MSP430FR5949 Device Erratasheet

1 Revision History

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

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

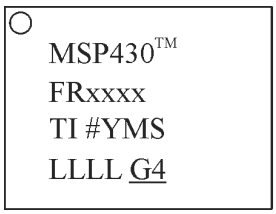
Errata Number	Rev H	Rev G	Rev F
ADC38	✓	✓	✓
ADC41	✓	✓	✓
ADC42	✓	✓	✓
ADC43	✓	✓	✓
ADC64	✓	✓	✓
AES1	✓	✓	✓
COMP7	✓	✓	✓
COMP10	✓	✓	✓
CPU21	✓	✓	✓
CPU22	✓	✓	✓
CPU40	✓	✓	✓
CPU46	✓	✓	✓
CS3	✓	✓	✓
CS7	✓	✓	✓
CS12	✓	✓	✓
DMA7	✓	✓	✓
DMA11	✓	✓	✓
EEM19	✓	✓	✓
EEM23	✓	✓	✓
EEM27	✓	✓	✓
EEM28	✓	✓	✓
EEM30	✓	✓	✓
EEM31	✓	✓	✓
GC1	✓	✓	✓
JTAG27	✓	✓	✓
PMM24		✓	✓
PORT28	✓	✓	✓
REF9	✓	✓	✓
RTC10	✓	✓	✓
USCI41	✓	✓	✓
USCI42	✓	✓	✓
USCI45	✓	✓	✓
WDG5	✓	✓	✓

2 Package Markings

DA38 *TSSOP (DA), 38 Pin*

YM = Year and Month Date Code
S = Assembly Site Code
# = Die Revision
LLLL = Assembly Lot Code
○ = Pin 1

RHA40 *QFN (RHA), 40 Pin*



YM = Year and Month Date Code
S = Assembly Site Code
# = Die Revision
LLLL = Assembly Lot Code
○ = Pin 1

3 Memory-Mapped Hardware Revision (TLV Structure)

Die Revision	TLV Hardware Revision
Rev H	40h
Rev G	31h
Rev F	30h

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

4 Detailed Bug Description

ADC38 *ADC12_B Module*

Function External ADC trigger without toggling ENC bit might prevent further ADC conversions.

Description The ADC may stop sampling and converting until the module is reset if an external (timer) trigger occurs without toggling the ADC12CTL0.ADC12ENC bit at:

- The end of sequence in the sequence-of-channel mode.

- The end of conversion in single-channel mode.

Workaround Ensure ADC12CTL0.ADC12ENC bit is always toggled before providing any new External Trigger to ADC.

ADC41 *ADC12_B Module*

Function ADC conversion only triggers when sourced from TA0 and TA3 in LPM3/LPM4

Description If the ADC clock is sourced from ADC12OSC (ADC12CTL1.ADC12SSELx = ADC12OSC), the reference module is not used (REFCTL0.REFON = 0), and the device is in LPM3/LPM4 mode, ADC conversion cannot be triggered when using Timer TB0, TA1 and TA2 (ADC12CTL1.ADC12SHSx).

Workaround 1. Use Timer TA0 or TA3 to trigger ADC conversion in LPM3/LPM4 mode

OR

2. Configure ADC clock to be sourced from ACLK (ADC12CTL1.ADC12SSELx = ACLK).

OR

3. Use Active/LPM0/LPM1/LPM2 mode

OR

4. Use the internal voltage reference (REFCTL0.REFON = 1)

ADC42 *ADC12_B Module*

Function ADC stops converting when successive ADC is triggered before the previous conversion ends

Description Subsequent ADC conversions are halted if a new ADC conversion is triggered while ADC is busy. ADC conversions are triggered manually or by a timer. The affected ADC modes are:

- sequence-of-channels

- repeat-single-channel

- repeat-sequence-of-channels (ADC12CTL1.ADC12CONSEQx)

In addition, the timer overflow flag cannot be used to detect an overflow (ADC12IFGR2.ADC12TOVIFG).

Workaround 1. For manual trigger mode (ADC12CTL0.ADC12SC), ensure each ADC conversion is completed by first checking ADC12CTL1.ADC12BUSY bit before starting a new conversion.

2. For timer trigger mode (ADC12CTL1.ADC12SHP), ensure the timer period is greater than the ADC sample and conversion time.

