Errata

AM62x Processor Silicon Revision 1.0

ABSTRACT

This document describes the known exceptions to the functional specifications (advisories). This document may also contain usage notes. Usage notes describe situations where the device's behavior may not match presumed or documented behavior. This may include behaviors that affect device performance or functional correctness.

Table of Contents

1 Usage Notes and Advisories Matrices

lists all usage notes and the applicable silicon revision(s). lists all advisories, modules affected, and the applicable silicon revision(s).

Table 1-1. Usage Notes Matrix

ID	DESCRIPTION	SILICON REVISIONS AFFECTED
		AM62x 1.0

Table 1-2. Advisories Matrix

MODULE	DESCRIPTION	SILICON REVISIONS AFFECTED
		AM62x 1.0
Boot	i2328 — Boot: USB MSC boots intermittently	YES
DDR	i2232 — DDR: Controller postpones more than allowed refreshes after frequency change	YES
DDR	i2244 — DDR: Valid stop value must be defined for write DQ VREF training	YES
DSS	i2097 — DSS: Disabling a layer connected to Overlay may result in synclost during the next frame	YES
ECC_AGGR	i2049 — ECC_AGGR: Potential IP Clockstop/Reset Sequence Hang due to Pending ECC Aggregator Interrupts	YES
Internal Diagnostic Modules	i2103 — Incorrect Reporting of ECC_GRP, ECC_BIT and ECC_TYPE Information for Functional Safety Errors	YES
Interrupt Aggregator	i2196 — IA: Potential deadlock scenarios in IA	YES
OSPI	i2189 — OSPI: Controller PHY Tuning Algorithm	YES
RAT	i2062 — RAT: Error Interrupt Triggered Even When Error Logging Disable Is Set	YES
RTC	i2327 — RTC: Hardware wakeup event limitation	YES
USART	i2310 — USART: Erroneous clear/trigger of timeout interrupt	YES
USART	i2311 — USART Spurious DMA Interrupts	YES
USB	i2134 — USB: 2.0 Compliance Receive Sensitivity Test Limitation	YES

1.1 Devices Supported

This document supports the following devices:

- AM62x

Reference documents for the supported devices are:

- AM62x Processors Technical Reference Manual (SPRUIV7)
- AM62x Processors Data Sheet (SPRSP58)

2 Silicon Usage Notes and Advisories

This section lists the usage notes and advisories for this silicon revision.

2.1 Silicon Usage Notes

2.2 Silicon Advisories

i2049 ***ECC_AGGR: Potential IP Clockstop/Reset Sequence Hang due to Pending ECC Aggregator Interrupts***

Details: The ECC Aggregator module is used to aggregate safety error occurrences (which are rare) and generate interrupts to notify software. The ECC Aggregator provides software control over the enabling/disabling and clearing of safety errors interrupts.

When software is performing a clockstop/reset sequence on an IP, the sequence can potentially not complete because the IP's associated ECC Aggregator instance is not idle. The ECC Aggregator idle status is dependent upon any pending safety error interrupts either enabled or disabled, which have not been cleared by software. As a result, the IP's clockstop/reset sequence may never complete (hang) if there are any pending safety errors interrupts that remain uncleared.

Workaround(s): General Note:

Clockstopping the ECC Aggregator is not supported in functional safety use-cases.

Software should use the following workaround for non-functional safety use-cases:

1. Enable all ECC Aggregator interrupts for the IP
2. Service and clear all Pending interrupts
3. Step 3:
   a. Disable all interrupt sources to the ECC Aggregator, followed by performing Clockstop/reset sequence.
   b. Perform Clockstop/reset sequence, while continuing to service/clear pending interrupts.

Due to interrupts being external stimuli, software has two options for step 3:

1. Disable all interrupt sources (EDC CTRL checkers) that can generate pending ECC_AGGR interrupts prior to performing the clockstop/reset sequence
2. Continue to service/clear pending interrupts that occur while performing the clkstop/reset sequence. The sequence would proceed when all interrupts are cleared.

Software in general may need to detect pending interrupts that continuously fire during this entire sequence (ex. in the case of a stuck-at fault scenario), and disable their associated EDC CTRL safety checkers to allow the clockstop/reset sequence to progress towards completion.

i2062 ***RAT: Error Interrupt Triggered Even When Error Logging Disable Is Set***

Details: If the RAT error logging is programmed to disable logging and enable interrupts, then an error will incorrectly trigger an interrupt but the error log registers will correctly not be updated. The error interrupt should not have been generated.

