Apollo2 MCU

Errata List

Doc. ID: SE-A2-5p00

Revision 5.0

Sep 2017

Table of Content

Introduction 3

Document Revision History 3

Errata Summary List 4

Detailed Silicon Errata 5

ERR007: Marginal Timing Condition in the IOM4 Module 6
ERR008: I2C Master Can Capture Incorrect Data on a Clock Stretch 10
ERR009: I2S Slave BCLK Propagation Delay 12
ERR010: MCU State Corrupted on Exit from Deep Sleep with only ADC Domain On 15
ERR011: Writes to I/O Host Registers at 0x78-0x7D Corrupt the I/O Slave (IOS) FIFO 16
ERR012: Missing IOS XCMPRR Interrupt 18
ERR013: IOINTW Interrupt Gets Raised Too Soon 19
ERR014: IOS Race Condition in Updating FIFOSIZ Field in FIFOPTR Register 20
ERR015: GENAD bit Gets Asserted in IOS INTSTAT 21
ERR016: Very Short I2C Stretch May Cause Short SCL “Glitch” Pulse 22
ERR017: Max SPI Clock on High-speed IOMs for VDDH < 2.0V 25
ERR018: Full Duplex Write Can Corrupt Last Input Write 26
ERR019: High Current Draw during Transition from Deep Sleep to Buck Active 27
ERR020: Read of FIFOREM Field of IOM FIFOPTR Register is not Guaranteed 30
ERR021: I2C Stretch on Last ACK Cycle Causes Timing Violation 31

Silicon Errata for the Apollo2 MCU (AMAPH1KK-xxx)

1. Introduction

This document is a compilation of detailed Silicon Errata for the Apollo2 MCU.

2. Document Revision History

Rev #	Date	Description
1.0	Jan 2017	Document initial public release
2.0	Mar 2017	ERR009 and ERR010 added
3.0	Apr 2017	ERR011 - ERR016 added
4.0	Jun 2017	ERR017 - ERR019 added
5.0	Sep 2017	ERR020 and ERR021 added; status of errata updated for B2; ERR019 updated

Table 1: Document Revision History

3. Errata Summary List

Below is a list of the errata described in this document. The reference number for each erratum is listed along with its description and link to the page where detailed information can be found.

Erratum Number, Title and Page	Affected Silicon Revisions	Resolution Status	Work-around
“ERR007: Marginal Timing Condition in the IOM4 Module” on page 6	Up to and including B0	Fixed in B2	Software
“ERR008: I2C Master Can Capture Incorrect Data on a Clock Stretch” on page 10	Up to and including B2	Partially fixed in B2	Software
“ERR009: I2S Slave BCLK Propagation Delay” on page 12	Up to and including B0	Fixed in B2	Hardware
“ERR010: MCU State Corrupted on Exit from Deep Sleep with only ADC Domain On” on page 15	Up to and including B0	Fixed in B2	Software
“ERR011: Writes to I/O Host Registers at 0x78-0x7D Corrupt the I/O Slave (IOS) FIFO” on page 16	Up to and including B0	Fixed in B2	Software
“ERR012: Missing IOS XCMPRR Interrupt” on page 18	Up to and including B0	Fixed in B2	Software
“ERR013: IOINTW Interrupt Gets Raised Too Soon” on page 19	Up to and including B0	Fixed in B2	Software
“ERR014: IOS Race Condition in Updating FIFOSIZ Field in FIFOPTR Register” on page 20	Up to and including B0	Fixed in B2	Software
“ERR015: GENAD bit Gets Asserted in IOS INTSTAT” on page 21	Up to and including B0	Fixed in B2	Software
“ERR016: Very Short I2C Stretch May Cause Short SCL “Glitch” Pulse” on page 22	Up to and including B2	Partially fixed in B2	Software
“ERR017: Max SPI Clock on High-speed IOMs for VDDH < 2.0V” on page 25	Up to and including B0	Fixed in B2	Software
“ERR018: Full Duplex Write Can Corrupt Last Input Write” on page 26	Up to and including B0	Fixed in B2	Software
“ERR019: High Current Draw during Transition from Deep Sleep to Buck Active” on page 27	Up to and including B2	Not fixed	Software
“ERR020: Read of FIFOREM Field of IOM FIFOPTR Register is not Guaranteed” on page 30	Up to and including B2	Not fixed	Software
“ERR021: I2C Stretch on Last ACK Cycle Causes Timing Violation” on page 31	Up to and including B2	Not fixed	Software

Table 2: Errata Summary

4. Detailed Silicon Errata

This section gives detailed information about each erratum. Information covered for each erratum includes the following:

- **Erratum Reference Number and Title** – Lists reference number and title of the erratum
- **Description** – Provides a detailed description of the erratum
- **Affected Silicon Revisions** – Specifies the silicon revisions on which the erratum exists
- **Application Impact** – Describes the impact of the erratum on a user application
- **Workarounds** – Proposes software or hardware workarounds to minimize or eliminate the risk of the erratum occurring
- **Erratum Resolution Status** – Specifies which silicon revision, if any, that the erratum was initially fixed
- **AmbiqSuite Workaround Status** – Specifies whether the erratum has been worked around in the AmbiqSuite software

4.1 ERR007: Marginal Timing Condition in the IOM4 Module

4.1.1 Description

This erratum concerns a marginal timing condition in the I2C/SPI Master Module, in particular, IOM4. This condition only occurs on the asynchronous load of the first byte of an IOM4 SPI or I2C transfer. No other byte transfers are affected on IOM4, and no other IOMs exhibit this condition.

This IOM4 timing condition does not occur on every part, and it only occurs on the first byte of a transfer. Bits affected can be different from part to part. If the condition occurs, a bit value of 1 will load as a 0 for the first byte of a master transfer.

4.1.2 Affected Silicon Revisions

This silicon erratum applies to all versions and packages of Apollo2 silicon, AMAPH1KK-xxx, up to and including silicon revision B0.

4.1.3 Application Impact

For all applications requiring the use of I2C/SPI IOM4, precautions need to be taken to assure that this timing condition does not occur in the application whereby the first byte of a master transfer could be misloaded.

4.1.4 Workarounds

In order to guarantee correct device operation, Ambiq Micro has developed a software workaround for the IOM4 marginal timing issue which has been implemented for the SPI in the Hardware Abstraction Layer (HAL) in AmbiqSuite. This workaround has been validated for up to 16 MHz operation only. Also, SPI flow control is not supported in this workaround. A workaround for I2C is not available at this time.

To avoid the timing condition of this erratum, the workaround requires that the maximum write transfer size is 4091 bytes instead of 4095 bytes. The CONT function is not supported for writes. The workaround takes advantage of an IOM feature which allows glitch-free changes in the IOM clock frequency, and is described in detail below. The HAL implements the software workaround only for IOM4 and only for the SPI communication channel.

HAL Write:

Refer to the timing sequence in Figure 1 for references to the occurrence of each of the following steps.

