Errata

AWR2544 Device Errata

Table of Contents

1 Introduction

This document describes the known exceptions to the functional and performance specifications to TI CMOS Radar Devices (AWR2544)

2 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of Radar / millimeter Wave sensor devices. Each of the Radar devices has one of the two prefixes: XAx or AWR2x (for example: XA2E44BGALTX). These prefixes represent evolutionary stages of product development from engineering prototypes (XAx) through fully qualified production devices (AWR2x).

Device development evolutionary flow:

XAx — Experimental device that is not necessarily representative of the final device's electrical specifications and may not use production assembly flow.

AWR2x — Production version of the silicon die that is fully qualified.

XAx devices are shipped with the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

Texas Instruments recommends that these devices not to be used in any production system as their expected end –use failure rate is still undefined.

3 Device Markings

Figure 3-1 shows an example of the AWR2544 Radar Device's package symbolization.

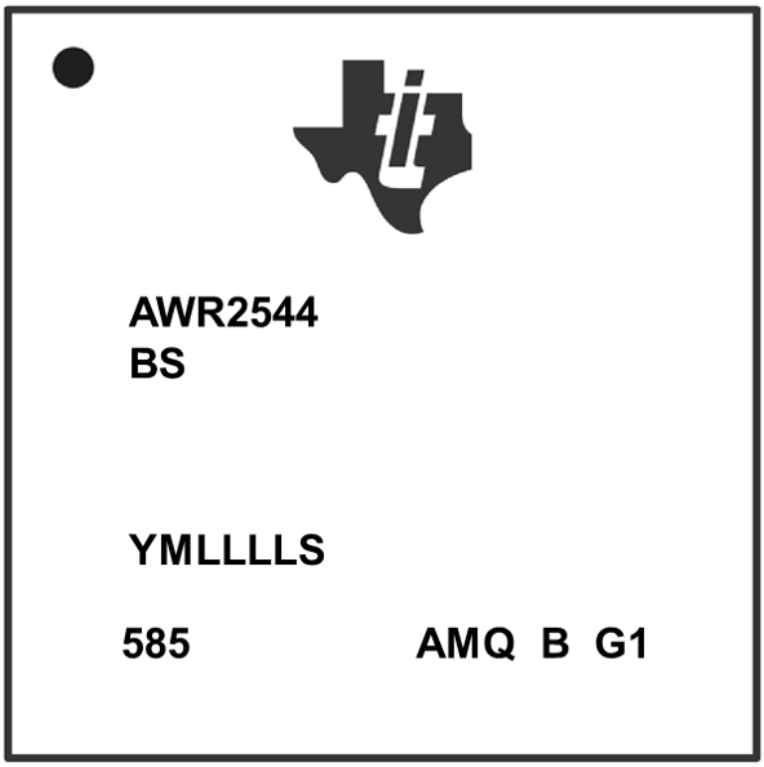

Figure 3-1. Example of Device Part Markings

This identifying number contains the following information:

- **Line 1:** Device Number
- **Line 2:** Safety Level and Security Grade
- **Line 3:** Lot Trace Code
  - YM = Year/Month Date Code
  - Z = Secondary Site Code
  - LLLL = Assembly Lot Code
  - S = Primary Site Code
- **Line 4:**
  - 585 = Device Identifier
  - AMQ = Package Identifier
  - G1 = "Green" Package Build (must be underlined)

4 Advisory to Silicon Variant / Revision Map

Table 4-1. Silicon Variant / Revision Map

ADVISORY NUMBER	ADVISORY TITLE	AWR2544
MAIN SUBSYSTEM		
MSS#25	Debugger May Display Unpredictable Data in the Memory Browser Window if a System Reset Occurs	X
MSS#27	MibSPI in Peripheral Mode in 3- or 4-Pin Communication Transmits Data Incorrectly for Slow SPICLK Frequencies and for Clock Phase = 1	X
MSS#28	A Data Length Error is Generated Repeatedly in Peripheral Mode When IO Loopback is Enabled	X
MSS#29	Spurious RX DMA REQ From a Peripheral Mode MibSPI	X
MSS#30	MibSPI RX RAM RXEMPTY Bit Does Not Get Cleared After Reading	X
MSS#33	MibSPI RAM ECC is Not Read Correctly in DIAG Mode	X
MSS#40	Any EDMA Transfer That Spans ACCEL_MEM1 +ACCEL_MEM2 Memories of Hardware Accelerator May Result In Data Corruption Without Any Notification Of Error From The SoC	X
MSS#49	Issues seen in potential interoperability with receiver supporting only Strict Alignment User Flow Control Stripping during Overflow message transmission in Aurora 64B/66B Protocol	X
MSS#52	DSS L2 Parity Issue: When DSP sends out an access beyond configured memory size	X
MSS#53	Incorrect behavior seen when context switch happens in the last parameter-set in HWA 2.0	X
MSS#54	Aurora TX UDP size<=4 is invalid	X
MSS#55	PMIC CLKOUT dithering in chirp-to-chirp staircase mode not supported	X
MSS#56	CR4 STC Boot Monitor Failure	X
MSS#57	Loss of data observed on Flush/Marker or completion of packet over MDO interface.	X
MSS#58	ePWM: Glitch during Chopper mode of operation	X
MSS#59	CRC: CRC 8-bit data width and CRC8-SAE-J1850 and CRC8-H2F possible use in CAN module is not supported	X
MSS#60	Mismatch in Read and Write address for 6-internal registers of PCR	X
MSS#61	Data aborts seen while access made to last 24 bytes of the configured MPU region and cache is enabled	X
MSS#62	HWA hangs when using back to back FFT3X paramsets	X
MSS#63	Potential L2 Cache Corruption During Block Coherence operations issue	X
MSS#64	Single MFENCE Issue	X
MSS#65	L2 Cache Corruption during block and global coherence operations issue	X
MSS#68	Potential corruption in FFT data for 0th BCNT in Real2X FFT Mode in HWA	X
MSS#71	Single bit ECC (error correction) mechanism can cause an incorrect memory update	X
ANALOG / MILLIMETER WAVE		
ANA#12A	Second Harmonic (HD2) Present in the Receiver	X
ANA#37A	High RX gain droop across LO frequency	X
ANA#39	HPF cutoff frequency 2800kHz configuration can result in incorrect RX IFA gains and filter corner frequencies	X
ANA#43	Errors seen in Synthesizer Frequency Live monitor	X
ANA#44	In 3.3V IO mode, back power is observed on the 1.8V rail from 3.3V rail	X
ANA#45	Spurs Caused due to Digital Activity	X
ANA#47	RX Spurs observed across RXs in Idle Channel Scenario	X
ANA#59	Spurs caused due to OSC_CLK_OUT_ETH activity	X

