

MSP430F551x/552x Device Erratasheet – PRELIMINARY Current Version

Devices	Rev:	ADC25	CPU26	CPU27	CPU28	CPU29	CPU30	CPU31	CPU32	CPU33	CPU34	CPU35	DMA4	EEM8	EEM9	EEM12	FLASH33	PMM9	PMM10	PORT15	TB23	USCI26
MSP430F5514	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F5515	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F5524	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F5525	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F5526	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F5527	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F5528	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MSP430F5529	A	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Note: See Appendix for prior revisions

☒ The checkmark means that the issue is present in that revision

Package Markings

RGC64: QFN (RGC) 64 pin

○ M430Fxxx

TI YMS

LLLL #

TI = TI

YM = Year and Month Date Code

LLLL = LOT Trace Code

S = Assembly Site Code

= DIE Revision

o = PIN 1

○ M430Fxxxx

TI YMS #

LLLL G4

TI = TI

YM = Year and Month Date Code


LLLL = LOT Trace Code

S = Assembly Site Code

= DIE Revision

o = PIN 1

PN80: LQFP (PN) 80 pin

 YMLLLLS

M430Fxxx

○ REV #

YM = Year and Month Date Code


LLLL = LOT Trace Code

S = Assembly Site Code

= DIE Revision

o = PIN 1

ZQE80: BGA (ZQE) 80 pin



M430Fxxxxx

YMLLLLS #

○ G1

YM = Year and Month Date Code

LLLL = LOT Trace Code

S = Assembly Site Code

= DIE Revision

o = PIN 1

Detailed Bug Description

ADC25 ADC25 - Bug description

Module: ADC12, Function: Write to ADC12CTL0 triggers ADC12 when CONSEQ = 00

If ADC conversions are triggered by the Timer_B module and the ADC12 is in single-channel single-conversion mode (CONSEQ = 00), ADC sampling is enabled by write access to any bit(s) in the ADC12CTL0 register. This is contrary to the expected behavior that only the ADC12 enable conversion bit (ADC12ENC) triggers a new ADC12 sample.

Workaround:

When operating the ADC12 in CONSEQ = 00 and a Timer_B output is selected as the sample and hold source, temporarily clear the ADC12ENC bit before writing to other bits in the ADC12CTL0 register. The following capture trigger can then be re-enabled by setting ADC12ENC = 1.

CPU26 CPU26 - Bug description

Module: CPU, Function: CALL SP does not behave as expected

When the intention is to execute code from the stack, a CALL SP instruction skips the first piece of data (instruction) on the stack. The second piece of data on SP+2 is used as the first executable instruction.

Workaround:

Write the op code for a NOP as the first instruction on the stack. Begin the intended subroutine at address SP + 2.

CPU27 CPU27 - Bug description

Module: CPU, Function: Program Counter (PC) is corrupted during the context save of a nested interrupt

When a low power mode is entered within an interrupt service routine that has enabled nested interrupts (by setting the GIE bit), and the instruction that sets the low power mode is directly followed by a RETI instruction, an incorrect value of PC + 2 is pushed to the stack during the context save. Hence, the RETI instruction is not executed on return from the nested interrupt and the PC becomes corrupted.

Workaround:

Insert a NOP or __no_operation() intrinsic function between the instruction that sets the lower power mode and the RETI instruction.

Detailed Bug Description (continued)

CPU28 CPU28 - Bug description

Module: CPU, Function: PC is corrupted when using certain extended addressing mode combinations

An extended memory instruction that modifies the program counter executes incorrectly when preceded by an extended memory write-back instruction under the following conditions:

First instruction:

2-operand instruction, extended mode using (register,index), (register,absolute), OR (register,symbolic) addressing modes

Second instruction:

2-operand instruction, extended mode using the (indirect,PC), (indirect auto-increment,PC), OR (indexed [with ind 0], PC) addressing modes

** e.g - BISX.A R6,&AABCD
ANDX.A @R4+,PC **

Workaround:

- 1) Insert a NOP or a `__no_operation()` intrinsic function between the two instructions or
- 2) Do not use an extended memory instruction to modify the PC

CPU29 CPU29 - Bug description

Module: CPU, Function: Using a certain instruction sequence to enter low power mode(s) affects the instruction width of the first instruction in an NMI ISR

If there is a pending NMI request when the CPU enters a low power mode (LPMx) using an instruction of Indexed source addressing mode, and that instruction is followed by a 20-bit wide instruction of Register source and destination addressing modes, the first instruction of the ISR is executed as a 20-bit wide instruction.

