MAX32665/MAX32666 ERRATA SHEET

Revision A1 Errata

The errata listed below describe situations where components of this revision perform differently than expected or differently than described in the data sheet. Analog Devices may, at its own discretion, take future steps to correct these errata when the opportunity to redesign the product presents itself. Prior to that, Analog Devices has determined the following potential workarounds that customers may want to consider when addressing one of the situations described below.

This errata sheet only applies to components of this revision. These components are branded on the topside of the package with a six-digit code in the form yywwRR, where yy and ww are two-digit numbers representing the year and work week of manufacture, respectively, and RR is the revision of the component. To obtain an errata sheet on other die revisions, visit the product webpage at www.analog.com/MAX32665.

1) SPIXF BUS IDLE FEATURE CANNOT BE DISABLED

Description:
The SPIXF bus idle feature cannot be disabled. The SPIXF will unexpectedly force the start of a new transaction, requiring resending of the SPI transaction header, if two consecutive reads of the cache both generate a cache miss. (14358)

Workaround:
Set the SPIXF_BUS_IDLE.busidle to 0x1 to approximate the operation of the SPIXF when the bus idle feature is disabled. This will have minimal effect on sequential code execution.

2) QSPI IN SLAVE MODE 0 OR MODE 2 DOES NOT OPERATE AS EXPECTED IF Tx FIFO IS FULL

Description:
When the QSPIn_SS0 signal transitions to its active state, the first bit of the first byte of the MISO signal is always 0 if the transaction starts when the Tx FIFO is full. (14354)

Workaround:
Ensure that the number of bytes in the Tx FIFO is always less than 32 (*QSPIn_DMA.tx_fifo_cnt* < *32*).

3) ECC FEATURE REMOVED FROM DATA SHEET

Description:
The ECC functionality has been defeatured from all revisions of the device due to the complexity of the workarounds. This information is provided for legacy use only and mention of the ECC feature was removed starting with revision 7 of the data sheet. (14477)

Workaround:
None.

419-100018; Rev 4; 7/23

4) AUDIO SYSTEM CLOCK AND SAMPLE RATE ACCURACIES GOVERNED BY HSCLK SPECIFICATION

Description:
The HSCLK clock is the source for the digital audio peripheral that has an accuracy of approximately ±2.5%. Refer to the MAX32665–MAX32668 data sheet *Electrical Characteristics* table for guaranteed minimum and maximum values of f_{HSCLK}. (14416)

Workaround:
Ensure the application can tolerate audio clock and sample rate frequencies derived from the HSCLK.

5) SPI MODE 1 AND MODE 3 OPERATION MAY BE AFFECTED WHEN QSPIn_CLK_CFG.*SCALE* = 0

Description:
The following settings are invalid when operation the SPI is operating in mode 1 or mode 3:
SCALE=0, CLOCKHI=0, CLOCKLO=0
SCALE=0, CLOCKHI=1, CLOCKLO=1

Workaround:
Do not use the invalid settings. The operating speeds generated by all other field combinations are valid.

6) UART RECEIVER REQUIRES ONE EXTRA BIT TIME BETWEEN STOP AND START BITS

Description:
The UART requires an external transmitter to send at least one more stop bit than specified in the UARTn_CTRL0.stop field. (10650)

Workaround:
There are two workarounds:
1) Configure the transmitter to send at least one additional stop bit than specified in the UARTn_CTRL0.stop field.
2) Ensure the transmitter generates at least one idle bit time between byte transmissions.

7) DO NOT USE UART Tx AND Rx FUNCTIONALITY SIMULTANEOUSLY

Description:
The peripheral does not operate as expected if both the receive and transmit functions are used simultaneously. (10685)

Workaround:
There are two workarounds:
1) Use flow control to prevent an external UART from transmitting while the peripheral is transmitting.
2) Assign the transmit and receive functionality to different peripherals if full-duplex operation is required.