5 Known Design Exceptions to Functional Specifications

MSS#25 ***Debugger May Display Unpredictable Data in the Memory Browser Window if a System Reset Occurs***

Revision(s) Affected: AWR2544

Description:

If a system reset (nRST goes low) occurs while the debugger is performing an access on the system resource using system view, a peripheral error should be replied to the debugger. If the access was a read, instead the response might indicate that the access completed successfully and return unpredictable data.

This issue occurs under this condition: when a system reset is asserted (nRST low) on a specific cycle, while the debugger is completing an access on the system, using the system view. An example would be, when a debugger, like the CCS-IDE memory browser window, is refreshing content using the system view. This is not an issue for a CPU only reset and, this is not an issue during a power-on-reset (nPORRST) either.

Workaround(s):

Avoid performing debug reads and writes while the device might be in reset.

MSS#27 ***MibSPI in Peripheral Mode in 3- or 4-Pin Communication Transmits Data Incorrectly for Slow SPICLK Frequencies and for Clock Phase = 1***

Revision(s) Affected:

AWR2544

Description:

The MibSPI module, when configured in multibuffered peripheral mode with 3-functional pins (CLK, SIMO, SOMI) or 4-functional pins (CLK, SIMO, SOMI, nENA), could transmit incorrect data when all the following conditions are met:

- MibSPI module is configured in multibuffered mode,
- Module is configured to be a peripheral in the SPI communication,
- SPI communication is configured to be in 3-pin mode or 4-pin mode with nENA,
- Clock phase for SPICLK is 1, and
- SPICLK frequency is MSS_VCLK frequency / 12 or slower

Workaround(s):

The issue can be avoided by setting the CSHOLD bit in the control field of the TX RAM (Multi-Buffer RAM Transmit Data Register). The nCS is not used as a functional signal in this communication; hence, setting the CSHOLD bit does not cause any other effect on the SPI communication.

MSS#28 ***A Data Length Error is Generated Repeatedly in Peripheral Mode When IO Loopback is Enabled***

Revision(s) Affected:

AWR2544

Description:

When a DLEN error is created in Peripheral mode of the SPI using nSCS pins in IO Loopback Test mode, the SPI module re-transmits the data with the DLEN error instead of aborting the ongoing transfer and stopping. This is only an issue for an IOLPBK mode peripheral in Analog Loopback configuration, when the intentional error generation feature is triggered using CTRLDLENERR (IOLPBKTSTCR.16).

Workaround(s):

After the DLEN_ERR interrupt is detected in IOLPBK mode, disable the transfers by clearing the SPIEN (bit 24) in the SPIGCR1 register and then, re-enable the transfers by resetting the SPIEN bit.

MSS#29 ***Spurious RX DMA REQ From a Peripheral Mode MibSPI***

Revision(s) Affected:

AWR2544

Description:

A spurious DMA request could be generated even when the SPI Peripheral is not transferring data in the following condition sequence:

- The MIBSPI is configured in standard (non-multibuffered) SPI mode, as a Peripheral
- The DMAREQEN bit (SPIINT0.16) is set to enable DMA requests
- The Chip Select (nSCS) pin is in an active state, but no transfers are active
- The SPI is disabled by clearing the SPIEN (SPIGCR1.24) bit from '1' to '0'

The above sequence triggers a false request pulse on the Receive DMA Request as soon as the SPIEN bit is cleared from '1' to '0'.

Workaround(s):

Whenever disabling the SPI, by clearing the SPIEN bit (SPIGCR1.24), first clear the DMAREQEN bit (SPIINT0.16) to '0', and then, clear the SPIEN bit.

MSS#30 ***MibSPI RX RAM RXEMPTY bit Does Not Get Cleared After Reading***

Revision(s) Affected:

AWR2544

Description:

The RXEMPTY flag may not be auto-cleared after a CPU or DMA read when the following conditions are met:

- The TXFULL flag of the latest buffer that the sequencer read out of transmit RAM for the currently active transfer group is 0,
- A higher-priority transfer group interrupts the current transfer group and the sequencer starts to read the first buffer of the new transfer group from the transmit RAM, and
- Simultaneously, the Host (CPU/DMA) is reading out a receive RAM location that contains valid received data from the previous transfers.

Workaround(s):

Avoid transfer groups interrupting one another.

