Mask Set Errata for Mask 1N87P

This report applies to mask 1N87P for these products:

- SAC57D54H

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR010835	2D-ACE: Incorrect values may be read while reading Heads Up Display registers
ERR008962	2D-ACE: Layers using RLE 16bpp compressed data require a byte swap
ERR009629	2D-ACE: Spurious generation of writeback underrun flag with HUD operation
ERR010327	ADC: Incorrect channel under measure is indicated after sampling phase of conversion until conversion completes
ERR011407	CMU: Sudden loss of clock does not signal the Fault Collection and Control Unit
ERR050877	Core: Fused MAC instructions give incorrect results for rare data combinations
ERR050878	Core: Processor reset asserted asynchronously could corrupt FPB comparator registers and remap to wrong address
ERR050090	DSPI/SPI: Incorrect data may be transmitted in slave mode
ERR009656	DSPI: Frame transfer does not restart in case of SPI parity error in master mode
ERR010755	DSPI: Transmit and Receive FIFO fill flags in status register is not cleared when DMA is improperly configured
ERR050395	ENET: Ethernet RX hang when receiving traffic through multiple queues
ERR010900	FCCU: False indication of a fault state for a single safe clock period can be generated on the error output pin
ERR010963	Flash: UTEST memory accesses may be corrupted when flash is operating between 33 MHz and 75MHz
ERR008004	FLASH: Array Integrity with Breakpoints enabled may skip addresses for certain RWSC and APC combinations
ERR007991	FLASH: Rapid Program or Erase Suspend fail status
ERR008341	FlexCAN: Entering Freeze Mode or Low Power Mode from Normal Mode can cause the FlexCAN module to stop operating.
ERR050246	FlexCAN: Receive Message Buffers may have its Code Field corrupted if the Receive FIFO function is used
ERR009265	FTM: Incorrect match may be generated if intermediate load feature is used in toggle mode

Table continues on the next page...

Table 1. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR008951	I2C: Attempting a start cycle while the bus is busy may generate a short clock pulse
ERR010526	IOP_modes: IOP Modes and related mode transition features are not supported.
ERR009156	LDB: OpenLDI 18-bit SPWG mode does not work correctly
ERR007274	LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state
ERR010379	MC_ME: Mode transition from DRUN/RUN mode to STANDBY may not complete if any Low Voltage Detect or Power-On Reset event is asserted during a susceptibility window
ERR010377	MC_ME: Mode transition from DRUN/RUN mode to STANDBY will not complete if a wakeup event is triggered in a 50ns window
ERR010265	MC_ME: Wakeup event while transitioning to a low power mode may cause the Mode Transition Status bit to remain asserted
ERR009250	PASS: JTAG password not working during reset
ERR050130	PIT: Temporary incorrect value reported in LMTR64H register in lifetimer mode
ERR010590	PLL: Might remain unlocked when enabled after Standby or power on
ERR010170	QuadSPI: Insufficient read data may be received in the RX Data Buffer register
ERR009461	QuadSPI: Read data errors may occur with data learning in 4x sampling method
ERR008950	RLE_DEC: eDMA transaction may not finish correctly on certain conditions
ERR010202	RTC: Software update of Time of Day register (RTC_TOD) might cause the internal counters to sample invalid values
ERR011306	SAR ADC: Incorrect value of ADC power down exit delay evaluated by the formula given in PDEDR [PDED] field description
ERR008799	SGM: 8-bit writes to FIFORP register incorrectly clears/flushes all channels
ERR050573	SIRC: Clock output may contain extra clock pulses
ERR009658	SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event
ERR009214	SPI: Only 16 bits are accessible on the SPI_PUSHR_SLAVE register
ERR009892	STCU: System not able to handle PLL Loss of Lock event during offline/online selftest
ERR050593	XRDC: DERRLOCn register may not capture the PAC and the MRC instance correctly in case of error violation
ERR009114	XRDC:Illegal values written to MRGD_W2[SNUM] and PDAC_W0[SNUM] will result is undefined behavior.

Table 2. Revision History

Revision	Changes
0	Initial revision
1	The following errata were added. • ERR010170 • ERR010202 • ERR010265 • ERR010327 • ERR010377 • ERR010379

Table continues on the next page...