8) UART TRANSMITTER GENERATES SPURIOUS PULSE WHEN USING 7.3728MHz CLOCK SOURCE

Description:
The UART generates a negative pulse two PCLK periods wide on the UART_TX line one bit period before the falling edge of the start bit. (10689)

Workaround:
There are two workarounds:
1) Use PCLK as the clock source when transmitting.
2) Ensure the external receiver will reject the spurious pulse.

9) INCORRECT DATA MAY BE RETURNED DURING STANDARD DMA ACCESSES TO ECC-ENABLED SYSTEM RAM INSTANCES IF 8- OR 16-BIT OPERATIONS IMMEDIATELY FOLLOW 8- OR 16-BIT WRITE OPERATIONS

Description:
The ECC functionality has been defeatured from all revisions of the device due to the complexity of the workarounds. This information is provided for legacy use only and mention of the ECC feature was removed starting with revision 7 of the data sheet.

Incorrect data may be transferred if operations are performed on system RAM instances with their ECC enabled. (21206)

Workaround:
Ensure all DMA operations are 32 bits in length.
OR
Use a secondary DMA channel in conjunction with a primary DMA channel to perform DMA operations to and from ECC-enabled memory.

Considerations:
- The secondary DMA channel must be enabled before the primary DMA channel.
- The value of DMA_CHn_CNT.cnt for the secondary channel must be:

  (4 x number_of_primary_bursts) + 1

  For example, if both DMA_CHn_CNT.cnt = 2048 and DMA_CHn_CTRL.burst_size = 32 for the primary channel, then 64 bursts are required. DMA_CHn_CNT.cnt for the secondary channel is therefore:

  (4 x 64) + 1 = 257

Use the following steps to configure both the secondary and primary DMA channels:

1) Configure a secondary DMA channel for a dummy transfer of data. The destination is an unused location in the memory map, so this use of the secondary channel does not affect any usable memory.
   a) Set DMA_CHn_SRC.addr to 0x4000 0000.
   b) Set DMA_CHn_DST.addr to 0x4000 0200.
   c) Clear DMA_CHn_CTRL.srcinc to 0 to disable the source automatic incrementing.
   d) Clear DMA_CHn_CTRL.dstinc to 0 to disable the destination automatic incrementing.
   e) Set DMA_CHn_CTRL.srcwd to 0b10.
   f) Set DMA_CHn_CTRL.dstwd to 0b10.
   g) Set DMA_CHn_CTRL.burst_size to 0b00011.

h) Configure DMA_CHn_CTRL.pri of the secondary DMA channel to the same value as DMA_CHn_CTRL.pri of the primary DMA channel.
i) Configure DMA_CHn_CNT.cnt to the value calculated above.

2) Configure the primary DMA channel based on the transfer width. The source and destination address of the primary DMA channel must be aligned to a 32-bit boundary.
   - For a transfer width of 4 bytes:
     - DMA_CHn_CNT.cnt must be a multiple of 4.
     - DMA_CHn_CTRL.burst_size must be a multiple of 4 bytes.
   - For a transfer width of 2 bytes:
     - DMA_CHn_CNT.cnt must be a multiple of 8.
     - DMA_CHn_CTRL.burst_size must be a multiple of 8 bytes.
     - Set GCR_MEMCTRL.ramws_en to 1 to enable SRAM wait states.
   - For a transfer width of 1 byte:
     - DMA_CHn_CNT.cnt must be a multiple of
     - DMA_CHn_CTRL.burst_size must be a multiple of 4 bytes.
     - Set GCR_MEMCTRL.ramws_en to 1 to enable SRAM wait states.

10) CRYPTOGRAPHIC DMA OPERATIONS INVOLVING ECC-ENABLED SYSTEM RAM INSTANCES MAY TRANSFER INCORRECT DATA

Description:
The ECC functionality has been defeatured from the device due to the complexity of the workarounds. This information is provided for legacy use only and mention of the ECC feature was removed starting with revision 7 of the data sheet.

Incorrect data may be transferred if operations are performed on system RAM instances with their ECC enabled. (21207)

Workaround:
Do not use the cryptographic DMA (CDMA) on system RAM instances with their ECC enabled.