** main:

```
...
MOV.W indexed,SR ; Enter LPMx
MOVX.A Reg, Reg ; 20-bit wide instruction
...
```

ISR_start:

```
MOV.B index, Reg ; ERROR - Executed as a 20-bit instruction! **
```

Workaround:

- 1) Insert a NOP or a `__no_operation()` intrinsic function following the instruction that enters the LPMx using indexed addressing mode ... OR ...
- 2) Use a NOP or a `__no_operation()` intrinsic function as first instruction in the ISR ... OR ...
- 3) Do not use the indexed mode to enter LPMx

Detailed Bug Description (continued)

CPU30 CPU30 - Bug description

Module: CPUX

The extended address instructions ADDA, SUBA, CMPA in immediate addressing mode are represented by 4-bytes of opcode (see the *MSP430x5xx User's Guide* Section 5.6.1 for details). In cases where the program counter (PC) is used as the destination register only 2 bytes of the current instruction's 4-byte opcode are accounted for in the PC value. The resulting operation executes as if the immediate value were offset by a value of -2.

Example:

Ideal: ADDA #Immediate-4, PC ...is equivalent to...

Actual: ADDA #Immediate-2, PC

** NOTE: The MOV instruction is not affected **

Workaround:

- 1) Modify immediate value in software to account for the offset of 2
- or
- 2) Use extended 20-bit instructions (addx.a, subx.a, cmpx.a) instead

CPU31 CPU31 - Bug description

Module: CPUX

When the instruction PUSHX.A is executed using the indirect auto-increment mode with the stack pointer (SP) as the source register [PUSHX.A @SP+] the SP is consequently corrupted. Instead of decrementing the value of the SP by four, the value of the SP is replaced with the data pointed to by the SP previous to the PUSHX.A instruction execution.

Workaround:

None. The compiler will not generate a PUSHX.A instruction that involves the SP.

CPU32 CPU32 - Bug description

Module: CPUX

When the instruction CALLA PC is executed, the program counter (PC) that is pushed onto the stack during the context save is incorrectly offset by a value of -2.

Workaround:

None. The compiler will not generate a CALLA PC instruction.

Detailed Bug Description (continued)

CPU33 CPU33 - Bug description

Module: CPUX

When the Stack Pointer (SP) is used as the destination register in the CALLA index(Rdst) instruction and is preceded by a PUSH or PUSHX instruction in any of the following addressing modes: Absolute, Symbolic, Indexed, Indirect register or Indirect auto increment, the "index" of the CALLA instruction is not sign extended to 20-bits and is always treated as a positive value. This causes the Program Counter to be set to a wrong address location when the index of the CALLA instruction represents a negative offset.

NOTE:

- 1) This erratum only applies when the instruction sequence is: PUSH or PUSHX followed by CALLA index(SP)
- 2) This erratum does not apply if the PUSH or PUSHX instruction is used in the Register or Immediate addressing mode
- 3) This erratum only applies when SP is used as the destination register in the CALLA index(Rdst) instruction

Workaround:

Place a "NOP" instruction in between the PUSH or PUSHX and the CALLA index(SP) instructions.

NOTE: This bug has no compiler impact as the compiler will not generate a CALLA instruction that uses indexed addressing mode with the SP.

CPU34 CPU34 - Bug description

Module: CPU, Function: CPU may be halted if a conditional jump is followed by a rotate PC instruction

If a conditional jump instruction (JZ, JNZ, JC, JNC, JN, JGE, JL) is followed by an Address Rotate instruction on the PC (RRCM, RRAM, RLAM, RRUM) and the jump is not performed, the CPU is halted.

Workaround:

Insert a NOP between the conditional jump and the rotate PC instructions.

CPU35 CPU35 - Bug description

Module: CPU, Function: Instruction BIT.B @Rx,PC uses the wrong PC value

The BIT(.B/.W) instruction in indirect register addressing mode uses the wrong PC value. This instruction is represented by 2 bytes of opcode. If the Program Counter (PC) is used as the destination register, the 2 opcode bytes of the current BIT instruction are not accounted for. The resulting operation executes the instruction using the wrong PC value and this affects the results in the Status Register (SR).

Workaround:

None.

Note: The compiler will not generate a BIT instruction that uses the PC as an operand.