Table 2. Revision History (continued)

Revision	Changes
2	The following errata were removed. • ERR008759 • ERR009386 The following errata were added. • ERR010542 • ERR010590
3, Apr 2017	The following erratum was removed. • ERR010542 The following errata were added. • ERR010526 • ERR010755 • ERR010835
4, JULY 2021	The following errata were added. • ERR010900 • ERR010963 • ERR011306 • ERR011407 • ERR050090 • ERR050130 • ERR050246 • ERR050395 • ERR050573 • ERR050593 • ERR050877 • ERR050878

ERR010835: 2D-ACE: Incorrect values may be read while reading Heads Up Display registers

Description: The 2D ACE block has a functionality of supporting Heads up display. There are registers related to Heads up display functionality (0x5000 – 0x5020) integrated within the IP. Reading these registers might return incorrect values.

Workaround: Software must not rely on read back values from these registers.

ERR008962: 2D-ACE: Layers using RLE 16bpp compressed data require a byte swap

Description: When using Run Length Encoding (RLE) mode on a layer from the Display Controller (2D-ACE) with a 16bpp format (RGB565, ARGB1555, ARGB4444, APAL8), the pixel values will appear incorrect because the compressed data requires a byte swap before compression, compared to uncompressed 16bpp formats.

Workaround: The RLE compression of images is usually performed on a computer or external host. If the RLE compressed images will be used on 2D-ACE layers, perform a byte swap operation for each 16bit pixel value on the image before performing the RLE compression.

For example, when not using RLE (raw data) a pixel value may be 0x00,0xFF. When using RLE mode on a layer, that pixel needs to be reordered to 0xFF,0x00 and then RLE compression can be applied to obtain the same output as in raw mode.

ERR009629: 2D-ACE: Spurious generation of writeback underrun flag with HUD operation

Description: When writeback operation is enabled with HUD operation, a false writeback underrun event is generated if the HUD descriptor has a different segment for last pixel of the last HUD line in comparison to the penultimate pixel. In such a case, the image is correctly written back to the system memory but writeback done flag is not set while writeback underrun is generated.

Workaround: The tool used to generate the HUD descriptor should ensure that the last pixel of the last line is in the same segment as the penultimate pixel.

ERR010327: ADC: Incorrect channel under measure is indicated after sampling phase of conversion until conversion completes

Description: The Main Status Register Channel under measure address field (ADC_MSR[CHADDR]) indicates which ADC channel is currently performing a conversion. This field indicates the correct channel during the sampling phase of conversion, but will display an incorrect value in the subsequent phases until conversion is complete.

Workaround: User must only consider ADC_MSR[CHADDR] to be valid when the ADC is in the sample phase of conversion. The Main Status Register Status of the ADC field shows when the ADC is in the sample phase (ADC_MSR[ADCSTATUS] = 0b100).

ERR011407: CMU: Sudden loss of clock does not signal the Fault Collection and Control Unit

Description: The Clock Monitor Unit (CMU) detects when the frequency of a monitored clock drops below a programmed threshold and asserts the Frequency Less than Low Threshold (FLL) signal if this occurs. The FLL signal is routed to the Fault Collection and Control Unit (FCCU) providing a mechanism to react to the clock fault but due to the monitoring implementation the FLL signal will not be triggered when the monitored clock suddenly stops.

Workaround: Each of the CMU monitored clocks has been analysed for the system level failure effect upon loss of the monitored clock and the safety mechanisms present to detect this. From this analysis it is concluded that loss of all the monitored clocks can be detected by other existing safety mechanisms in the system.

Further, since the CMU monitored clocks are derived from one of the system clock sources (IRC_CLK, XOSC_CLK, PLL0_PHI_CLK, PLL1_PHI_CLK or SDPLL_CLK) if the loss of the monitored clock is caused by the loss of the source clock then this will be detected and reported to FCCU by existing source clock loss detection mechanisms.

CMU_5 is an exception because it is possible to source the monitored LFAST_CLK from external LFAST_REF_CLK input instead of a system clock source. In the externally provided LFAST_CLK use-case the loss of LFAST_REF_CLK can only be detected by loss of the LFAST frame transmit/receive functionality which leads to interruption in the LFAST communication.