1. The nCE pin is initially configured as a GPIO and held high.
2. The FIFO is loaded with a dummy 32-bit word, which will be ignored by the slave device because nCE will not be asserted when it is transmitted. The dummy word will have all zeros in the first byte, so that the hardware issue, the erroneous load of a bit value of 0 instead of a 1, does not affect it. In the remaining dummy bytes, all bits will be the opposite polarity of the high order bit of the first real transfer byte (the first byte of the 2nd 32-bit word written to the FIFO). In this first scenario, the high order bit of Byte0 (real) is a 1, so the dummy bytes are all zero.

3. A write to the CMD register to perform a Raw Write is made, with the desired transfer length + 4 bytes. Interrupts are disabled prior to the CMD write.
4. The internal nCE signal goes low, but the nCE pin is held high as a GPIO.
5. The HAL code waits until the transfer is in the middle of the 4th byte. At that point the IOM frequency is divided by the smallest power of 2 which reduces it below 1 MHz. This allows the HAL to reliably change the nCE pin back to normal mode.
6. SW waits until it sees a transition on the MOSI pin, which(1)h can be observed as a GPIO even in MOSI mode. This indicates that the falling edge of the 32nd clock has occurred.
7. When the edge is detected, the nCE pin is switched back to the normal nCE function and the clock frequency is returned to the original requested frequency. Interrupts are enabled at this point.
8. The transfer then proceeds normally to completion.

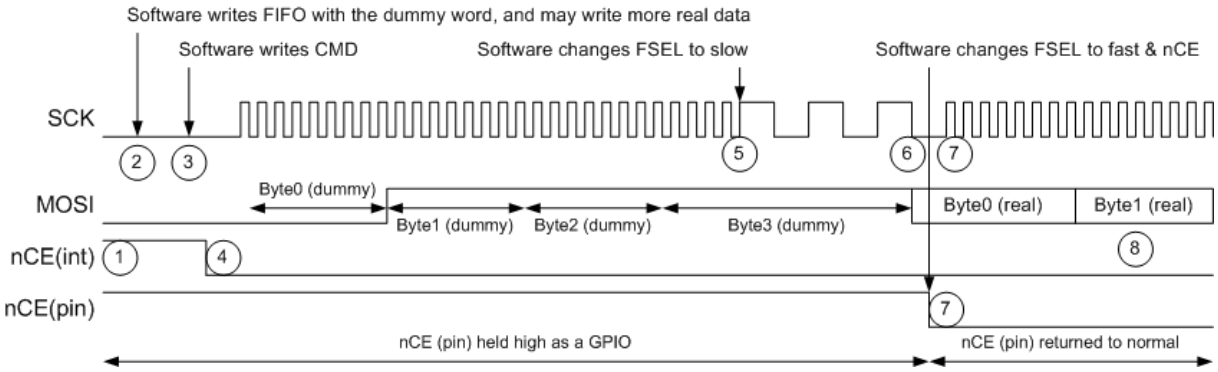

Figure 1. Software Workaround Write Timing

1. Timing between steps 3 – 5 is regulated by a ROM-based calibrated delay routine and the processing between steps 5 – 6 is hand coded in assembly language. The HAL code has been tested across the Ambiq supported toolchains (Keil, IAR, GCC), however we recommend that optimization be used.

The scenario in Figure 2 below presents the alternate case from the above (see step 2) where the high order bit of Byte0 (real) is a zero. In this case Byte0 (dummy) is all zeros, but the other dummy bytes are all ones. This produces an edge on MOSI at the same point (6) as the above scenario.

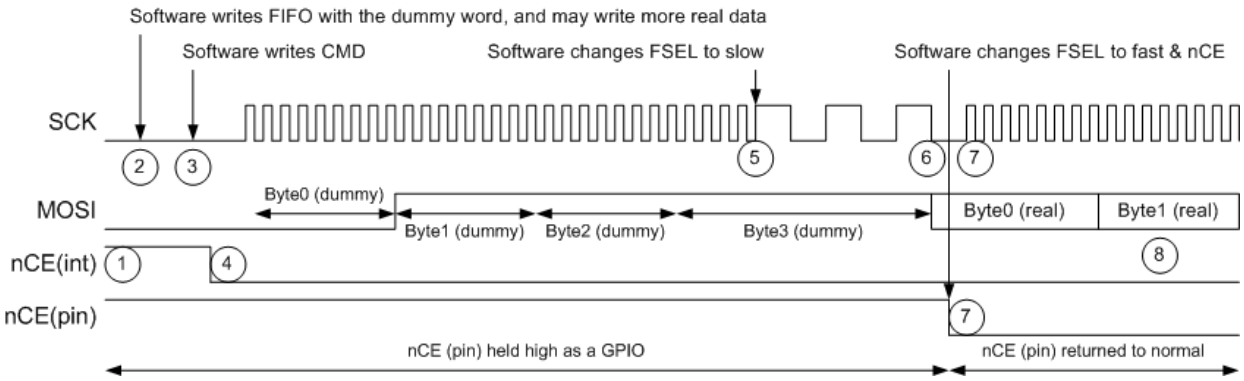

Figure 2. Software Workaround Write Timing - Alternate First-byte Scenario

HAL Read:

SPI read operation in the IOM4 workaround is implemented primarily in two separate operations which are combined into a single transfer using the CONT function. Refer to the timing sequence in Figure 3 for references to the occurrence of each of the following steps.

1. The nCE pin is initially configured as a GPIO and held high.
2. The initial operation is a write of 5 bytes, with the first four being dummy bytes as described above in the HAL Write operation and the 5th byte being the read offset byte. The CONT bit is set for this operation, which holds nCE low after it is configured as a normal function in step 4.
3. A write to the CMD register to perform a Raw Write is made, with the desired transfer length + 4 bytes. Interrupts are disabled prior to the CMD write.
4. nCE is configured as normal function and not as a GPIO at the end of the write operation.
5. The second operation is a raw read with the length from the desired read operation, and nCE is handled normally. The CONT bit is not set for this operation.