Workaround(s): If the RAT error logging is disabled, then the error interrupt should also be disabled by software.

i2097 ***DSS: Disabling a Layer Connected to Overlay May Result in Synclost During the Next Frame***

Details: Disabling a layer (for example VID1) connected to an OVR (that is toggling DSS_VID_ATTRIBUTESx[0] ENABLE from 1 to 0) may result in synclost during the next

i2097 (continued) ***DSS: Disabling a Layer Connected to Overlay May Result in Synclost During the Next Frame***

frame. The synclost may result in a corrupted or blank frame (all pixel data sent out of DSS during the frame is 0x0). The occurrence of synclost is dependent on the timing of setting the GO bit (that is DSS_VP_CONTROL[5] GOBIT to 1) vis-à-vis the disabling of the layer. If the “disable layer” MMR write operation and “set GO bit” MMR write operation happens within the same frame boundary, no synclost occurs. If the operations happen across the frame boundary, then synclost occurs (for one frame). The design automatically recovers and returns to normal operation from the next frame after GO bit is set, see Figure 2-1.

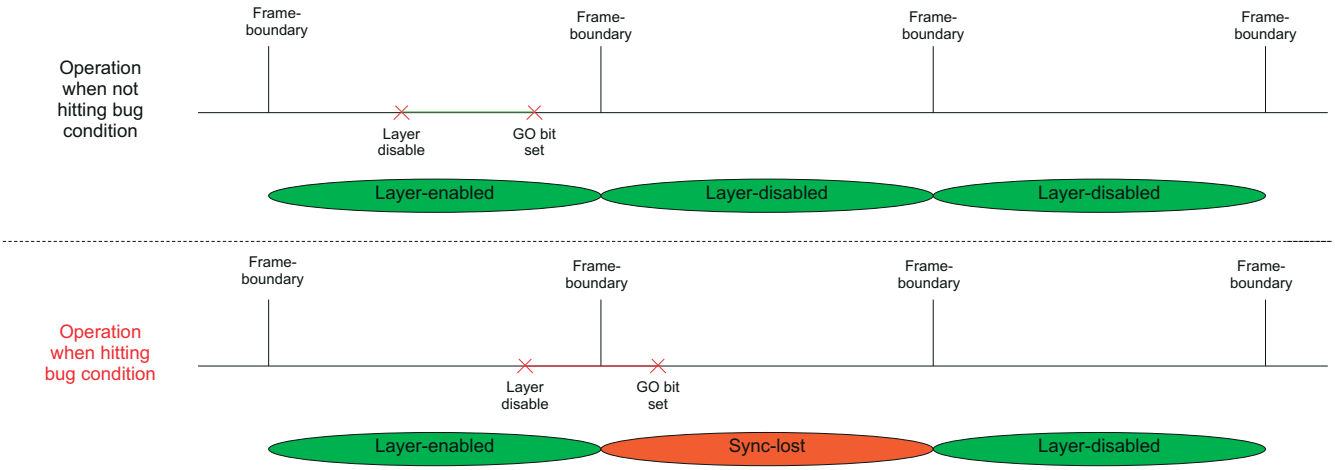

Figure 2-1. Bug Condition

Workaround(s): A simple software workaround exists. In the workaround, prior to disabling a layer on the OVR, it is moved to the “non-visible” area of the OVR (for example: DSS_OVR_ATTRIBUTES_x[17-6] POSX = max_value_of_posx or DSS_OVR_ATTRIBUTES_x[30-19] POSY = max_value_of_posy). This avoids the synclost when the layer is disabled.