If dummy buffers are used in lower-priority transfer groups, select the appropriate "BUFMODE" for them (like, SKIP/DISABLED) unless, there is a specific need to use the "SUSPEND" mode.

MSS#33	*MibSPI RAM ECC is Not Read Correctly in DIAG Mode*
Revision(s) Affected:	AWR2544
Description:	A Read operation to the ECC address space of the MibSPI RAM in DIAG mode does not return the correct ECC value for the first 128 buffers, if the Extended Buffer support is implemented, but the Extended Mode is disabled for the particular MibSPI instance.
Workaround(s):	None

MSS#40 ***Any EDMA Transfer That Spans ACCEL_MEM1 +ACCEL_MEM2 Memories of Hardware Accelerator May Result In Data Corruption Without Any Notification Of Error From The SoC***

Revision(s) Affected: AWR2544

Description:

As per TPTC IP Spec, a Transfer request (TR) is supposed to access a single peripheral end point. ACCEL_MEM0/ACCEL_MEM1 memory banks of HWA are available via single peripheral point and ACCEL_MEM2/ ACCEL_MEM3 memory banks of HWA are available as another peripheral point (different from that of ACCEL_MEM0/ ACCEL_MEM1). Hence if a single TR is used to access a buffer spanning ACCEL_MEM1 and ACCEL_MEM2 memories of the HWA (i.e. a single buffer spanning 2 different peripheral points), the spec is not being adhered to. This errata is explicitly highlighting this spec requirement.

Note

The ACCEL_MEM1 and ACCEL_MEM2 memories are referred to as DSS_HWA_DMA0 and DSS_HWA_DMA1 at the SoC level.

Workaround(s):

Split the access into 2 TRs so that a single TR does not span ACCEL_MEM1 +ACCEL_MEM2. The 2 TRs can be chained.

MSS#49 ***Issues seen in potential interoperability with receiver supporting only Strict Alignment User Flow Control Stripping during Overflow message transmission in Aurora 64B/66B Protocol.***

Revisions Affected

AWR2544

Details

Measurement Data Output (MDO) is used to capture the transactions on the bus connected from different interfaces of the device and transmit outside over Aurora LVDS Interface (4-data lanes). MDO is comprised of a sniffer, FIFO, and an aggregator. The MDO sniffer module is responsible for monitoring the hardware interfaces in the chip and capturing the transactions on the bus which are within the configured addressing region of interest.

Data loss due to overflow can occur at the sniffer. This overflow information is sent as an interrupt to the CPU and the Aurora Tx IP. A User-Flow-Control (UFC) packet is generated by the Aurora TX IP in case of a data overflow condition in order to notify the user of this error condition. This is an error scenario and is not expected to occur in normal transfer functionality. At this stage, the data integrity is already compromised.

Aurora IP only supports UFC packet generation as per Section 6.6 of Aurora 64B/66B Protocol Specification, i.e. the UFC header block precedes the UFC data blocks. *Strict Alignment User Flow Control Stripping* (refer to Section 6.7 of Aurora 64B/66B Protocol Specification) is currently not supported.

Workaround

For MDO, the input data rate should be less than the output data rate so as to keep the effective data rate well within reasonable limits to avoid any overflow condition altogether.

Note

It is inadvisable for Aurora 64B/66B protocol to use *TOP_AURORA_TX:AURORA_TX_UFC_MSG_REQ* register to send UFC packets without overflow.

MSS#52 ***DSS L2 Parity Issue: When DSP sends out an access beyond configured memory size***

Revision(s) Affected: AWR2544

Description:

The DSP IP is sending out an access to the L2 memory for access beyond the configured DSP L2 memory size of 384 KB (reserved space access) i.e. beyond 0x8085 FFFC.

If parity is enabled, an L2 Parity error is observed for reads to the reserved locations beyond 0x80860000 - 0x8087FFFC

Note

Reserved Memory locations from 0x80860000 to 0x8087FFFC is accessible to read and write. Memory Locations from 0x80860000 to 0x8087FFFC are aliased at 0x80840000 to 0x8085FFFC and 0x80850000 to 0x8085FFFC is replicated at 0x80870000 to 0x8087FFFC, hence the actual L2RAM is of 384KB only.

Workaround(s):

Configuring the MPU : (L2MPPA24- L2MPPA31) to 0.

Write access to reserved space is blocked. No Aliasing & No L2 Parity Error. This ensures the data integrity of valid L2 Region is maintained.

Read access to reserved space still leads to L2 Parity Error (If Parity is enabled).

Debug access(Read & Write) are not blocked: Still leads to Aliasing + L2 Parity Error : Its not feasible to block the debug access despite configuring the MPPA registers for Protection enabled

Memory Protection Fault Address Register(0184 A000h:: L2MPFAR/0184 AC00h:: L1DMPFAR) are populated with the address which are blocked(beyond 384KB boundary in this case) & still accessed

Address(L2MPFAR/L1DMPFAR) & Status(L2MPFSR/L1DMPFSR) Registers are required to be cleared for the next read using Clear registers(L2MPFCR/L1DMPFCR) with values 1

Observations(Both when L1D Cache Enabled/Disabled)

For Read : MPU Protection Errors are observed on L1D with L1MPFAR registers populated with the blocked address access

For Write : MPU Protection Errors are observed on L2 with L2MPFAR registers populated with the blocked address access

MSS#53 ***Incorrect behavior seen when context switch happens in the last parameter-set in the Hardware Accelerator (HWA 2.1)***

Revision(s) Affected: AWR2544 Only

Description:

At the end of the last parameter-set of the last loop in low-priority context, if a context-switch happens to high priority, then an incorrect behavior is observed when returning back to low-priority. This incorrect behavior can manifest itself as a fresh (unintended) re-start of the low-priority loop once completed. Following are the erroneous conditions:

- CONTEXT_SW_EN or FORCED_CONTEXT_SW_EN set in the last paramset of the low priority thread.

