using the UART clock source thereby keeping the clock turned on even when the UART module is idle.

## USCI47 *USCI Module*

**Category** Functional

**Function** eUSCI SPI slave with clock phase UCCKPH = 1

**Description** The eUSCI SPI operates incorrectly under the following conditions:

1. The eUSCI_A or eUSCI_B module is configured as a SPI slave with clock phase mode UCCKPH = 1

AND

2. The SPI clock pin is not at the appropriate idle level (low for UCCKPL = 0, high for UCCKPL = 1) when the UCSWRST bit in the UCxxCTLW0 register is cleared.

If both of the above conditions are satisfied, then the following will occur:
eUSCI_A: the SPI will not be able to receive a byte (UCAxRXBUF will not be filled and UCRXIFG will not be set) and SPI slave output data will be wrong (first bit will be missed and data will be shifted).
eUSCI_B: the SPI receives data correctly but the SPI slave output data will be wrong (first byte will be duplicated or replaced by second byte).

**Workaround** Use clock phase mode UCCKPH = 0 for MSP SPI slave if allowed by the application.

OR

The SPI master must set the clock pin at the appropriate idle level (low for UCCKPL = 0, high for UCCKPL = 1) before SPI slave is reset (UCSWRST bit is cleared).

OR

For eUSCI_A: to detect communication failure condition where UCRXIFG is not set, check both UCRXIFG and UCTXIFG. If UCTXIFG is set twice but UCRXIFG is not set, reset the MSP SPI slave by setting and then clearing the UCSWRST bit, and inform the SPI master to resend the data.

## USCI48 *USCI Module*

**Category** Functional

**Function** SPI communication stalled when polling UCxxIFG.UCRXIFG

**Description** When using the USCI_A or USCI_B SPI module as a slave, if an asynchronous event occurs during the read of the UCRXIFG (UCxxIFG.UCRXIFG) interrupt flag, the flag could be unintentionally cleared. This may result in the SPI communication being stalled.

**Workaround** 1. If the device functions as a SPI slave, ensure that the USCI clock source is synchronous to the system clock (MCLK).

OR

2. Avoid polling UCxxIFG.UCRXIFG. Using the standard interrupt service routine to service the UCxxIFG.UCRXIFG interrupt flag significantly reduces the probability of this errata occurring. Avoid register access to UCxxCTLW0, UCxxSTATW, UCxxRXBUF, UCxxTXBUF, UCxxIFG, & UCxxIV while transmit or receive operation is ongoing and UCRXIFG is expected to be set.

## USCI50 *USCI Module*

**Category** Functional

**Function** Data may not be transmitted correctly from the eUSCI when operating in SPI 4-pin master mode with UCSTEM = 0

**Description** When the eUSCI is used in SPI 4-pin master mode with UCSTEM = 0 (STE pin used as an input to prevent conflicts with other SPI masters), data that is moved into UCxTXBUF while the UCxSTE input is in the inactive state may not be transmitted correctly. If the eUSCI is used with UCSTEM = 1 (STE pin used to output an enable signal), data is transmitted correctly.

**Workaround** When using the STE pin in conflict prevention mode (UCSTEM = 0), only move data into UCxTXBUF when UCxSTE is in the active state. If an active transfer is aborted by UCxSTE transitioning to the master-inactive state, the data must be rewritten into UCxTXBUF to be transferred when UCxSTE transitions back to the master-active state.

## USCI51 *USCI Module*

**Category** Functional

**Function** UART could lose bytes while transmitting with DMA

**Description** Accessing the RXIFG (which is at the same address as the TXIFG) while transmitting with the DMA, could cause a second interrupt for the DMA and overwrite the transmit buffer, before moving to the Shift register.

**Workaround** Clear pending UART RX-interrupt flags with dummy read and enable RX-interrupt as the below example code:

```
volatile uint8_t temp;
temp =  EUSCI_A_CMSIS(EUSCI_A2_BASE)->RXBUF;
MAP_UART_enableInterrupt(EUSCI_A2_BASE, EUSCI_A_UART_RECEIVE_INTERRUPT);
```

## 7 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

**Changes from April 24, 2019 to May 17, 2021** **Page**

- Changed the document format and structure; updated the numbering format for tables, figures, and cross references throughout the document........................................................................................................6