To recover the conversion halt:

1. Disable ADC module (ADC12CTL0.ADC12ENC = 0 and ADC12CTL0.ADC12ON = 0)

2. Re-enable ADC module (ADC12CTL0.ADC12ON = 1 and ADC12CTL0.ADC12ENC = 1)

3. Re-enable conversion

ADC43 *ADC12_B Module*

Function DMA does not trigger at the end of an ADC12 sequence of channels

Description The DMA transfer is triggered at the end of every ADC conversion when the ADC is configured to convert in a sequence of channels (ADC12CTL1.CONSEQ = 1 or 3.) This causes the DMA transfer to trigger prematurely after each ADC conversion instead of triggering only at the end of the conversion sequence.

Workaround Design the application to expect the DMA trigger at the end of every ADC conversion. For example, if a block transfer at the end of the sequence is originally desired, configure the DMA in single transfer mode with size = length of the sequence. The DMA transfer occurs at each conversion, but the DMA interrupt will still occur at the end of the sequence.

ADC64 *ADC12_B Module*

Function Incorrect conversion result in extended sample mode in some conditions

Description The ADC12 conversion result can be incorrect if the extended sample mode is selected (ADC12SHP = 0), ADC12VRSEL is set to 0, 2, 4, 6, 12, 14 (VR+ and VR- unbuffered), and the ADC sample time is less than 6 ADC clock cycles.

Workaround 1) Use Pulse sample mode (ADC12SHP=1) if sample time less than 6 ADC clock cycles is needed;

OR

2) In extended sample mode (ADC12SHP = 0) increase the sample time to at least 6 ADC clock cycles;

OR

3) Use reference mode corresponding to ADC12VRSEL =1,3,5,7,9,13,15

AES1 *AES256 Module*

Function Ongoing AES operation cannot be aborted by writing to AESAXIN

Description Writing to AESAXIN register when AESASTAT.AESBUSY bit is set does abort the ongoing AES operation or set the AESACTL0.AESERRFG bit.

Workaround Always let AES operation run to completion (i.e. do not abort). Ignore the encryption/decryption output if AESAXIN is written when AESASTAT.AESBUSY is set.

COMP7 *COMP_E Module*

Function Comparator triggers false output at low overdrive levels

Description When the differential voltage on the comparator input pins is smaller than the comparator offset according to the datasheet, the comparator can provide a false output.

Workaround Drive the differential voltage to above the comparator offset according to the datasheet.

COMP10 *COMP_E Module*

Function Comparator port output toggles when entering or leaving LPM3/LPM4

Description The comparator port pin output (CECTL1.CEOUT) erroneously toggles when device enters or leaves LPM3/LPM4 modes under the following conditions:

1) Comparator is disabled (CECTL1.CEON = 0)

AND

2) Output polarity is enabled (CECTL1.CEOUTPOL = 1)

AND

3) The port pin is configured to have CEOUT functionality.

For example, if the CEOUT pin is high when the device is in Active Mode, CEOUT pin becomes low when the device enters LPM3/LPM4 modes.

Workaround When the comparator is disabled, ensure at least one of the following:

1) Output inversion is disabled (CECTL.CEOUTPOL = 0)

OR

2) Change pin configuration from CEOUT to GPIO with output low.

CPU21 *CPUXv2 Module*

Function Using POPM instruction on Status register may result in device hang up

Description When an active interrupt service request is pending and the POPM instruction is used to set the Status Register (SR) and initiate entry into a low power mode , the device may hang up.

Workaround None. It is recommended not to use POPM instruction on the Status Register.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU21
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

CPU22 *CPUXv2 Module*

Function Indirect addressing mode with the Program Counter as the source register may produce unexpected results

Description When using the indirect addressing mode in an instruction with the Program Counter (PC) as the source operand, the instruction that follows immediately does not get executed.

For example in the code below, the ADD instruction does not get executed.

```
mov @PC, R7
add #1h, R4
```

Workaround Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	User is required to add the compiler or assembler flag option below. --silicon_errata=CPU22
MSP430 GNU Compiler (MSP430-GCC)	MSP430-GCC 4.9 build 167 or later	

CPU40 *CPUXv2 Module*

Function PC is corrupted when executing jump/conditional jump instruction that is followed by instruction with PC as destination register or a data section

Description If the value at the memory location immediately following a jump/conditional jump instruction is 0X40h or 0X50h (where X = don't care), which could either be an instruction opcode (for instructions like RRCM, RRAM, RLAM, RRUM) with PC as destination register or a data section (const data in flash memory or data variable in

RAM), then the PC value is auto-incremented by 2 after the jump instruction is executed; therefore, branching to a wrong address location in code and leading to wrong program execution.