ERR050877: Core: Fused MAC instructions give incorrect results for rare data combinations

Description: Arm errata 839676

Affects: Cortex-M4F

Fault Type: Programmer Category B Rare

Fault Status: Present in: r0p0, r0p1. Open.

The Cortex-M4 processor includes optional floating-point logic which supports the fused MAC instructions (VFNMA, VFNMS, VFMA, VFMS). This erratum causes fused MAC operations on certain combinations of operands to result in either or both of the following:

• A result being generated which is one Unit of Least Precision (ULP) greater than the correct result.

• Inexact or underflow flags written to the Floating-point Status Control Register, FPSCR, incorrectly.

Configurations Affected:

Cortex-M4F configured with floating-point.

Conditions:

The conditions for this erratum are all of the following:

1. Cortex-M4 is configured with floating-point and it is enabled in the Coprocessor Access Control Register, CPACR.

2. Flush-to-Zero (FZ) mode is not enabled, that is, FPSCR.FZ is clear.

3. A fused MAC instruction (VFNMA, VFNMS, VFMA, or VFMS) is executed with all of the following properties:

• The addition part of the operation is Unlike Signed Addition (USA). This implies that the combination of the instruction used and the signs of the operands means that a subtraction is being performed.

• The result of the instruction, before rounding, is subnormal. This implies that the result is smaller in magnitude than 2-126.

• The significance of the product and the addend operand are the same or differ by one.

Implications:

If this erratum occurs, then the processor either produces an incorrect result to a computation or fails to run a floating-point exception routine when it should.

Note:

It is expected that most algorithms only use normalised numbers because:

• Subnormal results have low precision.

• It is easier to avoid underflow.

This erratum does not affect algorithms which only use normalized numbers.

Workaround: Enable FZ mode in the FPSCR.

ERR050878: Core: Processor reset asserted asynchronously could corrupt FPB comparator registers and remap to wrong address

Description: Arm errata 1299509

Affects: Cortex-M4, Cortex-M4F

Fault Type: Programmer Category C

Fault Status: Present in r0p0, r0p1. Open.

Normally, the debugger uses the Flash Patch and Breakpoint (FPB) unit for breakpoints or for patching ROM code during debugging. However, you can also use the FPB functionality as a method of in-the-field ROM code patching. On the Cortex-M4 processor, the processor reset can be asynchronously asserted and this could potentially cause problems if the processor is in the middle of writing to debug components (which are not reset by the processor reset) such as the FPB unit.

Configurations Affected:

A Cortex-M4 processor that is configured with debug. Conditions

1) The processor is programming the FPB to remap an address in ROM to fetch a replacement opcode or vector from an area in RAM.

2) The processor is asynchronously reset during the programming of the FPB.

3) The data is incorrectly written to the FPB register due to metastability.

Implications:

In the unlikely event of this occurring it may cause incorrect functional operation of the processor to occur. If the FPB is programmed with incorrect data it may cause the processor to unintentionally patch instructions.

Workaround: The problem does not arise if the FPB is not used to patch instructions. If the FPB is used to patch instructions, a software workaround is to ensure that the FPB is globally disabled before programming any comparators. If code exists which programs the FPB other than at reset, the software workaround should include: 1. Disable the FPB. 2. Program the individual comparators required. 3. Explicitly disable the individual comparators not required. 4. Re-enable the FPB. This sequence prevents any corruptly programmed FPB comparators from being activated.

ERR050090: DSPI/SPI: Incorrect data may be transmitted in slave mode

Description: If the Serial Peripheral Interface (SPI or the Deserial/Serial Peripheral Interface) is operating in slave mode, incorrect or stale data may be transmitted in next transaction without underflow interrupt generation if the set up time of the Peripheral Chip Select (PCS) to the SPI Serial Clock (SCLK) is short and the transmit FIFO may become empty after one transaction.

This can occur if the PCS to SCK is less than:

4 X IPG_CLOCK_PERIOD + 4 X DSPI_CLOCK_PERIOD + 0.5 x SCK_CLOCK_PERIOD

Where:

IPG_CLOCK is the internal bus clock (“system” clock)

DSPI_CLOCK is the protocol clock.

SCK_CLOCK is the Line-Side Serial Communication Clock.