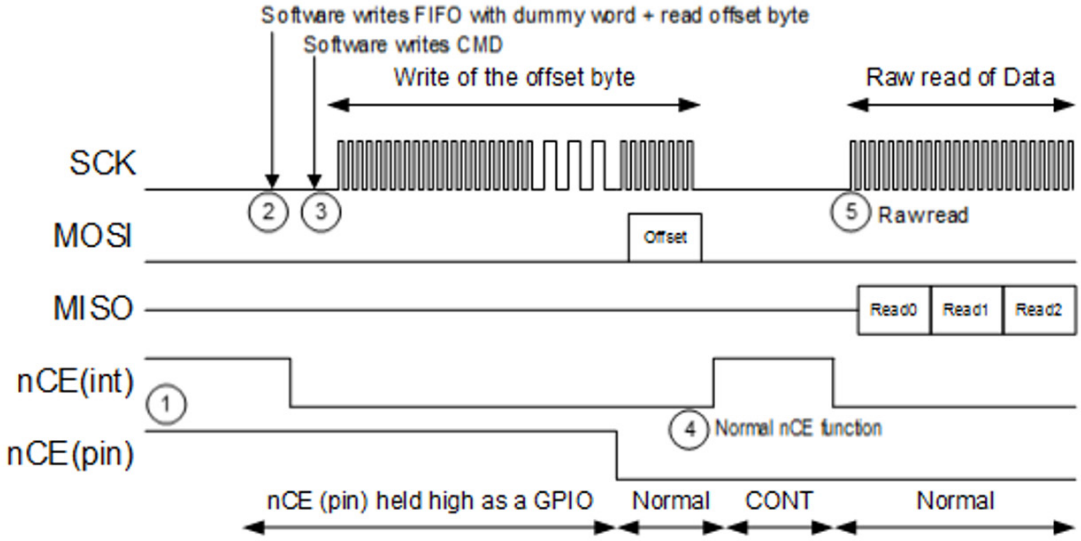

Figure 3. Software Workaround Read Timing

If a customer application does not use the AmbiqSuite software, it is recommended that the customer implements the workaround so that this condition is not encountered in application. In this event, the workaround implemented in the AmbiqSuite and described above can be used as a reference.

4.1.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.1.6 AmbiqSuite Workaround Status

The workaround code described in this erratum has been integrated in AmbiqSuite. Since the issue is fixed in Apollo2 revision B2, the workaround is automatically disabled when using B2.

4.2 ERR008: I2C Master Can Capture Incorrect Data on a Clock Stretch

4.2.1 Description

This erratum involves a potential limitation in the I2C communication channel in all I2C/SPI Master (IOM) modules on the MCU. The issue concerns the inability of the MCU, acting as an I2C master, to react to an I2C slave's attempt to stretch the clock (SCL). The slave normally is able to hold the SCL line low on any clock cycle of an I2C data transfer from the master for an unlimited amount of time to stall further master transmission. This is typically done when the slave is processing the I2C data in order to evaluate the address put on the bus and/or is preparing the next byte of data to be sent by the slave to the master.

Due to this erratum, when SDA is high (NAK) at the start of the ACK cycle and then later goes low (ACK) and the cycle is stretched by the slave, the master captures the value early in the cycle instead of at the end, so the wrong value gets processed. This anomaly could also produce an error when the SDA line starts out low and then goes high during the stretched cycle, and may also produce wrong data even for non-ACK cycles.

The effect of this erratum is that either the master may not process the data from the slave correctly after a clock stretch or, in some cases, may cease to operate entirely.

4.2.2 Affected Silicon Revisions

This issue affects all IOM modules on Apollo2 silicon revisions up to and including B0. It also affects the high-speed IOMs - IOM0 and IOM4 - on revision B2 but has been fixed for non-high-speed IOMs - IOM1, IOM2, IOM3 and IOM5 - on B2.

4.2.3 Application Impact

The impact of this erratum is that, without a software workaround, an MCU configured as an I2C master may not react properly to a slave's attempt to stretch the I2C clock. Therefore the MCU may be incompatible with certain I2C slave devices which utilize clock stretching as part of their I2C communication.

4.2.4 Workarounds

The recommended hardware workaround for this issue is to upgrade to silicon revision B2 and utilize IOM1, IOM2, IOM3 or IOM5 for I2C channel communication when there is the possibility of clock stretching by any slave in the system. Otherwise, ensure that no I2C-connected peripheral in the system requires, or will attempt to employ, clock stretching when the MCU is configured as the I2C master. When possible, peripherals which use SPI as the local bus should be selected instead of those requiring the use of I2C. Alternatively, a bit-bang implementation of an I2C master port can be employed instead of using the MCU's IOM module to communicate with I2C slave devices.

4.2.5 Erratum Resolution Status

This erratum has been fixed for IOM1, IOM2, IOM3 and IOM5 in silicon revision B2. The issue remains in the high-speed IOMs - IOM0 and IOM4 - on B2.

4.2.6 AmbiqSuite Workaround Status

For designs using B0 silicon, the AmbiqSuite IOM HAL provides for “bit bang” I2C transactions using a virtual IOM module. All of the HAL I2C function calls route to using this interface when the virtual IOM module (AM_HAL_IOM_I2CBB_MODULE) is selected, minimizing the code changes to switch from a physical IOM to this interface. The interface uses generic GPIO pins in order to produce the correct SDA/SCL waveforms for I2C reads and writes.

4.3 ERR009: I2S Slave BCLK Propagation Delay

4.3.1 Description

The I2S Slave module has internal propagation delays in the I2S_BCLK input path which prevent proper triggering and transmission of the PCM data from the I2S slave to an external controller (master).

Background:

The I2S Slave sends data to the external master synchronous with the master's bus clock, I2S_BCLK. Another line, the I2S_WDCLK, specifies the channel associated with the accompanying data. The data sent by the I2S Slave is synchronized with the BCLK, making 16 bits of data available per channel from the MSB to the LSB in order and readable on the rising edge of each BCLK cycle. Although there are 16 bits of data (one word) transmitted per WDCLK phase, there are 16 unused BCLK cycles within each WDCLK phase totaling 32 BCLK cycles per WDCLK phase.

The presentation of the data, I2S_DAT, by the slave occurs on the second rising edge of BCLK after a transition of the master's WDCLK. This WDCLK transition triggers the slave to send its corresponding channel data (high-to-low transition = left channel, low-to-high transition = right channel). The timing relationship between these two clocks as seen by the slave is critical, and follows a specific sequence:

1. The master continuously sends the BCLK.
2. When the master wants the slave to send channel data, it transitions WDCLK to start the process of the slave to send its corresponding channel data.
3. On the first falling edge of BCLK after the WDCLK transition, the slave presents the first data bit on the data bus. The data must be presented in less than 1/2 of a BCLK cycle in order for the data to be valid by the next rising edge of BCLK.
4. On the next (second) rising edge of BCLK after the falling edge on which the data is presented by the slave above, the master reads the data.
5. The slave presents another bit of data in turn on each subsequent falling edge of BCLK, and the master reads the data on each following rising edge until all 16 bits have been presented and read
6. Another 16 unused cycles of BCLK of no data passes during the WDCLK phase.
7. The master transitions the WDCLK line to initiate the presentation of alternate channel data by the slave, and timing between the two clocks and the slave's presentation of data repeats as in steps 3-6 above.

The issue:

The issue for this erratum is that WDCLK is sampled on the falling edge of BCLK instead of on the rising edge. At least 4 ns of hold time is required on WDCLK after the falling edge of BCLK, and if WDCLK transitions too close to the falling edge of BCLK, the WDCLK edge is missed completely. Internally, BCLK encounters some propagation delays which cause the WDCLK transition to occur too close to the falling edge of BCLK. The solution is to delay the transitions of WDCLK so they do not occur near the falling edge of BCLK, and the delay can be anything from 4 to 60 ns.