Similarly, forced context switch (FORCED_CONTEXT_SW_EN) shouldn't be enabled in last paramset of high priority thread .

Workaround(s):

It is recommended to not enable context switch in the last parameter-set of the above mentioned conditions. In case, the last parameter-set has to have context switch enabled, user could add a dummy parameter-set with context switch disabled as the last parameter-set.

MSS#54 ***Aurora TX UDP size<=4 is invalid***

Revisions Affected: AWR2544

Description:

Aurora TX UDP size<=4 is invalid during transfer.

Valid UDP sizes for Aurora 8b/10b and Aurora 64b/66b are :

1. AURORA_TX_UDP_CONFIG_PACK_MODE_SEL = 0 (Bytes) : Valid Udp sizes - AURORA_TX_UDP_SIZE = 8, 12, 16, 20.....so on
2. AURORA_TX_UDP_CONFIG_PACK_MODE_SEL = 1 (TWP) : Valid Udp sizes - AURORA_TX_UDP_SIZE = 5, 6, 7, 8.....so on

Workaround:

It is recommended to use only the valid UDP sizes as described above.

MSS#55	*PMIC CLKOUT dithering in chirp-to-chirp staircase mode not supported*
Revision(s) Affected:	AWR2544
Description:	The PMIC CLKOUT has an option to add dithering to the clk frequency to reduce the impact of the clk spurs. The continuous mode of dithering is supported, while the chirp-to-chirp staircase mode of dithering is unsupported. This is because of the DFE reset not reaching the PMIC CLKOUT block.
Workaround(s):	It is recommended to use continuous dithering mode in PMIC CLKOUT.

MSS#56	*CR4 STC Boot Monitor Failure*
Revision(s) Affected:	AWR2544 ONLY
Description:	Cortex CR4 STC Boot Monitor Failure is observed in the device.
Workaround(s):	It is recommended to execute a sequence (MSS_CTRL:MSS_PBIST_KEY_RST[3:0] = 0) to clear the PBIST registers before starting CR4 (BSS) execution in the Secondary boot loader (SBL). Refer to the SBL example code provided by TI.

MSS#57 ***Loss of data observed on Flush/Marker or completion of packet over MDO interface.***

Revision(s) Affected: AWR2544

Description:

421-24008 : Data frames sent over MDO discards the last 6 bytes at the end of the frame

It is observed that data transfer over the MDO having data_size = 6 get dropped. To ensure complete data gets transferred, the data size needs to adhere to 4byte and 8byte aligned data. If not done, a loss of the last 6 bytes of data on Flush/Marker trigger or completion of packet could be observed.

Data Flow: EDMA -> SNIFFER -> FIFO --> AGGREGATOR -> STM -> TPIU -> AURORA TX.

With Data transfer using MSS_TPCC to MDO_DSS_FIFO having data size of 6 bytes results in the 6 bytes getting dropped. STM module has cxstm500_axislvif_write block which samples data based on WSTRB. There is no case inside the STM which can handle 6 bytes of incoming data.

Design Limitation in STM module to handle 6 bytes of data.

1, 2, 4, 8 bytes of data get handled. But 6 bytes results in data getting dropped.

Workaround(s):

It is recommended to include 2 dummy bytes during transfer to make the WSTRB handle 8 bytes.

MSS#58 *ePWM: Glitch during Chopper mode of operation*

Revision(s) Affected

AWRL6432 AWR2544

Details

During chopper mode operation, a glitch may be observed on the ePWMA and ePWMB output signals from the ePWM module.

Workaround

If the use case is impacted by a glitch, it is recommended to disable the PWM chopper control function by setting the LPRADAR:APP_PWM:PCCTL:CHPEN register bit to 0.

The below table shows the Register Address for above workaround.

Bits	Name	Address
0	LPRADAR:APP_PWM:PCCTL:CHPEN	0X57F7 FC3C

MSS#59	***CRC: CRC 8-bit data width and CRC8-SAE-J1850 and CRC8-H2F possible use in CAN module is not supported***
Revision(s) Affected	AWR2544
Details	1. 8-bit data width is not supported. Minimum data width supported is 16-bit. 2. CRC types CRC8-SAE-J1850 and CRC8-H2F are not supported.
Workaround	1. 16/32/64-bit data widths are supported. 2. It is recommended to not use the above mentioned unsupported polynomials.

MSS#60 *Mismatch in Read and Write address for 6-internal registers of PCR*

Revision(s) Affected

AWR2544

Details

Below is the set of common registers and their corresponding read-address offset and write-address offset for all PCRs in the Device

Register	Write Address offset	Read Address offset
PPROTSET_2	0x0000 0028	0x0000 002C
PPROTSET_3	0x0000 002C	0x0000 0040
PPROTCLR0	0x0000 0040	0x0000 0044
PPROTCLR1	0x0000 0044	0x0000 0048
PPROTCLR2	0x0000 0048	0x0000 004C
PPROTCLR3	0x0000 004C	0x0000 00260

Workaround

The above mentioned mapping to be used while performing any read-modify-writes or Read-back checks to these specific set of registers.