For example, a conditional jump instruction followed by data section (0140h).

@0x8012 Loop DEC.W R6

@0x8014 DEC.W R7

@0x8016 JNZ Loop

@0x8018 Value1 DW 0140h

Workaround In assembly, insert a NOP between the jump/conditional jump instruction and program code with instruction that contains PC as destination register or the data section.

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	IAR EW430 v5.51 or later	For the command line version add the following information Compiler: --hw_workaround=CPU40 Assembler:-v1
TI MSP430 Compiler Tools (Code Composer Studio)	v4.0.x or later	
MSP430 GNU Compiler (MSP430-GCC)	Not affected	

CPU46 *CPUXv2 Module*

Function POPM peforms unexpected memory access and can cause VMAIFG to be set

Description When the POPM assembly instruction is executed, the last Stack Pointer increment is followed by an unintended read access to the memory. If this read access is performed on vacant memory, the VMAIFG will be set and can trigger the corresponding interrupt (SFRIE1.VMAIE) if it is enabled. This issue occurs if the POPM assembly instruction is performed up to the top of the STACK.

Workaround

If the user is utilizing C, they will not be impacted by this issue. All TI/IAR/GCC pre-built libraries are not impacted by this bug. To ensure that POPM is never executed up to the memory border of the STACK when using assembly it is recommended to either

1. Initialize the SP to

a. TOP of STACK - 4 bytes if POPM.A is used

b. TOP of STACK - 2 bytes if POPM.W is used

OR

2. Use the POPM instruction for all but the last restore operation. For the the last restore operation use the POP assembly instruction instead.

For instance, instead of using:

```
POPM.W #5,R13
```

Use:

```
POPM.W #4,R12
POP.W R13
```

Refer to the table below for compiler-specific fix implementation information.

IDE/Compiler	Version Number	Notes
IAR Embedded Workbench	Not affected	C code is not impacted by this bug. User using POPM instruction in assembler is required to implement the above workaround manually.
TI MSP430 Compiler Tools (Code Composer Studio)	Not affected	C code is not impacted by this bug. User using POPM instruction in assembler is required to implement the above workaround manually.
MSP430 GNU Compiler (MSP430-GCC)	Not affected	C code is not impacted by this bug. User using POPM instruction in assembler is required to implement the above workaround manually.

CS3 *CS Module*

Function

Possible PMM overload when increasing SMCLK frequency to higher than MCLK frequency

Description

When switching SMCLK frequency from lower to higher than MCLK frequency, the device can crash and hang until it is restarted by a reset.

Workaround

1.

a. Increase SMCLK to MCLK frequency to ensure they are the same

b. Then, increase SMCLK frequency to target frequency

OR

2.

a. Increase MCLK frequency to the desired SMCLK frequency

b. Then, increase SMCLK to target frequency

c. Then, decrease MCLK back to the original MCLK frequency

CS7 *CS Module*

Function DCO clock frequency out of specification when returning from LPM2, LPM3 or LPM4

Description When waking up from LPM2, LPM3 or LPM4 the first clocks generated by the DCO are not within the specified frequency range for approximately 13us (independent of the selected frequency). Any observable overshoot of the frequency is not critical for the device functionality. Frequency undershoots can be considered as additional wake-up delay because the frequency is below the target and less clocks are generated than expected. The overall impact of the clock overshoots and undershoots during stabilization is approximately 2us of additional delay.

Workaround Account for frequency undershoots as additional wake-up delay of about 2us.

CS12 *CS Module*

Function DCO overshoot at frequency change

Description When changing frequencies (CSCTL1.DCOFSEL), the DCO frequency may overshoot and exceed the datasheet specification. After a time period of 10us has elapsed, the frequency overshoot settles down to the expected range as specified in the datasheet. The overshoot occur when switching to and from any DCOFSEL setting and impacts all peripherals using the DCO as a clock source. A potential impact can also be seen on FRAM accesses, since the overshoot may cause a temporary violation of FRAM access and cycle time requirements.