A sample software workaround pseudo-code is shown on Figure 2-2. In this case, the regular “disable layer” MMR write operation and “set GO bit set” MMR write operation are replaced with macros which implement the software workaround.

```
macro disable_layer (overlay n , layer m)
set OVR[n].ATTRIBUTES2[m].POSX = posx_max;
set OVR[n].ATTRIBUTES2[m].POSY = posy_max;
global_ovr_layer_disable_tracker[n][m] = 1;
endmacro

macro set_go_bit (vp n)
if(|(global_ovr_layer_disable_tracker[n])//any bit set
{
   set VP[n].CONTROL.GOBIT  = 1;
   Wait for 10 DSS FUNC CLK cycles;
   for (i=0;i<NUM_LAYERS;i++)
   {
     if(global_ovr_layer_disable_tracker[n][i])
     {
       Clear OVR[n].ATTRIBUTES[i].ENABLE = 0;
       global_ovr_layer_disable_tracker[n][i] = 0;
     }
   }
}
set VP[n].CONTROL.GOBIT  = 1;
endmacro
```

- Replace layer disable MMR write operation with a macro which positions the layer to the non-visible area of the OVR
- Track which layers are disabled. This will be used while GO bit is set

- Replace GO bit set MMR write operation with this macro
- First, set GO Bit for the changes in “disable_layer” macro (and any other earlier changes) to take effect
- After the first GO bit set, few idle_cycles (10 DSS functional clock cycles) are necessary before we move to the second step

- In the second step, actually disable the layers based on the previously tracked information
- Set the GO bit for the second time for the disable of the layers to take effect

Figure 2-2. Workaround Pseudo-code

i2103 ***Internal Diagnostics Modules: Incorrect Reporting of ECC_GRP, ECC_BIT and ECC_TYPE Information for Functional Safety Errors***

Details:

For functional safety errors, the logged information - ECC_GRP, ECC_BIT, and ECC_TYPE in the Error Status Registers may be incorrect for certain safety checkers. This only applies to safety checkers that map to ECC_GRP = 0,15,31,47,63...(N*16-1). In the case for the DDR Bridge/Controller, the issue only applies to the safety checkers where ECC_GRP = 0,31,63...(N*32-1).

This issue affects all Internal Diagnostics Module instances and their sub-banks.

Note: The detection and interrupt signaling of these safety errors is unaffected. Only the logging of the aforementioned fields of the Error Status Registers are affected.

Workaround(s):

None. For these specific safety checkers, software is limited to knowing whether a correctable or uncorrectable error occurred and which Internal Diagnostics Module instance had the error (thus knowing the IP module), but not which exact safety checker encountered the error.

i2134 ***USB: 2.0 Compliance Receive Sensitivity Test Limitation***

Details:

Performing receive sensitivity tests (EL_16 and EL_17) as defined in the USB-IF USB 2.0 Electrical Compliance Test Specification may invoke the problem described in Advisory .

The issue was originally found while performing these tests using automation software, which increased USB signal amplitude while sending packets. The software was sweeping the amplitude from a value less than 100 mV to a value greater than 150 mV while verifying the device under test (DUT) NAK'd all packets below 100 mV and NAK'd no packets above 150 mV. However, increasing the amplitude through the squelch threshold while sending valid packets may lock the PHY as described in Advisory .

Workaround(s):

i2189 ***OSPI: Controller PHY Tuning Algorithm***

Details:

The OSPI controller uses a DQS signal to sample data when the PHY Module is enabled. However, there is an issue in the module which requires that this sample must occur within a window defined by the internal clock. Read operations are subject to external delays, which change with temperature. In order to guarantee valid reads at any temperature, a special tuning algorithm must be implemented which selects the most robust TX, RX, and Read Delay values.

Workaround(s):

The workaround for this bug is described in detail in the application note spract2 (link: https://www.ti.com/lit/spract2). To sample data under some PVT conditions, it is necessary to increment the Read Delay field to shift the internal clock sampling window. This allows sampling of the data anywhere within the data eye. However, this has these side effects:

1. PHY Pipeline mode must be enabled for all read operations. Because PHY Pipeline mode must be disabled for writes, reads and writes must be handled separately.
2. Hardware polling of the busy bit is broken when the workaround is in place, so SW polling must be used instead. Writes must occur through DMA accesses, within page boundaries, to prevent interruption from either the host or the flash device. Software must poll the busy bit between page writes. Alternatively, writes can be performed in non-PHY mode with hardware polling enabled.
3. STIG reads must be padded with extra bytes, and the received data must be right-shifted.

i2196 ***IA: Potential deadlock scenarios in IA***

Details:

The interrupt Aggregator (IA) has one main function, which is to convert events arriving on the Event Transport Lane (ETL) bus, can convert them to interrupt status bits which are used to generate level interrupts. The block that performed this function in IA version 1.0 was called the status event block.