MSS#61 ***Data aborts seen while access made to last 24 bytes of the configured MPU region and cache is enabled.***

Revision(s) Affected

AWR2544

Details

When R5F performs access to a byte or word in the cacheable region, the access from cache is 32bytes long (One cache line size) with the starting address being the critical word being fetched.

The MPU assumes (Incorrectly) that the end address of the ongoing transaction to be Critical word + 32Bytes and compares this with the end address programmed in the MPU. MPU treats this as access violation and faults the transaction (Ex : 0x701FFFF8 + 32 byte = 0x7020 0018 > 0x70FF FFFF).

This issue is not applicable if MPU regions are marked as non-cacheable.

Workaround

If Cache is enabled, do not have any data in the last 32Bytes of the MPU region.

MSS#62	***HWA hangs when using back to back FFT3X paramsets***
Revision(s) Affected	AWR2544
Details	If FFT3X is enabled in back to back paramsets or as first and last paramsets of a loop, the HWA state machine hangs after the 1st FFT3X paramset is executed without raising any param done interrupt.
Workaround	Use any paramset with FFT3X disabled before using a paramset with FFT3X enabled (users may also go for a No Operation paramset with ACCEL_MODE = 0b111)

MSS#63 ***Potential L2 Cache Corruption During Block Coherence operations issue***

Revision(s) Affected:

AWR2544

Description:

A potential L2 cache corruption issue during block coherence operations has been identified. Under a specific set of circumstances, L1D or L2 block coherence operations can cause L2 cache corruption. The problem arises when the following four actions happen back to back in the same L2 set:

1. L1D write miss
2. Victim writeback due to block coherence operations
3. Write allocate for some address
4. Read or write allocate for some address

This issue applies to all the block coherence operations listed below:

- L1D writeback
- L1D invalidate
- L1D writeback with invalidate
- L2 writeback
- L2 invalidate
- L2 writeback with invalidate

Workaround(s):

The workaround requires that the memory system be idle during the block coherence operations. Hence programs must wait for block coherence operations to complete before continuing. This applies to L1D and L2 memory block coherence operations. To issue a block coherence operation follow the sequence below.

1. Disable interrupts.
2. Write the starting address to the corresponding BAR register.
3. Write the word count to the corresponding WC register.
4. Wait for completion by one of the following methods:
    a. Issue an MFENCE instruction (preferred).
    b. Poll the WC register until the word count field reads as 0.
5. Perform 16 NOPs.
6. Restore interrupts.

For further information about the cache control registers (BAR and WC) see the TMS320C66x DSP CorePac User Guide (SPRUGW0). The MFENCE instruction is new to the C66x DSP. It stalls the DSP until all outstanding memory operations complete. For further information about the MFENCE instruction, see the C66x DSP and Instruction Set Reference Guide (SPRUGH7).

MSS#64 ***L2 Cache Corruption during block and global coherence operations issue***

Revision(s) Affected:

AWR2544

Description:

Under a specific set of circumstances, L1D or L2 block and global coherence operations can cause L2 cache corruption. The problem arises when the following four actions happen back-to-back in the same L2 set:

1. L1D write miss
2. Invalidate or writeback-with-invalidate due to block and global coherence operations
3. Write allocate for some address
4. Read or write allocate for some address

This issue applies to all the block and global coherence operations EXCEPT:

- L1D block writeback
- L1D global writeback
- L2 block writeback
- L2 global writeback

See also: Advisory 63 - Potential L2 Cache Corruption During Block Coherence Operations Issue

Workaround(s):

Generic Workaround:

The workarounds below are very generic and may have performance impacts. Customers are requested to understand the application and see which one suits them better.

Workaround #1:

This workaround requires that the memory system be idle during the block and global coherence operations. Hence programs must wait for block and global coherence operations to complete before continuing. This applies to L1D and L2 memory block and global coherence operations.

To issue a block coherence operation, follow the sequence below:

1. Disable interrupts.
2. Write the starting address to the corresponding BAR register.
3. Write the word count to the corresponding WC register
4. Wait for completion by one of the following methods:
    a. Issue an MFENCE instruction (preferred)
    b. Poll the WC register until the word count field reads as 0
5. Perform 16 NOPs
6. Restore interrupts

To issue a global coherence operation, follow the sequence below:

1. Disable interrupts.
2. Write 1 to the corresponding global coherence register (L1DINV, L1DWBINV, L2DINV, and L2DWBINV)
3. Wait for completion by one of the following methods
    a. Issue an MFENCE instruction (preferred)
    b. Poll the corresponding global coherence register (L1DINV, L1DWBINV, L2DINV, and L2DWBINV) until the bit [0] field reads as 0
4. Perform 16 NOPs

MSS#64 (continued) ***L2 Cache Corruption during block and global coherence operations issue***

5. Restore interrupts

Workaround #2:

This workaround is also generic, but will allow CPU traffic to go on in parallel with cache coherence operations. To issue a block coherence operation, follow the sequence below:

1. Issue a MFENCE command.
2. Freeze L1D cache
3. Start L1D WBINV
4. Restart CPU traffic (CPU operations happen in parallel with WBINV and do not need to wait for cache coherency operation to complete)
5. Poll the WC register until the word count field reads as 0.
6. WBINV completes when word count field reads 0
7. Issue an MFENCE command.
8. Unfreeze L1D cache.

For more information about the cache control registers (BAR, WC, L1DINV, L1DWBINV, L2DINV, and L2DWBINV) see the TMS320C66x DSP CorePac User Guide (SPRUGW0). The MFENCE instruction is new to the C66x DSP. It stalls the DSP until all outstanding memory operations complete. For further information about the MFENCE instruction, see the C66x DSP and Instruction Set Reference Guide (SPRUGH7).