Workaround When changing the DCO settings, use the following procedure:

1) Store the existing CSCTL3 divider into a temporary unsigned 16-bit variable

2) Set CSCTL3 to divide all corresponding clock sources by 4 or higher

3) Change DCO frequency

4) Wait ~10us

5) Restore the divider in CSCTL3 to the setting stored in the temporary variable.

The following code example shows how to increase DCO to 16MHz.

```
uint16_t tempCSCTL3 = 0;
CSCTL0_H = CSKEY_H;        // Unlock CS registers
/* Assuming SMCLK and MCLK are sourced from DCO */
/* Store CSCTL3 settings to recover later */
tempCSCTL3 = CSCTL3;
/* Keep overshoot transient within specification by setting clk sources
to divide by 4*/
/* Clear the DIVS & DIVM masks (~0x77)and set both fields to 4 divider */
CSCTL3 = CSCTL3 & (~(0x77)) | DIVS__4 | DIVM__4;
CSCTL1 = DCOFSEL_4 | DCORSEL;        // Set DCO to 16MHz
/* Delay by ~10us to let DCO settle. 60 cycles = 20 cycles buffer +
(10us / (1/4MHz)) */
__delay_cycles(60);
CSCTL3 =  tempCSCTL3;      // Set all dividers
CSCTL0_H = 0;                    // Lock CS registers
```

DMA7 *DMA Module*

Function DMA request may cause the loss of interrupts

Description If a DMA request starts executing during the time when a module register containing an

interrupt flags is accessed with a read-modify-write instruction, a newly arriving interrupt from the same module can get lost. An interrupt flag set prior to DMA execution would not be affected and remain set.

Workaround 1. Use a read of Interrupt Vector registers to clear interrupt flags and do not use read-modify-write instruction.

OR

2. Disable all DMA channels during read-modify-write instruction of specific module registers containing interrupts flags while these interrupts are activated.

DMA11 *DMA Module*

Function DMA trigger during transition into LPM1/LPM2/LPM3/LPM4 may cause the device to reset

Description The device performs a PUC if the DMA is triggered while the system transitions the power mode from active to LPM1/LPM2/LPM3/LPM4. The susceptible time window at power mode transition is 1 MCLK cycle + 500ns.

Workaround Ensure that the DMA is not triggered during active mode to LPM1/LPM2/LPM3/LPM4 transition.

1. Perform a software DMA trigger inside a module interrupt service routine instead of using the modules DMA trigger.

OR

2. Use Active Mode or LPM0 when using DMA.

EEM19 *EEM Module*

Function DMA may corrupt data in debug mode

Description When the DMA is enabled and the device is in debug mode, the data written by the DMA may be corrupted when a breakpoint is hit or when the debug session is halted.

Workaround This erratum has been addressed in MSPDebugStack version 3.5.0.1. It is also available in released IDE EW430 IAR version 6.30.3 and CCS version 6.1.1 or newer.

If using an earlier version of either IDE or MSPDebugStack, do not halt or use breakpoints during a DMA transfer.

NOTE: This erratum applies to debug mode only.

EEM23 *EEM Module*

Function EEM triggers incorrectly when modules using wait states are enabled

Description When modules using wait states (USB, MPY, CRC and FRAM controller in manual mode) are enabled, the EEM may trigger incorrectly. This can lead to an incorrect profile counter value or cause issues with the EEMs data watch point, state storage, and breakpoint functionality.

Workaround None.

NOTE: This erratum affects debug mode only.

EEM27 *EEM Module*

Function Switching off FRAM LDO stalls device during debug access

Description With the "Enable Ultra Low Power debug/LPMx.5 debug" option disabled in the IDE, if user application switches off the FRAM LDO (FRPWR = 0) and a debug halt is requested during this time, device debug control is lost and the debug session must be restarted. At this point, the code execution is also stalled.

The following error message is observed:

IAR - "Internal error: (State)"

CCS - "MSP430: Trouble Halting Target CPU: Internal error"

Workaround If IDE error message is observed, restart the debug session or perform a hardware reset. Turn on "Enable Ultra Low Power debug/LPMx.5 debug" option in the IDE debug settings.