11) ON DEVICE VARIANTS PROVIDING THE SCPBL FEATURE, SECURE BOOT HANGS WHEN ECC IS ENABLED ON SYSRAM0

Description:
The ECC functionality has been defeatured from the device due to the complexity of the workarounds. This information is provided for legacy use only and mention of the ECC feature was removed starting with revision 7 of the data sheet.

The secure boot process runs after every reset and after exiting low power modes except SLEEP and DEEPSLEEP. The device will not exit the secure boot after these events. (21204)

Workaround:
Do not enable ECC on sysram0. A POR recovers the device if this issue is encountered.

12) POWER SEQUENCING REQUIREMENT FOR V_{REGI} AND V_{DDA} SUPPLIES

Description:
The design of the SIMO requires that V_{REGI} must rise to a valid level within a window related to the V_{DDA} voltage. (14477)

Workaround:
If power to the device is cycled, V_{REGI} must exceed $V_{REGI\_POR(MIN)}$ within 20ms after V_{DDA} > 1.24V. After that, V_{REGI} can settle to its final value.

This information has already been updated in revision 7 of the device data sheet.

13) SIMO MAY INCORRECTLY ENTER SOFT-START MODE WHEN EXITING OR ENTERING DEEPSLEEP POWER MODE

Description:
When the SIMO goes into soft-start while entering DEEPSLEEP, VCOREB may brownout if the leakage is high. This will be observed if the USB power switch is on during DEEPSLEEP.

It is also possible for VCOREA to be in soft-start when the system exits DEEPSLEEP mode and switches from VCOREB to VCOREA for normal operation. Without the software work around, a high internal current load on VCOREA will cause a brownout during the soft-start sequence. (14477)