Workaround: When operating in slave mode, software must ensure that the time interval between PCS assertion to start of SCK Clock is greater than 4 X IPG_CLOCK_PERIOD + 4 X DSPI_CLOCK_PERIOD + 0.5 x SCK_CLOCK_PERIOD.

To meet this requirement, the Master SPI can either lengthen the PCS to SCK assertion time or decrease the frequency of the communication interface, or both.

ERR009656: DSPI: Frame transfer does not restart in case of SPI parity error in master mode

Description: In the Deserial Serial Peripheral Interface (DSPI) module, in the scenario when:

1. Master/slave mode select bit (MTSR) of Module Configuration register (MCR) is set (MCR[MSTR]=0b1) to configure the module in master mode

2. SPI communication is selected via DSPI Configuration field (DCONF) in MCR (MCR[DCONF] = 0b00)

3. Parity reception check on received frame is enabled by setting the Parity Enable or Mask tASC delay (PE_MASC) bit of DSPI PUSH FIFO Register In Master Mode (PUSHR), i.e. PUSHR[PE]=0b1.

4. Parity Error Stop bit (PES) of MCR is set (MCR[PES]=0b1) which stops SPI frame transfer in case of parity error

5. Parity error is detected on received frame.

Then the next frame transfer is stopped, the SPI Parity Error Flag bit (SPEF) of the DSPI Status Register (DSPI_SR) is set (SR[SPEF] =0b1) and the corresponding SPI parity error interrupt is asserted. Even after the interrupt is serviced and SR[SPEF] is reset, the frame transfer does not restart.

Workaround: Do not use SPI frame transfer stop in case of parity error detection for SPI transmission in master mode. For this, keep the Parity Error Stop bit of Module Configuration Register de-asserted (MCR[PES] = 0b0).

ERR010755: DSPI: Transmit and Receive FIFO fill flags in status register is not cleared when DMA is improperly configured

Description: The Deserial/Serial Peripheral Interface Transmit and Receive First In/First Out (FIFO) buffers can request additional information to be transferred via the Direct Memory Access (DMA) module when either the Transmit or Receive FIFO Fill/Drain Flags are set in the DSPI Status Register (SR[TFFF/RFDF]). However, the Transmit Fill Flag indicates that at least 1 location each (2 bytes each) in the Transmit and Command FIFOs is available to be written. It does not indicate that the FIFO is empty. Similarly, Receive FIFO fill flag only indicates at least 1 location (2 bytes) of the FIFO is available to be read. It does not indicate that the FIFO is full. If the DMA is configured to transfer more than 1 FIFO location size of data, the FIFO Fill/Drain Flags may not be properly cleared indicating that the FIFO is not full even when the FIFO is actually full (for Transmit FIFO) and not empty when the FIFO is actually empty (for Receive FIFO).

Workaround: Properly configure the DMA to fill/drain only 2 bytes to Transmit, Command and Receive FIFOs. Use the DMA loop to transfer more data if needed.

ERR050395: ENET: Ethernet RX hang when receiving traffic through multiple queues

Description: Two or more applications are enabled to share the same Ethernet module by using different queues. At least 2 queues are configured to receive packets, with flushing enabled (RX_FLUSHx). When queues become full, packets are normally flushed, but under certain conditions of traffic, a lock-up of the Rx path can happen instead. When this occurs, the buffer descriptor for the last received packet contains an incorrect packet size (equal to the maximum buffer size). Packets cannot be received anymore, but the TX path remains unaffected. To recover the RX path, the ENET hardware block must be reset and re-configured.

Workaround: Unless the use case demands it, disable flushing to ensure the problem does not happen.

Or if reset is acceptable:

To recover the RX path, the ENET hardware block must be reset and re-configured

ERR010900: FCCU: False indication of a fault state for a single safe clock period can be generated on the error output pin

Description: The error out pin from the Fault Collection and Control Unit (FCCU) may pulse to a logic low (0b0) when the following conditions are fulfilled:

- software changes the error out protocol from a toggling protocol to a not-toggling protocol, and programs the FCCU_CFG.FCCU_SET_AFTER_RESET bit to 0b1
- software switches the Fault Collection and Control Unit (FCCU) state machine from CONFIG to NORMAL state

The duration of the glitch is equal to a single clock period of the Internal RC oscillator and there is a 50% of probability of the pulse occurring.