EEM28 *EEM Module*

Function Clock outputs observed on port module during LPMx in debug mode

Description When the device is in LPMx mode, if a debug halt is requested and if the port pin is configured as MCLK, SMCLK, or ACLK output, these clocks are observed on the port pin. Depending on the LPM mode (see Device User's Guide), peripherals that are clocked from MCLK, SMCLK, or ACLK are still halted during debug halt state.

For example, if the device is in debug halt in LPM3 mode and a port pin is configured as SMCLK output, SMCLK can be observed on the pin. But the peripherals sourced from SMCLK are still halted as expected.

Workaround None

EEM30 *EEM Module*

Function Missed breakpoint if FRAM power supply is disabled

Description The FRAM power supply can be disabled (GCCTL0.FRPWR = 0) prior to LPM entry to save power. Upon wakeup, if a breakpoint is set on an the first instruction that accesses FRAM, the breakpoint may be missed.

Workaround None. This issue affects debug mode only.

EEM31 *EEM Module*

Function Breakpoint trigger may be lost when MPU is enabled

Description A data value written to FRAM can be used as a trigger condition for breakpoints during a debug session. This trigger can be lost if the FRAM access is made to an address that has been write-protected by the MPU.

Workaround None. This issue affects debug mode only.

GC1 *GC Module*

Function Uncorrectable memory bit error flag (GCCTL1.UBDIFG) does not trigger NMI

Description GCCTL1.UBDIFG flag is an interrupt flag that gets set if an uncorrectable bit error has been detected in non-volatile memory. The GCCTL1.UBDIFG flag does not trigger a NMI request even if GCCTL0.UBDRSTEN = 0. In this case, the application is not notified via a NMI request if an uncorrectable bit error occurred in non-volatile memory.

Workaround Set GCCTL0.UBDRSTEN = 1 to trigger a PUC. Check GCCTL1.UBDIFG flag after a PUC has been initiated.

JTAG27 *JTAG Module*

Function Unintentional code execution after programming via JTAG/SBW

Description The device can unintentionally start executing code from uninitialized RAM addresses 0x0006 or 0x0008 after being programming via the JTAG or SBW interface. This can result in unpredictable behavior depending on the contents of the address location.

Workaround 1. If using programming tools purchased from TI (MSP-FET, LaunchPad), update to CCS version 6.1.3 later or IAR version 6.30 or later to resolve the issue.

2. If using the MSP-GANG Production Programmer, use v1.2.3.0 or later.

3. For custom programming solutions refer to the specification on MSP430 Programming Via the JTAG Interface User's Guide (SLAU320) revision V or newer and use MSPDebugStack v3.7.0.12 or later.

For MSPDebugStack (MSP430.DLL) in CCS or IAR, download the latest version of the development environment or the latest version of the MSPDebugStack

NOTE: This only affects debug mode.

PMM24 *PMM Module*

Function Device may enter lockup state during wake-up from LPM3 and LPM4

Description The device may enter a lockup state during an interrupt-triggered wake up from LPM3 or LPM4. The device will remain in lockup state, unable to respond to the interrupt or continue application execution, until a power cycle brings it back to reset state.

LPM3.5 and LPM4.5 are not affected by this behavior.

Workaround 1) Use LPM2 instead of LPM3 or LPM4. Refer to the device specific datasheet for details on LPM2 wake up time and power consumption.

OR

2) If the application only uses RTC or GPIO as a wakeup source, use LPM3.5 or LPM4.5 instead. Refer to the device specific datasheet for details on LPM3.5/LPM4.5 wake up times and power consumption.

Note: When using LPM3.5/LPM4.5, the Compute Through Power Loss (CTPL) utility APIs (part of the FRAM software utilities) can be used to configure device behavior prior to LPM entry and on wake-up.

PORT28 *PORT Module*

Function Pull-down resistor of TEST/SBWTCK pin

Description The device's internal pull-down resistor on the TEST/SBWTCK pin gets disabled if the SYS control bit SFRRPCR.SYSRSTRE is cleared. This can lead to increased current

consumption and unintentionally-enabled JTAG access to the device.

Workaround 1) Do not clear the SFRRPCR.SYSRSTRE bit, use the SFRRPCR.SYSRSTRUP bit to define direction of the internal resistor on RST/NMI/SBWTDIO pin instead.

OR

2) Ensure a zero voltage level of TEST/SBWTCK pin by connecting the pin to an external component (e.g. external pull-down resistor) on the PCB.