In addition to the status event block, there are two other main processing blocks; the multicast event block, and the counted event block. The multicast block really functions as an event splitter. For every event it takes in, it can generate two output events. The counted event block is used to convert high frequency events into a readable count. It counts input events and generates output events on count transitions to/from 0 to/from non-zero count values. Unlike the status event block, the multicast and counted event blocks generate output ETL events that are then mapped to other processing blocks.

An issue was found after design that could cause the IA to deadlock. The issue occurs when event “loops” occur between these three processing blocks. It is possible to create a situation where a processing block can not output an event because the path is blocked, and since it can not output an event, it can not take any new input events. This inability to take input events prevents the output path from being able to unwind, and thus both paths remain blocked.

Workaround(s):

Figure 2-3 shows the conceptual block diagram of IA 1.0. Potential loops are avoided by adopting the policy of not allowing the counted event block to send events to the multicast block. This method was chosen because it is more common to split an event first, and then count one while sending the other elsewhere. With this path blocked by convention, it is not possible for a single event to visit any block more than once and thus not possible for paths to become blocked so long as the outputs remain unblocked.

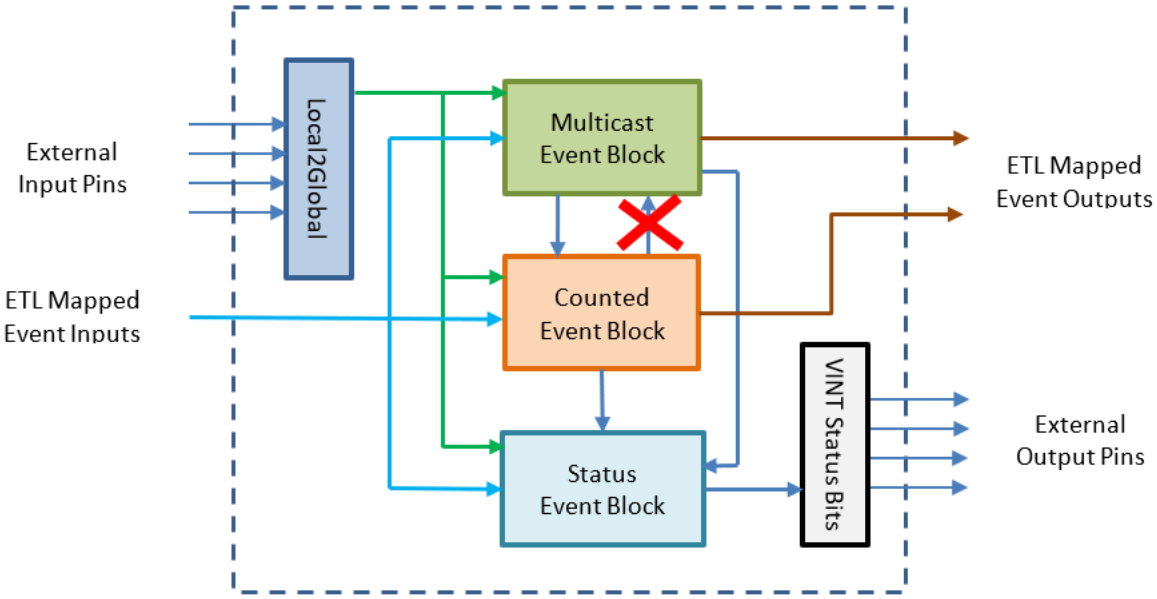

Figure 2-3. Interrupt Aggregator Version 1.0

By following the conventions outlined here, the system is safe from looping hazards that can create a deadlock scenario.

i2232 ***DDR: Controller postpones more than allowed refreshes after frequency change***

Details

When dynamically switching from a higher to lower clock frequency, the rolling window counters that control the postponing of refresh commands are not loaded correctly to scale to the lower clock frequency. This will result in controller postponing more refresh commands than allowed by the DRAM specification, thus violating refresh requirement for the DRAM.