Workaround:
Code compiled with SDK version 2023_06 or later implements this workaround and no action is required. Code compiled with earlier SDK versions can recompile to incorporate this workaround. The code implemented in the SDK is presented here for reference.

```
void MXC_LP_EnterDeepSleepMode(void)
{
    save_preDeepSleep_state();

    /*void prepForDeepSleep(void)*/
    {
        MXC_ICC_Disable();
        MXC_LP_ICache0Shutdown();

        /* Shutdown unused power domains */
        MXC_PWRSEQ->lpcn |= MXC_F_PWRSEQ_LPCN_BGOFF;

        switchToHIRCD4();

        /* Set SIMO clock, VDDCSW, VregO_B voltage for DeepSleep */
        /*void MXC_LP_SIMOprepForDeepSleep(uint32_t voltage)*/
        {
            /* Prevent SIMO soft start on wakeup */
            MXC_LP_FastWakeupDisable();

            /* Enable VDDCSWEN=1 prior to enter DEEPSLEEP */
            MXC_MCR->ctrl |= MXC_F_MCR_CTRL_VDDCSWEN;

            /* SIMO clock setup for deep sleep */
            *(volatile int *)0x40005434 = 1; /* SIMOCLKDIV [1:0] : 0=div1; 1=div8;
2=div1 3=div16 */
            /* BUCK_CLKSEL [25:24] : 0=8K; 1=16K; 2=30K; 3=RFU */
            *(volatile int *)0x40005440 = (*(volatile int *)0x40005440 & (~(0x3 <<
24))) |
```

```
                                        (0x2 << 24);
            /* BUCK_CLKSEL_LP [7:6] : 0=8K; 1=16K; 2=30K; 3=RFU */
            *(volatile int *)0x40005444 = (*(volatile int *)0x40005444 & (~(0x3 <<
6))) |
                                        (0x2 << 6);

            /* Wait for VCOREB to be ready */
            while (!(MXC_SIMO->buck_out_ready &
MXC_F_SIMO_BUCK_OUT_READY_BUCKOUTRDYB)) {}

            /* Lower VregB to reduce power consumption */
            MXC_SIMO_SetVregO_B(900);

            /* Move VCORE switch to VCOREB (< VCOREA) */
            MXC_MCR->ctrl = (MXC_MCR->ctrl & ~(MXC_F_MCR_CTRL_VDDCSW)) |
                            (0x2 << MXC_F_MCR_CTRL_VDDCSW_POS);

            /* Wait for VCOREA ready.  Should be ready already */
            while (!(MXC_SIMO->buck_out_ready &
MXC_F_SIMO_BUCK_OUT_READY_BUCKOUTRDYC)) {}
        }

    }

    MXC_LP_ClearWakeStatus();

    /* Set SLEEPDEEP bit */
    MXC_PWRSEQ->lpcn &= ~MXC_F_PWRSEQ_LPCN_BCKGRND;
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;

    /* Go into Deepsleep mode and wait for an interrupt to wake the processor */
    __WFI();

    /*void recoverFromDeepSleep(void)*/
    {
        /* SIMO soft start workaround on wakeup */
        /*void MXC_LP_recoverFromDeepSleep(void)*/
        {
            /* Check to see if VCOREA is ready on  */
            if (!(MXC_SIMO->buck_out_ready & MXC_F_SIMO_BUCK_OUT_READY_BUCKOUTRDYC))
{
                /* Wait for VCOREB to be ready */
                while (!(MXC_SIMO->buck_out_ready &
MXC_F_SIMO_BUCK_OUT_READY_BUCKOUTRDYB)) {}

                /* Move VCORE switch back to VCOREB */
                MXC_MCR->ctrl = (MXC_MCR->ctrl & ~(MXC_F_MCR_CTRL_VDDCSW)) |
                                (0x1 << MXC_F_MCR_CTRL_VDDCSW_POS);

                /* Raise the VCORE_B voltage */
                while (!(MXC_SIMO->buck_out_ready &
MXC_F_SIMO_BUCK_OUT_READY_BUCKOUTRDYB)) {}
                MXC_SIMO_SetVregO_B(1000);
                while (!(MXC_SIMO->buck_out_ready &
MXC_F_SIMO_BUCK_OUT_READY_BUCKOUTRDYB)) {}
            } else {
                if ((MXC_MCR->ctrl & MXC_F_MCR_CTRL_VDDCSW) == (1 <<
MXC_F_MCR_CTRL_VDDCSW_POS)) {
                    /* Raise the VCORE_B voltage */
                    while (!(MXC_SIMO->buck_out_ready &
MXC_F_SIMO_BUCK_OUT_READY_BUCKOUTRDYB)) {}
                    MXC_SIMO_SetVregO_B(1000);
```

```
                    while (!(MXC_SIMO->buck_out_ready &
MXC_F_SIMO_BUCK_OUT_READY_BUCKOUTRDYB)) {}
                }
            }
        }
    }
    restore_preDeepSleep_state();
}
```

14) SIMO MAY INCORRECTLY ENTER SOFT-START MODE WHEN EXITING OR ENTERING BACKUP POWER MODE

Description:
When the SIMO goes into soft-start while entering low power mode, VCOREB may brownout if the leakage is high. This will be observed if the USB power switch is on during the low power mode.

It is also possible for VCOREA to be in soft-start when the system exits BACKUP mode and switches from VCOREB to VCOREA for normal operation. Without the software work around, a high internal current load on VCOREA will cause a brownout during the soft-start sequence. (14477)