REF9 *REF_A Module*

Function REFON Feature

Description The Reference module does not not provide REF voltage to Comparator module when the REFON bit is set (REFCTL0.REFON=1).

Workaround 1. Use REFBGOT bit of the REFCTL0 regsiter instead of REFON bit to provide REF voltage to Comparator.

OR

2. Enable the Comparator module with internal REF setting (CEREFL + CERS bits of the CECTL2 register) to request the REF module.

RTC10 *RTC_B Module*

Function RTC interrupt flag can be lost during LPMx.5 entry

Description An RTC interrupt flag can get lost if it triggers within a small critical time window of the device's entry into LPM3.5. This results in the RTC interrupt flag not triggering a wake-up from LPM3.5. The subsequent RTC interrupt flag is captured to wake device up from LPM3.5.

Workaround Use LPM3 for timing-critical applications where the device is entering LPM3.5 close to the RTC interrupt flag triggering.

USCI41 *eUSCI Module*

Function UCBUSY bit of eUSCIA module stuck to 1 when device is in SPI mode.

Description When eUSCIA is configured in SPI mode, and the last transfer bit changes from 0 to 1, the UCBUSY bit gets stuck to 1. This happens in all four combinations of Clock Phase and Clock Polarity options (UCAxCTLW0.UCCKPH & UCAxCTLW0.UCCKPL bits). There is no data loss or corruption. Because the USCBUSY bit is stuck to 1, the clock request stays enabled and adds additional current consumption in low power mode operation.

Workaround Check on transmit or receive interrupt flag UCTXIFG/UCRXIFG instead of UCBUSY to know if the UCAxTXBUF buffer is empty or ready for the next complete character.

USCI42 *eUSCI Module*

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.

USCI45 *eUSCI Module*

Function Unexpected SPI clock stretching possible

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.

WDG5 *WDT_A Module*

Function Clock Fail-Safe feature in LPMx.5

Description The watchdog clock fail-safe feature does not prevent the device from going into LPMx.5. As a result, the device enters LPMx.5 state independently of running the watchdog. Note that the watchdog is off in LPMx.5.

Workaround None.

5 Document Revision History

Changes from device specific erratasheet to document Revision A.

1. Errata CS7 was added to the errata documentation.
2. Errata RTC10 was added to the errata documentation.

Changes from document Revision A to Revision B.

1. Errata DMA7 was added to the errata documentation.
2. Errata CPU43 was removed from the errata documentation.
3. Silicon Revision G was added to the errata documentation.

Changes from document Revision B to Revision C.

1. DMA7 Workaround was updated.
2. EEM23 Description was updated.
3. DMA7 Description was updated.

Changes from document Revision C to Revision D.

1. ADC38 Function was updated.
2. Errata AES1 was added to the errata documentation.
3. ADC38 Description was updated.
4. ADC38 Workaround was updated.

Changes from document Revision D to Revision E.

1. Errata USCI41 was added to the errata documentation.

Changes from document Revision E to Revision F.

1. Errata PMM24 was added to the errata documentation.

Changes from document Revision F to Revision G.

1. Errata PORT28 was added to the errata documentation.

Changes from document Revision G to Revision H.

1. EEM19 Workaround was updated.
2. Errata REF9 was added to the errata documentation.

Changes from document Revision H to Revision I.

1. Errata CS12 was added to the errata documentation.
2. Errata USCI42 was added to the errata documentation.
3. Errata EEM30 was added to the errata documentation.
4. Errata EEM31 was added to the errata documentation.
5. Errata JTAG27 was added to the errata documentation.
6. Errata COMP10 was added to the errata documentation.

Changes from document Revision I to Revision J.

1. Silicon Revision H was added to the errata documentation.
2. Errata CPU46 was added to the errata documentation.

Changes from document Revision J to Revision K.

1. CPU21 was added to the errata documentation.
2. CPU22 was added to the errata documentation.
3. USCI45 was added to the errata documentation.
4. Workaround for RTC10 was updated.
5. Workaround for CPU40 was updated.
6. Workaround for CPU46 was updated.
7. Description for USCI41 was updated.

Changes from document Revision K to Revision L.

1. ADC64 was added to the errata documentation.
2. TLV hardware revision ID for Rev H was updated.
3. Workaround for CPU46 was updated.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265