MSS#65 ***Single MFENCE Issue***

Revision(s) Affected:

AWR2544

Description:

The MFENCE instruction is used to stall the instruction fetch pipeline until the completion of all CPU-triggered memory transactions.

Under very particular circumstances, MFENCE may allow the transaction after the MFENCE to proceed before the preceding STORE completes.

For example,

1. STORE_A
2. MFENCE
3. TRANSACTION_B

The MFENCE implementation stalls the CPU until the memory system asserts that there are no transactions "in flight," i.e. it is idle. This prevents the CPU from proceeding to TRANSACTION_B before STORE_A completes. A small window exists where the memory system prematurely asserts that it is idle when STORE_A moves from L1D to L2 when it is otherwise idle. This can cause incorrect program behavior if TRANSACTION_B must occur strictly after STORE_A. For example, suppose STORE_A writes to DDR3, and TRANSACTION_B triggers an EDMA which reads the location written by STORE_A. MFENCE should guarantee that STORE_A commits before the EDMA executes, so that the EDMA sees the updated value. Due to the issue in this advisory, TRANSACTION_B could trigger the EDMA before STORE_A commits, so that the EDMA sees stale data.

Workaround(s):

Replace a single MFENCE with two MFENCEs back to back. This remedies the issue by resuming the stall in the case where the memory system prematurely indicated that it was idle when STORE_A passed from L1D to L2.

1. STORE_A
2. MFENCE
3. MFENCE
4. TRANSACTION_B

Note on coherence operations: For the following advisory, double MFENCE can also be used as a workaround in addition to the workarounds already listed:

- Advisory 63- Potential L2 Cache Corruption During Block Coherence Operations Issue
- Advisory 64- L2 Cache Corruption During Block and Global Coherence Operations Issue

Note that the second advisory listed above does not cover L1D and L2 block and global writebacks. If L1D and L2 block and global writebacks are followed by an MFENCE and a transaction that depends on the completion of the writebacks, the double MFENCE workaround should be used.

Please also note that the current release of the MCSDK software package does not include this workaround. It will be included in a future release

Note

Special Considerations for Trace. When trace generation is expected through software that includes the use of MFENCE, there are additional requirements for the workaround. Trace generation for MFENCE requires that every occurrence of the MFENCE instruction be followed with a NOP and a MARK instruction. The workaround is described in this white paper.

MSS#68	***Potential corruption in FFT data for 0th BCNT in Real2X FFT Mode in HWA***
Revision(s) Affected:	AWR2544
Description:	If Real2X FFT mode is enabled in the FFT parameter set, and FFT parameter set is used in combination with other parameter sets with dynamic clock gating disabled, it may produce faulty data for the 0th BCNT.
Workaround(s):	Enable dynamic clock gating in the HWA MMR space to reduce the probability of data corruption OR disable Real2X mode and run in Real1X mode.

MSS#71 ***Single bit ECC (error correction) mechanism can cause an incorrect memory update***

Revision(s) Affected: AWR2544

Description:

Note

The issue was uncovered during the debug of an incorrect memory access sequence in simulations. Till date there are no such issues reported in-field / deployment scenarios by any of our customers.

In the uncommon occurrences of single bit upset events on below tabulated memory ranges in the SoC, under a specific memory access sequence combination, the single bit error correction mechanism can cause an incorrect memory update.

The RAM memories on AWR294x are ECC protected with a Single bit Error Correction, Double bit Error Detection (SECDED) mechanism. On the occurrence of a specific sequence of events, the single bit error correction mechanism can cause an incorrect memory update.

For the issue to cause an impact to the application, all the below conditions must satisfy

- Random hardware faults, due to environmental conditions or other factors, leading to a single bit upset events occur, AND
- The Single bit upset event affects the impacted memory ranges, AND
- A read or partial-write access to the memory location with single bit error occurs (leading to the single bit error correction mechanism kicking in), AND
- A specific memory access sequence combination occurs after the single bit error correction happens, AND
- The incorrect memory update by the error correction mechanism is critical enough to impact the application program flow and is undetected by other safety mechanisms.

The following access combination (Conditions 3 and 4 above) to the impacted memory range after the single bit error correction happens can cause the issue.

- Read / Partial write access (from/to the location A with SEC) → (Followed by) Full write (to one or more memory locations in the same memory range) → (Followed by) Partial write (to any other location in the same memory range) : leads to incorrect update to last full-write location.
- Partial write access (to the location A with SEC) → (Followed by) Partial write (to any other location in the same memory range) : leads to incorrect update to location A.

Note

The issue doesn't occur for all other combinations of memory access sequence combinations.

Workaround(s):

The single bit upset events are uncommon with lower probability of occurrence.

- The scenario must lead to single bit errors alone. Double bit errors are only detected and on double bit errors, depending on the criticality, the device is taken to safe state.

Partial write memory accesses (needed to cause the issue) are limited as

- Cached memory ranges do not lead to partial write accesses as cache lines writes are always full writes.

MSS#71 (continued) ***Single bit ECC (error correction) mechanism can cause an incorrect memory update***

  - Ex., MSS L2 memories
- Code sections are read only (hence the entire code section accesses do not satisfy the conditions to cause the issue).
  - Ex., MSS L2 memories
- Impacted memories with partial write accesses can have other safety mechanisms that can detect or avoid such random errors.
  - Higher level processing algorithms of Radar data cube have built in outlier rejection capabilities due to tracking functions (temporal and logical monitoring).
    - Ex., DSS L3
  - Information redundancy techniques may be used on impact memories like Mailbox to detect errors.
    - Ex., Mailbox memories

In the impacted memory ranges, identify if there are possibilities of partial memory write accesses. Decide on the criticality for the need to take cation on such identified memory ranges with partial writes. Following are the possible courses of actions:

- If single bit upset events are unlikely in the operating environment.
- If there are other safety mechanisms that can detect or avoid such spurious random errors.