Workaround

Workaround 1:Disable dynamic frequency change by programing DFS_ENABLE = 0

Workaround 2:If switching frequency, program the register field values based on the pseudo code listed below.Note that the controller requires AREF_*_THRESHOLD values to be programmed before triggering initialization. Their values cannot be changed during mission mode after initialization . Therefore, the value of these parameters must be the lowest of all values needed for every frequency change transition planned to be used.

```
if (old_freq/new_freq >= 7){
    if (PBR_EN==1) { // Per-bank refresh is enabled
        AREF_HIGH_THRESHOLD = 19
        AREF_NORM_THRESHOLD = 18
        AREF_PBR_CONT_EN_THRESHOLD = 17
        AREF_CMD_MAX_PER_TREF = 8
    }
    else { // Per-bank refresh is disabled
        AREF_HIGH_THRESHOLD = 18
        AREF_NORM_THRESHOLD = 17
        // AREF_PBR_CONT_EN_THRESHOLD <=== don't care, PBR not enabled
        AREF_CMD_MAX_PER_TREF = 8
    }
}
else {
    AREF_HIGH_THRESHOLD = 21
    AREF_NORM_THRESHOLD //<=== keep AREF_NORM_THRESHOLD < AREF_HIGH_THRESHOLD
    AREF_CMD_MAX_PER_TREF = 8
    if (PBR_EN==1) { // Per-bank refresh is enabled
    //keep AREF_PBR_CONT_EN_THRESHOLD<AREF_NORM_THRESHOLD<AREF_HIGH_THRESHOLD
        AREF_PBR_CONT_EN_THRESHOLD
    }
}
```

i2244 ***DDR: Valid stop value must be defined for write DQ VREF training***

Details The DDR PHY uses start, stop, and step-size values for write DQ VREF training. If the stop value is not equal to the start value + a multiple of the step-size, then the final VREF setting can go beyond the maximum VREF range, causing the training to hang.

Workaround Program the stop value as follows:

PI_WDQLVL_VREF_INITIAL_STOP = (multiple of PI_WDQLVL_VREF_INITIAL_STEPSIZE) + PI_WDQLVL_VREF_INITIAL_START

This workaround is implemented in the DDR Subsystem Register Configuration Tool v0.03.00 or later. See https://dev.ti.com/sysconfig for more details.

i2310 ***USART: Erroneous clear/trigger of timeout interrupt***

Details: The USART may erroneously clear or trigger the timeout interrupt when RHR/MSR/LSR registers are read.

Workaround(s):

For CPU use-case.

If the timeout interrupt is erroneously cleared:

-This is OK since the pending data inside the FIFO will retrigger the timeout interrupt

If timeout interrupt is erroneously set, and the FIFO is empty, use the following SW workaround to clear the interrupt:

- Set a high value of timeout counter in TIMEOUTH and TIMEOUTL registers

- Set EFR2 bit 6 to 1 to change timeout mode to periodic

- Read the IIR register to clear the interrupt

- Set EFR2 bit 6 back to 0 to change timeout mode back to the original mode

For DMA use-case.

If timeout interrupt is erroneously cleared:

-This is OK since the next periodic event will retrigger the timeout interrupt

-User must ensure that RX timeout behavior is in periodic mode by setting EFR2 bit6 to 1

If timeout interrupt is erroneously set:

-This will cause DMA to be torn down by the SW driver

-OK since next incoming data will cause SW to setup DMA again

i2311 ***USART Spurious DMA Interrupts***

Details: Spurious DMA interrupts may occur when DMA is used to access TX/RX FIFO with a non-power-of-2 trigger level in the TLR register.

Workaround(s):

Use power of 2 values for TX/RX FIFO trigger levels (1, 2, 4, 8, 16, and 32).

i2327 ***RTC: Hardware wakeup event limitation***

Details: The RTC hardware wakeup event cannot get used if software is unable to unlock the RTC within one second after the reset to the RTC is released.

i2327 (continued)	***RTC: Hardware wakeup event limitation***
	All other functionality of the RTC (eg, time of day) is not affected by this errata
Workaround(s):	None
i2328	***Boot: USB MSC boots intermittently***
Details:	USB MSC Host boot may fail due to a protocol timing violation present in the ROM USB device enumeration process. USB DFU boot is unaffected.
Workaround(s):	No workaround is available. Some USB MSC devices may tolerate this protocol violation and function as expected. Due to the internal component variability of broad-market MSC devices, a list of tolerant devices cannot be provided.

Trademarks

All trademarks are the property of their respective owners.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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 will 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 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.

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

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