Another point that needs to be made is that the BCLK Input Inversion bit (BCLKINV) in the Voice Configuration Register (VCFG) is set to 0 (inverted) by default. For proper operation of the I2S, this bit must be set to 1 (normal polarity).

The timing diagram in Figure 4 shows the timing relationship among the BCLK, WDCLK and I2S_DATA signals as seen and required by the host controller. Here you can see that the I2S_WDCLK edge transition occurs an adequate time after the falling edge of I2S_BCLK for it to be sampled and observed.

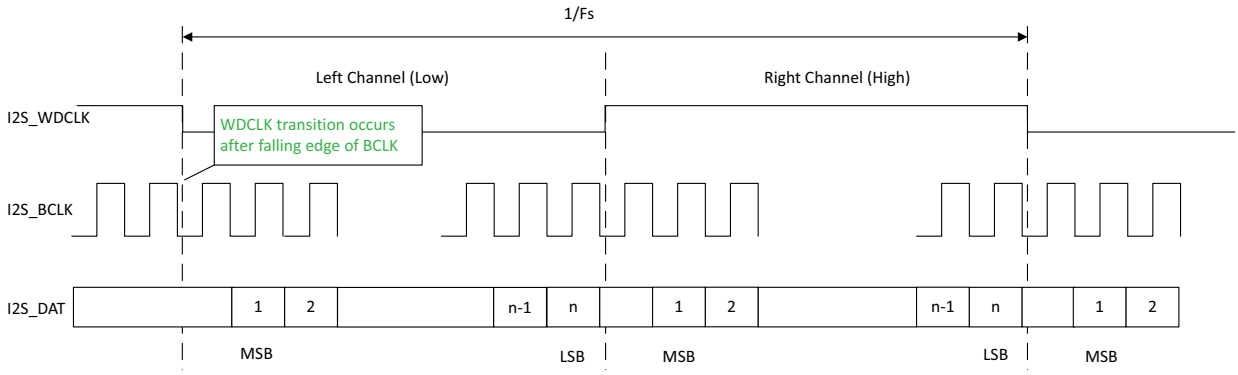

Figure 4. Valid I2S Timing

The timing diagram of Figure 5 shows the relative timing of the BCLK and WDCLK at the input of the I2S module internal to the MCU. This shows how propagation delays in the BCLK line causes the transition of WDCLK to occur too close to, or just prior to, a falling (sampling) edge of BCLK. This prevents the slave from seeing the WDCLK transitions and thus results in no transmission of data by the slave.

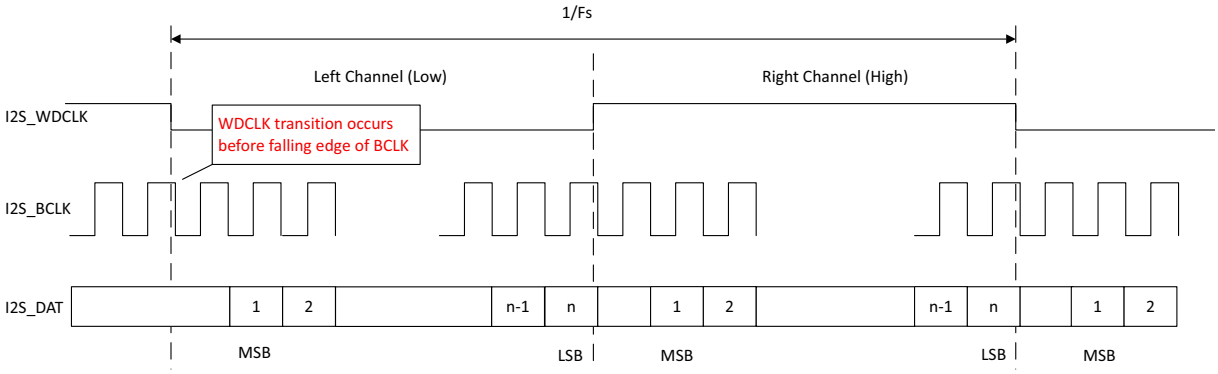

Figure 5. I2S Timing Due to Propagation Delays on BCLK input

4.3.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0.

4.3.3 Application Impact

Because of this issue the I2S Slave module is unusable without a workaround, which would eliminate the only useful communication channel for the transmission of PCM data.

4.3.4 Workarounds

The only known workaround to address this issue is a hardware implementation to delay the WDCLK input to the MCU's I2S module to offset the propagation delays on the BCLK line. A delay of only 4 ns is required on the WDCLK to eliminate this race condition, and can be accomplished by adding an R-C network on the WDCLK input to the MCU. This R-C lengthens the rise and fall times of the WDCLK clock edges long enough to ensure that the falling edge of BCLK occurs before a WDCLK transition. Suitable resistor and capacitor values are as follows:

- R = 4.7 kΩ
- C = 1000 pF

The required circuit is as shown in Figure 6.

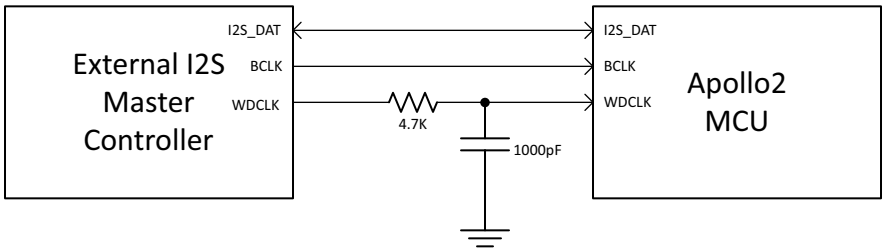

Figure 6. Interface Circuit for BCLK Propagation Delay Workaround

Implementation of this circuit causes an adequate delay on the WDCLK line which results in the proper timing relationship as seen in Figure 4. The I2S_WDCLK transition correctly occurs at least 4 ns after a falling edge of I2S_BCLK which triggers the initiation of left and right channel data transmissions.

4.3.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.3.6 AmbiqSuite Workaround Status

No software workaround is available or possible for this issue. A software workaround is not necessary as long as the hardware workaround mentioned above is implemented.

4.4 ERR010: MCU State Corrupted on Exit from Deep Sleep with only ADC Domain On

4.4.1 Description

The issue is caused by a timing relationship that exists when resuming from deep sleep when only the ADC device is enabled (PWRCTRL_DEVICEEN = 0x200). Because of this issue, the MCU state could become corrupted in certain conditions resulting in non-deterministic behavior.

4.4.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0.

4.4.3 Application Impact

Without a workaround impacted devices are not able to enter deep sleep with Core Buck enabled and with only the ADC device enabled (in DEVICEEN register).

4.4.4 Workarounds

The workaround for this issue is to disable the Core Buck in software by clearing the PWRCTRL_SUPPLYSRC_COREBUCKEN bit before going into deep sleep (only if the ADC is the only device enabled). Having the Core Domain in LDO mode ensures MCU state restoration under all conditions. There is a short delay of 20 cycles required prior to deep sleep entry to make sure that there is sufficient time for the posted write from the core to complete the switch from Core Buck to Core LDO mode.