Workaround: Split the configuration of the FCCU in 2 phases.

During the first phase, software should do the following:

1) move the FCCU to the CONFIG state

2) configure the FCCU including the error out protocol, but without setting the FCCU_CFG.FCCU_SET_AFTER_RESET flag to 0b1 (leave as 0b0)

3) exits to the NORMAL state

During the second phase, software should do the following:

4) move the FCCU to the CONFIG state

5) set the FCCU_CFG.FCCU_SET_AFTER_RESET flag to 0b1

6) exit to the NORMAL state

Note: The default (after reset) error out protocol is the Dual Rail. Since this is a toggling protocol, the software must execute the above steps each time the user wants to switch to a non-toggling error out protocol.

ERR010963: Flash: UTEST memory accesses may be corrupted when flash is operating between 33 MHz and 75MHz

Description: Error Correction Code (ECC) errors may be generated in the flash due to corrupted data when all of the following conditions are met:

- The flash operating frequency is greater than 33MHz and less than or equal to 75MHz
- Read Wait State Control (PFLASH_PFCR1[RWSC]) = 2 and Address Pipeline Control (PFLASH_PFCR1[APC]) = 1
- Pipelined reads are executed between the UTEST flash block and any other flash block

Workaround: When the flash clock frequency is greater than 33 MHz and less than or equal to 75 MHz configure RWSC = 2 and APC = 0. The configuration for all other flash clock frequencies is specified in the MCU datasheet.

ERR008004: FLASH: Array Integrity with Breakpoints enabled may skip addresses for certain RWSC and APC combinations

Description: For certain combinations of the Flash Read Wait State Control (RWSC) and Address Pipeline Control (APC) settings in the Platform Flash Configuration Register (PFLASH_PCFR1) the Flash's array integrity (AI) check when run with breakpoints enabled may skip addresses resulting in an incorrect Multiple Input Signature Register (MISR) value or in the case of back to back ECC event errors (EER) or Single Bit Correction (SBC) events, a skipped breakpoint. This occurs for the following combinations:

RWSC=1 and APC=1

RWSC=3 and APC=2

RWSC=5 and APC=3

If breakpoints are enabled and an EER or SBC cause a breakpoint to occur the address after the breakpoint will be skipped, and the resulting MISR will not match expectations. Likewise, if there are back to back errors (EER or SBC) during AI with the above RWSC/APC combinations the 2nd error (and breakpoint) will be missed.

Margin Read (which by specification is a self timed event and is independent of wait states selected) is not affected by this erratum. This erratum only applies to Array Integrity.

Workaround: One workaround is to follow the recommended RWSC and APC combinations for given frequencies. If this is done, Array Integrity with Breakpoints feature works as expected. Valid RWSC/APC combinations listed in the specification are:

Flash Operating Frequency	RWSC	APC
30 MHz	0	0
100 MHz	2	1
133 MHz	3	1
167 MHz	4	1
200 MHz	5	2

A second workaround is if the above RWSC and APC combinations (listed in the description) are desired to be checked, do so without enabling breakpoints. In this case, the first EER or SBC event will be logged, and the MISR will correctly reflect the result of all reads being executed.

ERR007991: FLASH: Rapid Program or Erase Suspend fail status

Description: If a flash suspend operation occurs during a 5us window during a verify operation being executed by the internal flash program and erase state machine, and the suspend rate continues at a consistent 20us rate after that, it is possible that the flash will not exit the program or erase operation. A single suspend during a single program or erase event will not cause this issue to occur.

Per the flash specification, a flash program or erase operation should not be suspended more than once every 20 us, therefore, if this requirement is met, no issue will be seen. IF the suspend rate is faster than 20 us continuously, a failure to program/erase could occur.

Workaround: When doing repeated suspends during program or erase ensure that suspend period is greater than 20us.

ERR008341: FlexCAN: Entering Freeze Mode or Low Power Mode from Normal Mode can cause the FlexCAN module to stop operating.

Description: In the Flexible Controller Area Network (FlexCAN) module, if the Freeze Enable bit (FRZ) in the Module Configuration Register (MCR) is asserted and the Freeze Mode is requested by asserting the Halt bit (HALT) in MCR, in some cases, the Freeze Mode Acknowledge bit (FRZACK) in the MCR may never be asserted.