- Action: One or more of the following options can be considered
  - Avoid the partial write access pattern to those memory ranges.
  - Re-initialise the impacted memory bank on single bit memory correction event.
  - Treat the single bit memory correction event as an un-correctable error and enter safe state.
    - This does not impact the Functional safety detectability claims and may impact the availability in the event of such single bit upset occurrence.

Refer below table for memory range and its corresponding ESM line & ECC aggregator bit if action (2-b-ii) needs to be taken.

This table includes only impacted memory list and corresponding details regarding

Memory Name	Start address	End Address	ESM Line	ECC Aggregator Status bit
DSS L3 Bank0	0x88000000	0x880BFFFF	DSS_ESM:: GROUP1 Line No-92	DSS_ECC_AGG::SEC_STATUS_REG0:: DSS_L3RAM0_PEND
DSS L3 Bank1	0x8800C000	0x8817FFFF	DSS_ESM:: GROUP1 Line No-92	DSS_ECC_AGG::SEC_STATUS_REG0:: DSS_L3RAM1_PEND
DSS L3 Bank2	0x88180000	0x881FFFFF	DSS_ESM:: GROUP1 Line No-92	DSS_ECC_AGG::SEC_STATUS_REG0:: DSS_L3RAM2_PEND
DSS L3 Bank3	0x88200000	0x8827FFFF	DSS_ESM:: GROUP1 Line No-92	DSS_ECC_AGG::SEC_STATUS_REG0:: DSS_L3RAM3_PEND
MSS L2 Bank0	0xC0200000	0xC027FFFF	MSS_ESM:: GROUP1 Line No-18	MSS_ECC_AGG_MSS::SEC_STATUS_REG0:: MSS_L2SLV0_PEND

MSS#71 (continued) ***Single bit ECC (error correction) mechanism can cause an incorrect memory update***

Memory Name	Start address	End Address	ESM Line	ECC Aggregator Status bit
MSS L2 Bank1	0xC0280000	0xC02EFFFF	MSS_ESM:: GROUP1 Line No-18	MSS_ECC_AGG_MSS::SEC_STATUS_REG0:: MSS_L2SLV1_PEND
MSS Mailbox	0xC5000000	0xC5001FFF	MSS_ESM:: GROUP1 Line No-18	MSS_ECC_AGG_MSS::SEC_STATUS_REG0:: MSS_MBOX_PEND
MSS_RETRAM	0xC5010000	0xC50107FF	MSS_ESM:: GROUP1 Line No-18	MSS_ECC_AGG_MSS::SEC_STATUS_REG0:: MSS_RETRAM_PEND
DSS Mailbox	0x83100000	0x83100FFF	DSS_ESM:: GROUP1 Line No-92	DSS_ECC_AGG::SEC_STATUS_REG0:: DSS_MAILBOX_PEND

Note

MSS_L2 address captured above is from DSS and EDMA addressing View. MSS_L2_BANK0 and MSS_L2_BAK1 address view from MSS-R5 is 0x10200000-0x1027FFFF. and 0x10280000-0x102EFFFF respectively

Other memories that are not utilized by the application but used by the BSS, such as BSS_Mailbox and BSS_Static_RAM, are also affected by this errata

- The BSS mailbox is primarily used for communication between the BSS and MSS/DSS using mmWaveLink, following a message protocol that incorporates CRC for data integrity. Using CRC during message exchanges over the BSS mailbox reduces the risk associated with this memory.
- When a fault occurs (in this case, an ECC SEC), BSS sends an ESM Fault Asynchronous event message to the MSS/DSS as a notification. The application must read the b20:ECC_AGG_SEC_ERROR from AWR_AE_RF_ADV_ESMFAULT_STATUS_SB async-event from the BSS. Treat this single-bit memory correction event as an uncorrectable error and enter to a safe state.
  - This workaround is only valid if the application uses BSS Patch from DFP version 2.4.14 or earlier

ANA#12A ***Second Harmonic (HD2) Present in the Receiver***

Revision(s) Affected:

AWR2544

Description:

There is a finite isolation between the RF pins/package and the FMCW synthesizer. This can create spurious tones at the synthesizer output and lead to appearance of 2nd order harmonics and inter-modulations of expected IF frequencies at RX ADC output. The amplitude of the 2nd harmonic could be as high as -55 dBc, referenced to the power level of the intended tone at the LNA input.

Workaround(s):

No workaround available at this time. However, in many typical radar use cases the HD2 does not affect the system performance due to two reasons:

1. Since the HD2 comes from a coupling to the LO signal, there is an inherent suppression of the HD2 level due to the self-mixing effect (that is, phase noise and phase spur suppression effect at the mixer).
2. In real-life scenarios there is often a double-bounce effect of the radar signal reflected from the target, which leads to a ghost object at twice the distance of the actual object. This effect is often indistinguishable from the effect of HD2 itself.

ANA#37A	*High RX gain droop across LO frequency*
Revisions Affected	AWR2544
Details	RX gain droop is ~4.5dB across the full operating frequency range of the device.
Workaround	Negligible impact on system performance since there is an insignificant impact on noise figure due to the gain droop.