4.4.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.4.6 AmbiqSuite Workaround Status

The deep sleep entry and exit routines in the AmbiqSuite HAL have been modified to implement this workaround to switch from Core Buck to Core LDO mode when the only device on is the ADC.

4.5 ERR011: Writes to I/O Host Registers at 0x78-0x7D Corrupt the I/O Slave (IOS) FIFO

4.5.1 Description

In the case where the FIFO area has been mapped such that some of its addresses coincide with the I/O Host Registers at 0x78 to 0x7D, writes to a register could corrupt the data at the corresponding address in the FIFO.

4.5.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0.

4.5.3 Application Impact

If not properly handled, host or slave writes to any of the R/W registers in the I/O Host Register set could cause corruption of data currently stored in the corresponding FIFO locations, resulting in incorrect data being read by the I/O Host. These registers include the HOST_IER (0x78), HOST_WCR (0x7A) and HOST_WCS (0x7B). This particular issue does not have any impact on the application if such communication is not needed.

4.5.4 Workarounds

There are two general strategies that could be used to work around this issue, which involves either using or not using the I/O Host Registers. In the latter case, the Direct Area of the I/O Host space can be used for interrupt-based communication between the host and MCU, and thus not needing to utilize the I/O Host Registers. Instead, Register Access (REGACC) interrupts are used for handshaking to facilitate host-MCU data transfer. In this scenario the issue described in this erratum does not have any impact on the application since the I/O Host registers are not being used for host-MCU coordination.

Using Register Access (REGACC) Interrupts

When host-MCU interrupts are needed, there are alternate means of providing them, which may involve handling the interrupts and communication completely out of band using GPIO's. For the host-to-MCU direction of communication, one approach for the host interfacing with the I/O slave (MCU) is to use register access (REGACC) interrupts in the Direct Area of LRAM. With this strategy the only accesses to the I/O Host Registers are host reads of the FIFO counter at 0x7C/7D and the FIFO Data at 0x7F.

As explained in Apollo2 Datasheet, up to 32 different interrupts could be used to communicate from host to MCU through REGACC status bits by reading Direct Access locations from 0x00 to 0x4F. By not having to write to the I/O Host Addresses at 0x78-0x7B and by being able to map the FIFO start address lower in LRAM, this enables the FIFO space to grow accordingly, optimizing data exchange with the host before the slave needs to come out of sleep mode to service the FIFO. If only 8 or fewer prioritized interrupts are required in this scheme, then a FIFOBASE setting of 0x01, reducing the Direct Area space to 0x00 to 0x07, can be used. In this case the FIFO Area could be set to take nearly all of LRAM.

Using I/O Host Registers

In this scenario, where the I/O Host registers from 0x78-0x7B are used for host-to-slave interrupts, the Direct Area should be sized to contain them so that these register addresses are not coincident with FIFO Area addresses. See the I2C/SPI Slave Module chapter of the Apollo2 Datasheet for background information about configuring the Local RAM (LRAM) for the I/O Host interface.

In this case the REG_IOSLAVE_FIFOCFG_FIFOBASE field must have a value of 16 (0x10), which will result in a total size of the Direct Area to be 128 bytes of directly mapped locations. With this mapping the resulting value for its end address is FIFOBASE*8-1, or 0x7F. Of course in this case the FIFO area would start at 0x80, and therefore be more limited than in the case where REGACC interrupts (only) are used.

Note that if ROBASE address is set such that the I/O Host Registers are coincident with host read-only addresses, writes to these registers could corrupt the values currently stored in these read-only locations.

4.5.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.5.6 AmbiqSuite Workaround Status

There is no specific workaround implemented in AmbiqSuite to avoid the effect of this erratum. Full flexibility is built into the software to allow the user to set FIFOBASE as desired. It is up to the user to handle this issue appropriately and as advised above.

4.6 ERR012: Missing IOS XCMPRR Interrupt

4.6.1 Description

A transfer complete interrupt for a read of the Direct Area of the I2C/SPI Slave's (IOS) LRAM may not happen under certain conditions. The conditions are a single-byte read from the last byte address of the Direct Area when the communication channel is the SPI.

4.6.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0.

4.6.3 Application Impact

This issue could impact the exchange of data or other information in the Direct Area between the I/O Host and slave.

4.6.4 Workarounds

To prevent this issue from occurring, do not use 1-byte reads from the last address of the Direct Area.

4.6.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.6.6 AmbiqSuite Workaround Status

There is no specific workaround implemented in AmbiqSuite to avoid the effect of this erratum. Full flexibility is built into the software to allow the user to do single-byte reads from any address in the Direct Area. It is up to the user to handle this issue appropriately and as advised above.

4.7 ERR013: IOINTW Interrupt Gets Raised Too Soon

4.7.1 Description

The interrupt which indicates that a host has written to an I/O Host register, IOINTW, gets asserted too early. Consequently, a read of the IOINT field of the IOINTCTL register to determine the interrupt source/cause can occur before the write interrupt has been recorded.

This issue occurs when the host is using the I2C channel, or the SPI channel at low speeds.

4.7.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0.

4.7.3 Application Impact

Host writes to any of the I/O Host Registers may not be observed properly and may not get acknowledged by the slave. If the slave depends on getting and reading these interrupts to take follow-up action, this functionality may not work reliably.

4.7.4 Workarounds

One workaround is not to use the I/O Host registers for the interrupt-driven host/slave interface, and instead use the register access (REGACC) interrupts to implement the interface.

A second workaround is to delay after getting an IOINTW interrupt long enough before reading the interrupt status. The assertion of the IOINTW interrupt occurs three interface clock cycles prior to when the INTSTAT_IOINTW field is updated if using the SPI (4 interface clock cycles if using I2C). A delay for that duration is needed after interrupt assertion before reading the INSTAT register in order for the slave software to determine the source (cause) of the interrupt.

4.7.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.7.6 AmbiqSuite Workaround Status

There is no specific workaround implemented in AmbiqSuite to avoid the effect of this erratum. Full flexibility is built into the software to allow reading an interrupt status field in the application software. It is up to the user to handle this issue appropriately and as advised above.

4.8 ERR014: IOS Race Condition in Updating FIFOSIZ Field in FIFOPTR Register

4.8.1 Description

In the Apollo2 I2C/SPI Slave Module (IOS), the decrement of FIFOSIZ on a FIFO read by the host can be missed if a slave software write to the FIFO occurs on the same clock cycle. Thus FIFOSIZ can be incorrect.

4.8.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0.

4.8.3 Application Impact

If not handled, this issue could cause the FIFO count, as reported in the FIFOPTR register, to be incorrect. This could cause data being missed or duplicated at the host.

4.8.4 Workarounds

One workaround is to implement a synchronizing handshake to guarantee that the slave software will never write the FIFO while the host is reading it.

Another option is to actively monitor for the occurrence of this race condition and to correct the FIFOSIZ field when needed. Refer to the AmbiqSuite HAL for an example of this solution.