In addition, the Low-Power Mode Acknowledge bit (LPMACK) in the MCR may never be asserted in some cases when the Low-Power Mode is requested.

Under the two scenarios described above, the loss of ACK assertion (FRZACK, LPMACK) causes a lock condition. A soft reset action is required in order to remove the lock condition.

The change from Normal Mode to Low-Power Mode cannot be done directly. Instead, first change mode from Normal to Freeze Mode, and then from Freeze to Low-Power Mode.

Workaround: To avoid the lock condition, the following procedures must be used:

A) Procedure to enter in Freeze Mode:

1. Set both the Freeze Enable bit (FRZ) and the Halt bit (HALT) in the Module Control Register (MCR).

2. Check if the Module Disable bit (MDIS) in MCR register is set. If yes, clear the MDIS bit.

3. Poll the MCR register until the Freeze Mode Acknowledge bit (FRZACK) in MCR is set or the timeout is reached (see NOTE below).

4. If the Freeze Mode Acknowledge bit (FRZACK) is set, no further action is required. Skip steps 5 to 8.

5. If the timeout is reached because the Freeze Mode Acknowledge bit (FRZACK) is still cleared, then set the Soft Reset bit (SOFTRST) in MCR.

6. Poll the MCR register until the Soft Reset bit (SOFTRST) bit is cleared.

7. Reconfigure the Module Control Register (MCR)

8. Reconfigure all the Interrupt Mask Registers (IMASKn).

After Step 8, the module will be in Freeze Mode.

NOTE: The minimum timeout duration must be equivalent to:

a) 730 CAN bits if the CAN FD Operation Enable bit (FDEN) in MCR is set (CAN bits calculated at arbitration bit rate),

b) 180 CAN bits if the FDEN bit is cleared.

B) Procedure to enter in Low-Power Mode:

1. Enter in Freeze Mode (execute the procedure A).

2. Request the Low-Power Mode.

3. Poll the MCR register until the Low-Power Mode Acknowledge (LPMACK) bit in MCR is set.

ERR050246: FlexCAN: Receive Message Buffers may have its Code Field corrupted if the Receive FIFO function is used

Description: If the Code Field of a Receive Message Buffer is corrupted it may deactivate the Message Buffer, so it is unable to receive new messages. It may also turn a Receive Message Buffer into any type of Message Buffer as defined in the Message buffer structure section in the device documentation.

The Code Field of the FlexCAN Receive Message Buffers (MB) may get corrupted if the following sequence occurs.

1- A message is received and transferred to an MB (i.e. MBx)

2- MBx is locked by software for more than 20 CAN bit times (time determines the probability of erratum to manifest).

3- SMB0 (Serial Message Buffer 0) receives a message (i.e. message1) intended for MBx, but destination is locked by the software (as depicted in point 2 above) and therefore NOT transferred to MBx.

4- A subsequent incoming message (i.e. message2) is being loaded into SMB1 (as SMB0 is full) and is evaluated by the FlexCAN hardware as being for the FIFO.

5- During the message2, the MBx is unlocked. Then, the content of SMB0 is transferred to MBx and the CODE field is updated with an incorrect value.

The problem does not occur in cases when only Rx FIFO or only a dedicated MB is used (i.e. either RX MB or Rx FIFO is used). The problem also does not occur when the Enhanced Rx FIFO and dedicated MB are used in the same application. The problem only occurs if the FlexCAN is programmed to receive in the Legacy FIFO and dedicated MB at the same application.

Workaround: This defect only applies if the Receive FIFO (Legacy Rx FIFO) is used. This feature is enabled by RFEN bit in the Module Control Register (MCR). If the Rx FIFO is not used, the Receive Message Buffer Code Field is not corrupted.

If available on the device, use the enhanced Rx FIFO feature instead of the Legacy Rx FIFO. The Enhanced Rx FIFO is enabled by the ERFEN bit in the Enhanced Rx FIFO Control Register (ERFCR).

The defect does not occur if the Receive Message Buffer lock time is less than or equal to the time equivalent to 20 x CAN bit time.