Workaround:
Code compiled with SDK version 2023_06 or later implements this workaround and no action is required. Code compiled with earlier SDK versions can recompile to incorporate this workaround. The code implemented in the SDK is presented here for reference.

```
void MXC_LP_EnterBackupMode(void *func(void))
{
    /*void prepForBackup(void)*/
    {
        MXC_ICC_Disable();
        MXC_LP_ICache0Shutdown();

        /* Shutdown unused power domains */
        MXC_PWRSEQ->lpcn |= MXC_F_PWRSEQ_LPCN_BGOFF;

        switchToHIRCD4();

        /* No RAM retention in BACKUP */
        MXC_LP_SetRAMRetention(MXC_S_PWRSEQ_LPCN_RAMRET_DIS);

        /* Disable VregB, VregD in BACKUP */
        MXC_LP_SIMOVregBPowerDown();
        MXC_LP_SIMOVregDPowerDown();

        /* Set SIMO clock, VDDCSW, VregO_C voltage for Backup */
        /*void MXC_LP_SIMOprepForBackup(uint32_t voltage)*/
        {
            /* Prevent SIMO soft start on wakeup */
            MXC_LP_FastWakeupDisable();

            /* Enable VDDCSWEN=1 prior to enter BACKUP */
            MXC_MCR->ctrl |= MXC_F_MCR_CTRL_VDDCSWEN;

            /* SIMO softstart workaround: clock 8KHz/16 for BACKUP, 30KHz/1
in ACTIVE */
```

```
            *(volatile int *)0x40005434 = 3; /* SIMOCLKDIV [1:0] : 0=div1;
1=div8; 2=div1 3=div16 */
            /* BUCK_CLKSEL [25:24] : 0=8K; 1=16K; 2=30K; 3=RFU */
            *(volatile int *)0x40005440 = (*(volatile int *)0x40005440 &
(~(0x3 << 24))) |
                                          (0x2 << 24);
            /* BUCK_CLKSEL_LP [7:6] : 0=8K; 1=16K; 2=30K; 3=RFU */
            *(volatile int *)0x40005444 = (*(volatile int *)0x40005444 &
(~(0x3 << 6))) |
                                          (0x0 << 6);

            /* Move VCORE switch to VCOREB (< VCOREA) */
            MXC_MCR->ctrl = (MXC_MCR->ctrl & ~(MXC_F_MCR_CTRL_VDDCSW)) |
                            (0x2 << MXC_F_MCR_CTRL_VDDCSW_POS);
            /* Lower VCOREA to save power */
            MXC_SIMO_SetVregO_C(850);

            /* Wait for VCOREA ready. */
            while (!(MXC_SIMO->buck_out_ready &
MXC_F_SIMO_BUCK_OUT_READY_BUCKOUTRDYC)) {}
        }
    }

    MXC_LP_ClearWakeStatus();

    MXC_PWRSEQ->buretvec = (uint32_t)(&Backup_Handler) | 1;
    if (func == NULL) {
        MXC_PWRSEQ->buaod = (uint32_t)(&Reset_Handler) | 1;
    } else {
        MXC_PWRSEQ->buaod = (uint32_t)(&func) | 1;
    }

    // Enable the VDDCSW to ensure we have enough power to start
    MXC_MCR->ctrl |= MXC_F_MCR_CTRL_VDDCSWEN;

    // Enable backup mode
    MXC_GCR->pm &= ~MXC_F_GCR_PM_MODE;
    MXC_GCR->pm |= MXC_S_GCR_PM_MODE_BACKUP;

}
```

15) MAXIMUM OPERATING TEMPERATURE REDUCED FROM +125°C to +85°C

Description:
The internal voltage regulator does not operate as expect for temperatures above +85°C. (14477)

Workaround:
None. The maximum operating temperature has been adjusted to +85°C for all revisions of the device.

16) MAXIMUM STORAGE TEMPERATURE REDUCED FROM +150°C to +125°C

Description:
Long-term storage at +150°C is not recommended as this will reduce the lifetime of the flash storage. (MBU929)

Workaround:
None. The maximum storage temperature has been adjusted to +125°C for all revisions of the device.

17) INITIAL VALUE OF GCR_SCON.OVR MUST BE 0b10

Description:
To ensure proper operation, the device must ensure GCR_SCON.ovr is 0b10 immediately after reset. (MBU709)

Workaround:
No workaround is needed if any of the following conditions are met:

- Code compiled with version 1.1.0-119 or later of the MAX32665 device libraries (released on August 7, 2022) implements this workaround and no action is required.
- The date code top marking on the device is 4022 or later.

Otherwise, ensure software writes GCR_SCON.ovr to 0b10 as soon as possible in the startup code.

MAX32665/MAX32666
REV A1 ERRATA

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	7/19	Initial release	—
1	12/20	Updated erratum 1, deleted erratum 3, added erratum 5	1, 2
2	8/21	Added errata 6–9	2, 3
3	3/22	Removed erratum 9, updated part numbers to match data sheet	All
4	7/23	Removed erratum 3 and replaced with new item, added errata 9-17	1, 3–9