4.8.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.8.6 AmbiqSuite Workaround Status

A workaround has been implemented in AmbiqSuite which re-syncs the FIFOSIZ if the race condition is detected. Since the issue is fixed in Apollo2 revision B2, the workaround is automatically disabled when using B2.

4.9 ERR015: GENAD bit Gets Asserted in IOS INTSTAT

4.9.1 Description

The General Address (GENAD) interrupt bit in the I2C/SPI Slave Module's (IOS) INTSTAT Register gets asserted during SPI operation under certain conditions. The GENAD interrupt is caused by a general address access during I2C operation, and should not occur during SPI operation.

4.9.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0.

4.9.3 Application Impact

If using SPI and the GENAD interrupt is enabled, then spurious interrupts may occur.

4.9.4 Workarounds

Ignore or disable GENAD interrupts during SPI operation.

4.9.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.9.6 AmbiqSuite Workaround Status

AmbiqSuite ignores the GENAD interrupt bit during SPI operation.

4.10 ERR016: Very Short I2C Stretch May Cause Short SCL “Glitch” Pulse

4.10.1 Description

When an IOM slave stretches the I2C clock for a period which is just slightly less than the I2C clock period, a short “glitch” pulse occurs which may result in erroneous operation within the slave device.

Erratum Details

The IOM operates on a free-running clock (ifc_clk) and during normal operation SCL is pulled low during the low period and released during the high period of the clock. A pull-up resistor pulls SCL high, so that normally SCL follows the clock. When the slave device determines on a falling edge of SCL that it needs to stretch the clock, it pulls SCL low with the internal signal stretch_n. At the rising edge of the clock, the IOM releases SCL but the slave device is holding it low.

The problem occurs because the slave device releases the pull-down of the internal stretch_n signal before the next falling edge of the clock. Since the IOM has already released SCL, it immediately goes high. However, since the SCL is sampled (as high) on the falling edge of the clock as shown in Figure 7, the stretch is not recognized and SCL is pulled low for the next cycle. This results in a very short high pulse on SCL (a "glitch") which can corrupt the operation within the slave device.

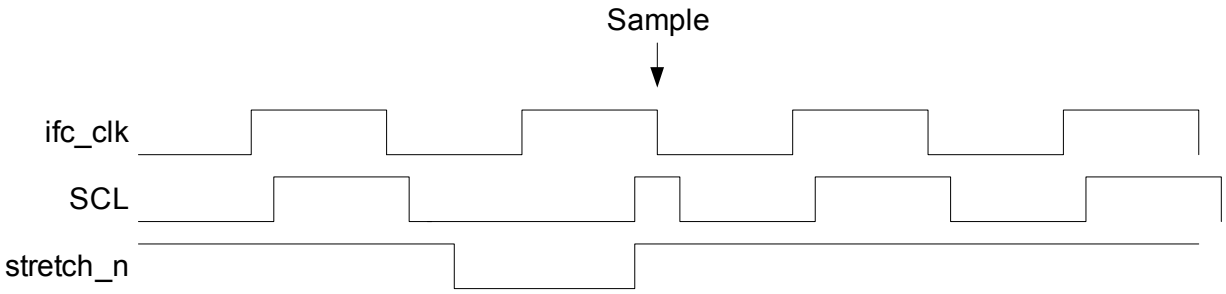

Figure 7. Shortened High Phase of SCL Due to Unrecognized Clock Stretch by Slave

Note that if the stretch lasts through the next falling edge of the clock, the stretch would be recognized and SCL would not be pulled low at that point. Thus the failure only occurs if three factors are evident, whereby the slave device:

1. Asserts a stretch,
2. Releases the stretch late enough in a cycle so as to produce a short SCL pulse, and
3. Fails when it sees the short SCL pulse.

It is rare for a slave device to assert stretch for such a short time to cause this issue. But with this short of a stretch, the occurrence of this anomaly is unavoidable since SCL cannot be sampled with the *rising* edge of the clock, as since SCL is later than the clock it would appear as though a stretch had occurred all the time.

4.10.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0. It also affects the high-speed IOMs on revision B2, but has been fixed for non-high-speed IOMs on B2 with I2C clock settings specified herein.

4.10.3 Application Impact

If using a I2C slave device which uses very short stretches, the slave device may be adversely affected by short glitch-like high pulses on the SCL line caused by untimely sampling and release of the SCL relative to the phase of the IOM clock.

4.10.4 Workarounds

This issue has been addressed and fixed in silicon revision B2 to eliminate the possibility of glitch pulses for non-high-speed IOMs, provided that certain I2C clock frequencies are used. The clock frequency must be a power-of-2 multiple of 50 kHz, e.g., 50 kHz, 100 kHz, 200 kHz, 400 kHz and 800 kHz, while staying within the specified limits as listed in the Apollo2 electrical specifications, in order for the clock stretching fix to work.

The following settings can be used to set the I2C clock at a couple of these valid frequencies where the clock stretching fix is effective.

- In the IOMSTR_CLKCFG register the TOTPER field must be set to 29 (0x1D) and the LOWPER field must be set to 14 (0xE).
- To run the I2C bus at 400 kHz, FSEL must be set to HFRC_DIV4 (to select a source clock of 12 MHz).
- To run 100 kHz, FSEL must be set to HFRC_DIV16 (to select 3 MHz).

For version B0 (all IOMs) and for the high-speed IOMs, IOM0 and IOM4, on B2 devices, there are two means of avoiding the issue - either will prevent the issue from occurring.

- **Run the clock faster** to make sure that the shortest stretch is longer than one clock cycle (so that a stretch will always be detected)
- **Run the clock slower** to make sure that the longest stretch is short enough that the resulting shortened SCL pulse is long enough to guarantee that it will not corrupt the slave device.

Each solution is described in detail below.

Running the Clock Faster

The first solution involves running the clock faster which, for a given stretch time, guarantees that the stretch time is longer than the clock cycle. This is shown in Figure 8 below. Since SCL is low at the Sample point, the stretch is detected and SCL is not pulled low then. Instead, the IOM waits until SCL is sampled high (at Sample1 in the figure) and then SCL is released one cycle later.

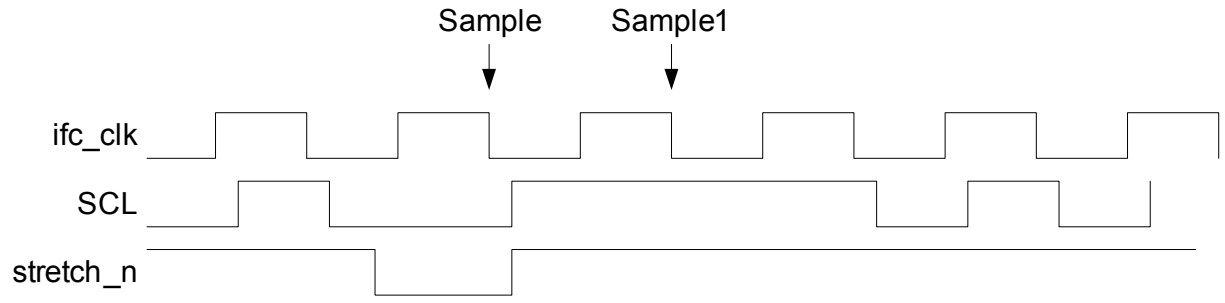

Figure 8. Occurrence of Sample with a Faster Clock

Running the Clock Slower

The second solution is to run the clock slower so that the short SCL pulse will still be long enough, as shown in Figure 9 below. The minimum required high time for the SCL in Fast Mode is 600 ns, so if the stretch time is 2.4 µs the clock frequency would have to be 1/(3.0 µs) = 333 kHz.