ANA#39	***HPF cutoff frequency 2800kHz configuration can result in incorrect RX IFA gains and filter corner frequencies.***
Revisions Affected	AWR2544
Details	The analog IF stages include a second order high pass filter that can be configured to the following -6dB corner frequencies : 300, 350, 700, 1400, 2800 kHz. Out of these, HPF cutoff frequency 2800kHz configuration can result in incorrect RX IFA gains and filter corner frequencies.
Workaround	Use of 2800kHz cutoff configuration is not recommended.

ANA#43	***Errors seen in Synthesizer Frequency Live monitor***
Revision(s) Affected:	AWR2544
Description:	Large errors are seen in excess of 20 MHz in the Synthesizer Frequency Live monitor after 100C for ramp configurations between 80.5GHz to 81GHz with a slope > 50MHz/us.
Workaround(s):	For slopes >50MHz/us, it is recommended to utilize chirps under 80.5GHz.

ANA#44	***In 3.3V IO mode, back power is observed on the 1.8V rail from 3.3V rail***
Revision(s) Affected:	AWR2544
Description:	When the 3.3V power rail comes up and 1.8V has not been supplied yet, there is a voltage rise seen on the 1.8V VIOIN rail due to the leakage path within the IO cell.
Workaround(s):	It is recommended to use the following workarounds: 1. Use appropriate Supply Sequencing: Supply 1.8V first and then 3.3V. 2. In case the PMIC fails to powerup due to sensing an existing voltage at its output, this voltage detection scheme in the PMIC should be disabled.

ANA#45 ***Spurs Caused due to Digital Activity***

Revision(s) Affected:

AWR2544

Description:

Digital filtering activity can potentially couple to analog circuits leading to spurs in the LO, which may also be seen in the Rx data. Such a spur in the Rx data would be seen at the spur frequency offset around a strong object.

Following are the different spurs that can potentially be observed:

1. Spurs at (2Fs-40) MHz IF frequency for sampling rate +/-0.5Msps around 20 Msps.
2. Spurs at (2Fs-60) MHz IF frequency for sampling rate +/-0.5Msps around 30 Msps.
3. Spurs at (4Fs-140) MHz IF frequency for sampling rate +/-0.3Msps around 35 Msps.
4. Spurs at (4Fs-100) MHz IF frequency for sampling rates in the range 22 to 23.5 Msps and 26.5 to 28 Msps

[Fs=Profile Sampling Rate]

Workaround(s):

The user should avoid sampling rates in the range mentioned above or use exactly the center of the sampling rate range (so that spur is at 0 Hz).

ANA#47 ***RX Spurs observed across RXs in Idle Channel Scenario***

Revision(s) Affected: AWR2544

Description:

In scenarios of no object being present, or a very weak object being present in the vicinity, the sigma delta ADC output could have spurs in the RX spectrum. This is observed only for low RX gain settings. The spur frequency could vary across RX channels. In presence of a real object, this would not be observed.

Workaround(s):

- *Workaround#1* : Use higher rx gain (>40dB) in these situations.
- *Workaround#2* : Idle channel spur is spread across all doppler bins in 2DFFT at the spur range bin. While detecting peaks in 2D-FFT, users can apply 2D neighborhood peak search (e.g. 2D CFAR-CA), which compares the level with all surrounding bins. This can help avoid detection of idle channel spur as ghost object.

ANA#59 ***Spurs caused due to OSC_CLK_OUT_ETH activity***

Revision(s) Affected:

AWR2544

Description:

Enabling OSC_CLK_OUT_ETH with XTAL/2 frequency option can result in weak spurs at XTAL/2 IF offset due to coupling of Ethernet clock signal with LO signal. In presence of strong real target, a potential ghost target would be present in the RX data at IF = (XTAL/2 - real target IF frequency).

Workaround(s):

ONE OR MORE of the following workarounds can be considered:

1. Avoid OSC_CLK_OUT_ETH = XTAL/2 option and instead use XTAL/1 option. Implement any necessary /2 division outside AWR2544 chip, e.g. inside Ethernet PHY chip, if that option is available.
2. Use IF BW < XTAL/4 to avoid any in-band ghost targets in presence of strong real targets.
3. If using IF BW > XTAL/4, after detecting strong real targets, use stricter detection threshold at IF offsets (XTAL/2 - real target IF frequency) to avoid detecting these OSC_CLK_OUT_ETH induced ghost targets.
4. Reduce drive strength of OSC_CLK_OUT_ETH to reduce spur level. Expect 3dB reduction in spur level for 0.5x drive strength.
5. Use OSC_CLK_OUT instead of OSC_CLK_OUT_ETH

Trademarks

All trademarks are the property of their respective owners.

6 Revision History

Changes from December 24, 2024 to October 1, 2025 (from Revision * (December 2024) to Revision A (October 2025)) **Page**

- *Advisory to Silicon Variant / Revision Map:* Added MSS#68 and MSS#71 advisory in the "Main Subsystem"..4
- *Advisory to Silicon Variant / Revision Map:* Removed ANA#46..... 4
- *CRC: CRC 8-bit data width and CRC8-SAE-J1850 and CRC8-H2F possible use in CAN module is not supported*: Details and workaround rephrased.....20
- *MSS#68:* Potential Corruption in FFT data for 0th BCNT in Real 2X FFT Mode in HWA 28
- *MSS#71:* Single-bit ECC (error correction) mechanism can cause an incorrect memory update.....29
- *ANA#59:* Updated effected device to AWR2544. Changed ANA48 to ANA59 for consistency..... 39

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025