Detailed Bug Description (continued)

DMA4 DMA4 - Bug description

Module: DMA, Function: Corrupted write access to 20-bit DMA registers

When a 20-bit wide write to a DMA address register (DMAxSA or DMAxDA) is interrupted by a DMA transfer, the register contents may be unpredictable.

Workaround:

- 1) Design the application to guarantee that no DMA access interrupts 20-bit wide accesses to the DMA address registers.
- or
- 2) When accessing the DMA address registers, enable the Read Modify Write disable bit (DMARMWDIS = 1) or temporarily disable all active DMA channels (DMAEN = 0).
- or
- 3) Use word access for accessing the DMA address registers. Note that this limits the values that can be written to the address registers to 16-bit values (lower 64K of Flash).

EEM8 EEM8 - Bug description

Module: EEM, Function: Debugger stops responding when using the DMA

In repeated transfer mode, the DMA automatically reloads the size counter (DMAxSZ) once a transfer is complete and immediately continues to execute the next transfer unless the DMA Enable bit (DMAEN) has been previously cleared. In burst-block transfer mode, DMA block transfers are interleaved with CPU activity 80/20% - of ten CPU cycles, eight are allocated to a block transfer and two are allocated for the CPU.

Since the JTAG system must wait for the CPU bus to be clear in order to halt the device, it can only do so when two conditions are met:

- 1) If three clock cycles after any DMA transfer, the DMA is no longer requesting the bus and
- 2) If the CPU is not requesting the bus

Therefore, if the DMA is configured to operate in the repeat burst-block transfer mode and a breakpoint is set between the line of code that triggers the DMA transfers and the line that clears the DMAEN bit the DMA will always request the bus and the JTAG system will never gain control of the device.

Workaround:

When operating the DMA in repeat burst-block transfer mode, set breakpoint(s) only when the DMA transfers are not active (before the start or after the end of the DMA transfers).

EEM9 EEM9 - Bug description

Module: EEM, Function: Combined triggers on the PUSH instruction may be missed

When the PUSH instruction is used in any addressing mode except register or immediate modes, a combined trigger whose conditions are defined by a PUSH instruction fetch AND a successful match of the value being pushed onto stack may be missed.

Workaround:

None.

Detailed Bug Description (continued)

EEM12 EEM12 - Bug description

Module: EEM, Function: Debugger halts at incorrect location

When using breakpoints or single stepping in the presence of multiple background-interrupts, the debugger may not halt at the breakpoint location and will instead halt at the first line of any interrupt service routine.

Workaround:
None

FLASH33 FLASH33 - Bug description

Module: FLASH, Function: Flash erase/program with fsystem <160 kHz causes code execution to fail

A flash erase or flash program operation with the system frequency (fsystem) <160 kHz causes the program execution (executing out of main or info memory) that follows to fail.

Workaround:
Make sure the fsystem >160 kHz before doing a flash erase or program operation

PMM9 PMM9 - Bug description

Module: PMM, Function: False SVSxIFG events when SVSxMD = 0 && SVSxFP = 1

When SVSxMD = 0 (default setting), the SVSx comparators are disabled automatically in LPM2/3/4 (disabling the respective SVSx events) and are then re-enabled on return to active mode. The comparators of the SVS require a certain amount of time to stabilize and output a correct result once re-enabled; this time is different for the Fast Performance versus the Normal Performance modes. This time to stabilize the SVS comparators is intended to be accounted for by a built-in event-masking delay of 2 μ s when Fast Performance mode is enabled. However, the comparators of the SVS in Fast Mode take longer than 2 μ s to stabilize so the possibility exists that a false positive will be triggered on the SVSH or SVSL.

Workaround:
If the Fast Performance mode is to be enabled for either the high- or low-side SVS comparators, the respective SVSxMD bits must be set (SVSxMD = 1) such that the SVS comparators are not temporarily shut off in LPM2/3/4. Note that this is equivalent to a 2 μ A adder to the average low power mode current, per the device-specific datasheet, for each SVSx that must remain enabled.

The Fast Performance mode may be desired for the low-side SVS due to its impact on the low power mode wake-up time of the CPU (FAST-WAKE-UP vs SLOW-WAKE-UP). In this case, it should be noted that the low-side SVS is only truly useful when switching between the different states of PMMCOREV, meaning during the application initialization when the correct core voltage level must be assured in order to attain a given CPU frequency. Therefore, the low-side SVS is recommended to be disabled following the setting of PMMCOREV so the CPU can still come up active in $\sim 5 \mu$ s and the high-side SVS is still available for the robust protection against an external low voltage condition.