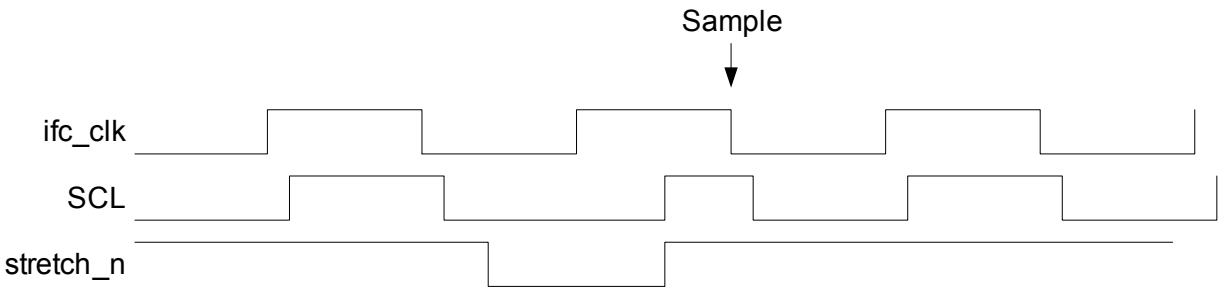

Figure 9. Occurrence of Sample with a Slower Clock

The challenge with this approach is that if the stretch time is longer than 2.4 µs, the glitch would be shorter and could still fail.

Conclusion

The problem with either of the above workarounds is that they require assumptions about the length of the stretch time. If some reasonable assumptions cannot be made about the expected duration of a slave device's stretch, then neither approach is guaranteed to work.

4.10.5 Erratum Resolution Status

This erratum has been addressed in silicon revision B2, enabling all non-high-speed IOMs to function normally and supporting slave clock stretching for specific clock settings as described herein.

4.10.6 AmbiqSuite Workaround Status

AmbiqSuite makes no assumptions about a slave's stretch duration in the case that they use clock stretching and the stretch length is very short (slightly less than the I2C clock period). It is up to the user to do this evaluation for the slave devices they have in their system as to whether this is an issue and, if so, whether the clock should be increased or decreased to avoid the issue.

4.11 ERR017: Max SPI Clock on High-speed IOMs for VDDH < 2.0V

4.11.1 Description

For the two high-speed I2C/SPI Master modules, IOM0 and IOM4, the maximum SPI clock is 16 MHz at V_{DDH} less than 2.0 V. For V_{DDH} higher than or equal to 2.0 V, the maximum SPI clock is 24 MHz as specified.

4.11.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0.

4.11.3 Application Impact

For applications which require that V_{DDH} be set to a voltage less than 2.0 V, the SPI clock must be set to no higher than 16 MHz on the high-speed IOMs.

4.11.4 Workarounds

There is no workaround for this erratum other than to operate the high-speed IOMs within the reduced SPI clock speed range when V_{DDH} is set to less than 2.0V.

4.11.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.11.6 AmbiqSuite Workaround Status

No software workaround is available or possible for this issue. AmbiqSuite supports the setting of the SPI clock in the high-speed IOMs to 16 MHz. There are no examples provided in AmbiqSuite which violate the timing constraints of this erratum.

4.12 ERR018: Full Duplex Write Can Corrupt Last Input Write

4.12.1 Description

On a write in full duplex mode (IOMSTRn_CFG_FULLDUP = 0x01), the last byte of a SPI transfer may be incorrectly placed in the FIFO because the FIFOPTR in the IOM is cleared too early, resulting in writing the last byte to the first location of the FIFO.

4.12.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B0.

4.12.3 Application Impact

The issue occurs very infrequently but is still a threat to get overwriting of existing data in the first byte of the receive FIFO when operating the SPI in any of the IOMs in full duplex mode.

4.12.4 Workarounds

The only workarounds for this issue is to either not use the SPI in full duplex mode, or account for the potential data loss/corruption in software (communication protocol) by padding the start and end byte of each transmission, or otherwise check for and handle this possible occurrence.

4.12.5 Erratum Resolution Status

This erratum has been fixed in silicon revision B2.

4.12.6 AmbiqSuite Workaround Status

AmbiqSuite has not supported full duplex SPI operation in versions up to SDK Release 1.2.10, so it is not affected by this erratum. Full duplex will be supported as a feature in Release 1.2.11 but, because of this issue existing on all silicon versions before B2, IOM full duplex should be used with B2 silicon only.

4.13 ERR019: High Current Draw during Transition from Deep Sleep to Buck Active

4.13.1 Description

The core buck converter supplied by VDDC and/or the memory buck converter supplied by VDDF could potentially go out of regulation during the transition from deep sleep to the buck active state. The buck converter could see much longer than expected turn-on time of the buck, which could draw a large amount of current from the supply. This issue is correlated with temperature such that the problem is more likely to occur at lower temperatures.

4.13.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions.

4.13.3 Application Impact

If the VDDC and/or VDDF supplies go out of regulation, a temporary high-current state could result in the rail(s) dropping out to 0V and the MCU/system hanging, thereby requiring a complete MCU power cycle to recover.

If using the AmbiqSuite workaround to prevent the effect of this erratum, the application impact is the shared or dedicated use of a CTIMER pair.

4.13.4 Workarounds

To prevent this problem, it is necessary to adjust the buck zero crossing trim field settings during the transition from deep sleep to buck active to provide additional noise immunity.

The workaround for this issue is to set the ZX trim for Core Buck to 0x7 in the MCUCTRL_BUCK3_COREBUCKZXTRIM field on deep sleep entry and restores ZX on deep sleep exit after TON count expiration. Use the latest version of AmbiqSuite with the workaround. If not using the SDK, then it is recommended to follow the general procedure in the SDK which is described below.

A brief outline for implementing the workaround is as follows:

- Designate a CTIMER pair (A and B) to use for the workaround.
- When going to deep sleep, clear timers and disable the bucks.
- Execute WFI.
- On wake, set the MCUCTRL.BUCK3.COREBUCKZXTRIM and MEMBUCKZXTRIM fields to 0x7.
- Enable bucks.
- Start timers.
- Continue with normal application execution.

Details of the above steps as implemented in the AmbiqSuite HAL are as described below.

CTIMER Interrupt Service Routines:

- Set up the main timer ISR (e.g., am_ctimer_isr) to handle the timer interrupts for the selected timer pair.
- Generally 2 sub-ISRs will be required - one for each timer. In AmbiqSuite these are named am_hal_sysctrl_buckA_ctimer_isr and am_hal_sysctrl_buckB_ctimer_isr.
- Each sub-ISR will serve to set the respective buck (core or mem) back to its original value.

NOTE: When using the AmbiqSuite HAL, all of this is conveniently handled with am_hal_sysctrl_buck_ctimer_isr_init, which uses the am_hal_ctimer_int_register function to register the two provided buck interrupt handlers, am_hal_sysctrl_buckA_ctimer_isr and am_hal_sysctrl_buckB_ctimer_isr.

One time initialization for the workaround:

- Save off the original buck ZX trim register settings.
- Register (or otherwise set up) the timer sub-ISRs.
- Clear and configure each of the CTIMERs as a one-shot (function 0) with TMR CLK for each set for buck input (0x10) and interrupts enabled.
- Enable the interrupts in the CTIMER interrupt enable register.
- Enable timer interrupts in the NVIC.

NOTE: As a reference, these steps are done in the am_hal_sysctrl_buck_ctimer_isr_init function.

When going to sleep, the following steps are to be performed:

- Check if Deep Sleep is requested. If normal sleep is requested, skip to the "Execute the WFI" step.
- Clear both designated CTIMERs.
- Set the CTIMER CMPR registers of each timer to 1, but don't enable the timers.
- Disable the bucks.
- Execute the WFI (Wait For Interrupt) instruction.

NOTE: As a reference, these steps are done in the am_hal_sysctrl_sleep function.

On waking from Deep Sleep, the following steps are to be performed:

- With bucks still disabled, set both MCUCTRL.BUCK3.COREBUCKZXTRIM and MEMBUCKZXTRIM fields to 0x7.
- Delay for 2us.
- Enable the bucks.
- Delay for 5us to allow the bucks to stabilize.
- Start the timers.
- Continue with normal application execution.

NOTE: As a reference, these steps are done in the am_hal_sysctrl_sleep function.

Inside each sub-ISR:

- Begin critical section.
- Delay 2us to allow buck to settle.
- Depending on which ISR is executing, restore the original setting to either the core buck or the mem buck.
  NOTE: For Apollo2, CTIMER buck inputs indicate bucks as follows:
  - For CTIMER 0 and 1: Timer A is mem buck, timer B is core buck.
  - For CTIMER 2 and 3: Timer A is core buck, timer B is mem buck.
- End critical section.

4.13.5 Erratum Resolution Status

A hardware fix is not planned at this time.

4.13.6 AmbiqSuite Workaround Status

AmbiqSuite implements a workaround for this erratum in SDK release 1.2.9.

The workaround requires the use of a pair of CTIMERS, that is, timers A and B of a designated CTIMER number. New HAL functions have been implemented to handle the buck zero cross workaround. They are:

- **am_hal_sysctrl_buck_ctimer_isr_init** - Does initializations and registers the following 2 ISRs.
- **am_hal_sysctrl_buckA_ctimer_isr** - Handler installed as part of the init function. Once registered, it is automatically called by am_hal_ctimer_int_service, which is called in am_ctimer_isr.
- **am_hal_sysctrl_buckB_ctimer_isr** - Handler installed as part of the init function. Once registered, it is automatically called by am_hal_ctimer_int_service, which is called in am_ctimer_isr.
- **am_hal_sysctrl_buck_update_complete** - Reports true when the buck trims have been restored.

The application calls am_hal_sysctrl_buck_timer_isr_init with only a single parameter, which is the CTIMER number to be used (0 - 3). It utilizes other existing CTIMER HAL features to set up and handle the interrupts from the designated timers.

The init function listed above initializes some state variables, initializes the two special HAL ISRs (named above) that handle the transitions, and sets up the designated timer pair to interrupt on Buck inputs.

The user must then only provide am_ctimer_isr, which must call am_hal_ctimer_int_service, which will in turn call the appropriate built-in ISRs (see the buckzx_demo example).

Once the buck trim restoration has been completed, the timers are disabled so no further interrupts will occur while awake. The workaround requires that the HAL function, am_hal_sysctrl_sleep, must be used for putting the device to sleep, otherwise the workaround will not function correctly.

4.14 ERR020: Read of FIFOREM Field of IOM FIFOPTR Register is not Guaranteed

4.14.1 Description

The FIFOREM field of IOM FIFOPTR register, which is intended to provide the number of bytes remaining in the FIFO, may be incorrect in the case when the count changes at the same cycle in which the register read occurs. In this case the FIFOSIZ field of the FIFOPTR register is correct, but FIFOREM may contain an intermediate and incorrect value.

4.14.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B2.

4.14.3 Application Impact

Using an incorrect value for FIFOREM could adversely affect the processing of FIFO data.

4.14.4 Workarounds

The recommended workaround is to read the FIFOPTR register's FIFOSIZ field, which contains the number of bytes currently in the FIFO, and calculate the remaining bytes in the FIFO (normally the FIFOREM field value). Calculating the remaining bytes in the FIFO is done by subtracting the FIFOSIZ field value from size of the FIFO, which is 128 in half duplex mode (IOM_CFG_FULLDUP = 0) or 64 in full-duplex mode (IOM_CFG_FULLDUP = 1).

4.14.5 Erratum Resolution Status

A hardware fix is not planned at this time.

4.14.6 AmbiqSuite Workaround Status

AmbiqSuite implements a workaround for this erratum in SDK release 1.2.9.

4.15 ERR021: I2C Stretch on Last ACK Cycle Causes Timing Violation

4.15.1 Description

If a slave stretches the clock on the last ACK cycle of a read and another IOM operation is initiated immediately, a violation of the required I2C timing on the initial cycle of the subsequent operation may occur. The IOM may proceed to the IDLE state even when a slave is stretching the clock on the last cycle of a read. If a new operation is initiated before the stretch ends, the timing of the SDA pulse to the first SCL edge may be violated.

4.15.2 Affected Silicon Revisions

This issue affects all Apollo2 silicon revisions up to and including B2.

4.15.3 Application Impact

A violation of I2C timing could result in corruption or misinterpretation of transmitted data.

4.15.4 Workarounds

Software can poll the SCL line prior to writing the CMD register on an I2C operation and verify that it is high (which indicates that the stretch has ended).

4.15.5 Erratum Resolution Status

A hardware fix is not planned at this time.

4.15.6 AmbiqSuite Workaround Status

AmbiqSuite will implement a workaround for this erratum in SDK release 1.2.10.

Contact Information

Address	Ambiq Micro, Inc. 6500 River Place Blvd. Building 7, Suite 200 Austin, TX 78730-1156
Phone	+1 (512) 879-2850
Website	http://www.ambiqmicro.com/
General Information	info@ambiqmicro.com
Sales	sales@ambiqmicro.com
Technical Support	support@ambiqmicro.com

Legal Information and Disclaimers