Detailed Bug Description (continued)

PMM10 PMM9 - Bug description

Module: PMM, Function: False SVSxIFG events when SVSxMD = 0 && SVSxFP = 1

SVS/SVM interrupt flag functionality is disabled after a Power Up Clear (PUC) Reset if the SVS was disabled before the PUC reset was applied.

Workaround:

A write access to the intended SVSx register after PUC re-enables the SVS & SVM interrupt flags.

PORT15 PORT15 - Bug description

Module: PortMapper, Function: In-system debugging causes the PMALOCKED bit to be always set

The port mapping controller registers cannot be modified when single-stepping or halting at break points between a valid password write to the PMAPWD register and the expected lock of the port mapping (PMAP) registers. This causes the PMAPLOCKED bit to remain set and not clear as expected.

Note that this erratum only applies to in-system debugging and is not applicable when operating in free-running mode.

Workaround:

Do not single step through or place break points in the port mapping configuration section of code.

Detailed Bug Description (continued)

TB23 TB23 - Bug description

Module: Timer_B, Function: Timer_B7 outputs not set correct to Hi-Z state

When the TBOUTH pin functionality is selected, the TimerB outputs on the non-port mapping pins are interchanged such that:

- If TB0 is selected as TimerB output, TBOUTH puts the TB6 output into hi-impedance state
 - If TB1 is selected as TimerB output, TBOUTH puts the TB5 output into hi-impedance state
 - If TB2 is selected as TimerB output, TBOUTH puts the TB4 output into hi-impedance state
 - If TB4 is selected as TimerB output, TBOUTH puts the TB2 output into hi-impedance state
 - If TB5 is selected as TimerB output, TBOUTH puts the TB1 output into hi-impedance state
 - If TB6 is selected as TimerB output, TBOUTH puts the TB0 output into hi-impedance state
- TB3 is not affected.

NOTE: This erratum does not apply if the TimerB outputs pins are selected via the port mapper controller.

Workaround:

- 1) Configure the TimerB output pin functionality via the port mapper controller
or
 - 2) Configure TimerB outputs on normal port pins such that the interchanging of TimerB outputs with TBOUTH functionality enabled does not affect the functionality.
- If one TimerB output is required, use TB3
 - If two TimerB outputs are required, use one of the following TimerB output combinations - TB0+TB6 or TB1+TB5 or TB2+TB4
 - If three TimerB outputs are required, combine TB3 with one of the following combinations - TB0+TB6 or TB1+TB5 or TB2+TB4
 - If four TimerB outputs are required, use any two of the following TimerB output combinations - TB0+TB6 or TB1+TB5 or TB2+TB4
 - If five TimerB outputs are required, combine TB3 with any two of the following TimerB output combinations - TB0+TB6 or TB1+TB5 or TB2+TB4
 - If six TimerB outputs are required, use the following TimerB output combinations - TB0+TB6 or TB1+TB5 or TB2+TB4
 - If all seven TimerB outputs are required, since all the TimerB outputs are selected, TBOUTH functionality erratum would not affect this case.

USCI26 USCI26 - Bug description

Module: USCI_B, Function: Tbuf parameter violation in I2C multi-master mode

In multi-master I2C systems the timing parameter t_{buf} (bus free time between a stop condition and the following start) is not guaranteed to match the I2C specification of 4.7 μ s in standard mode and 1.3 μ s in fast mode. If the UCTXSTT bit is set during a running I2C transaction, the USCI module waits and issues the start condition on bus release causing the violation to occur.

Note: It is recommended to check if UCBBUSY bit is cleared before setting UCTXSTT = 1.

Workaround:

None

Appendix: Prior Versions

None



IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products

Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
RF/IF and ZigBee® Solutions	www.ti.com/lprf

Applications

Audio	www.ti.com/audio
Automotive	www.ti.com/automotive
Broadband	www.ti.com/broadband
Digital Control	www.ti.com/digitalcontrol
Medical	www.ti.com/medical
Military	www.ti.com/military
Optical Networking	www.ti.com/opticalnetwork
Security	www.ti.com/security
Telephony	www.ti.com/telephony
Video & Imaging	www.ti.com/video
Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2009, Texas Instruments Incorporated