The recommended way for the CPU to service (read) the frame received in a mailbox is by the following procedure:

1. Read the Control and Status word of that mailbox.

2. Check if the BUSY bit is deasserted, indicating that the mailbox is not locked. Repeat step 1) while it is asserted.

3. Read the contents of the mailbox.

4. Clear the proper flag in the IFLAG register.

5. Read the Free Running Timer register (TIMER) to unlock the mailbox

In order to guarantee that this procedure occurs in less than 20 CAN bit times, the MB receive handling process in software (step 1 to step 5 above) should be performed as a 'critical code section' (interrupts disabled before execution) and should ensure that the MB receive handling occurs in a deterministic number of cycles.

ERR009265: FTM: Incorrect match may be generated if intermediate load feature is used in toggle mode

Description: When a channel (n) match is used as an intermediate reload, an incorrect second match may occur immediately following the correct match. The issue is problematic only if channel (n) is configured for output compare with the output configured to toggle mode. In this scenario, channel (n) toggles on the correct match and again on the incorrect match. The issue may also occur if a certain channel has a match which is coincident with an intermediate reload point of any other channel.

Workaround: If any channel is configured for output compare mode with the output set for toggle mode, the intermediate reload feature must not be used.

ERR008951: I2C: Attempting a start cycle while the bus is busy may generate a short clock pulse

Description: When the I2C (Inter-Integrated Circuit) is operating in a multi-master network and a start cycle is attempted by the I2C device when the bus is busy, the attempting master will lose arbitration as expected but a short extra clock cycle is generated in the bus. After losing arbitration, the master switches to slave mode but it does not detect the short clock pulse. The acknowledge signal is expected at the ninth clock by the current bus master but it is not sent as expected due to the undetected short clock pulse.

Workaround: Software must ensure that the I2C BUS is idle by checking the bus busy bit in the I2C Bus Status Register (I2C_IBSR.IBB) before switching to master mode and attempting a Start cycle.

ERR010526: IOP_modes: IOP Modes and related mode transition features are not supported.

Description: IOP Modes and related mode transition features are not supported.

Workaround: The user must not use the IOP mode and related mode transitions. Any reference to IOP mode(s) and related mode transition must be ignored. Please contact your NXP representative for further details.

ERR009156: LDB: OpenLDI 18-bit SPWG mode does not work correctly

Description: When the device is configured to work in 18 bit SPWG Mode, the LVDS Display Bridge (LDB) incorrectly selects the least significant 18 bits of the output bus rather than the 6 most significant bits from each color component.

Workaround: Configure the Module in 24 bit JEIDA mode and do not use the last line (TX3) from the OpenLDI module. By doing this the module behaves exactly as required in SPWG 18-bit mode.

ERR007274: LINFlexD: Consecutive headers received by LIN Slave triggers the LIN FSM to an unexpected state

Description: As per the Local Interconnect Network (LIN) specification, the processing of one frame should be aborted by the detection of a new header sequence and the LIN Finite State Machine (FSM) should move to the protected identifier (PID) state. In the PID state, the LIN FSM waits for the detection of an eight bit frame identifier value.

In LINFlexD, if the LIN Slave receives a new header instead of data response corresponding to a previous header received, it triggers a framing error during the new header's reception and returns to IDLE state.

Workaround: The following three steps should be followed -

1) Configure slave to Set the MODE bit in the LIN Time-Out Control Status Register (LINTCSR[MODE]) to '0'.

2) Configure slave to Set Idle on Timeout in the LINTCSR[IOT] register to '1'. This causes the LIN Slave to go to an IDLE state before the next header arrives, which will be accepted without any framing error.

3) Configure master to wait for Frame maximum time (T Frame_Maximum as per LIN specifications) before sending the next header.

Note:

THeader_Nominal = 34 * TBit

TResponse_Nominal = 10 * (NData + 1) * TBit

THeader_Maximum = 1.4 * THeader_Nominal

TResponse_Maximum = 1.4 * TResponse_Nominal

TFrame_Maximum = THeader_Maximum + TResponse_Maximum

where TBit is the nominal time required to transmit a bit and NData is number of bits sent.

ERR010379: MC_ME: Mode transition from DRUN/RUN mode to STANDBY may not complete if any Low Voltage Detect or Power-On Reset event is asserted during a susceptibility window

Description: Power-On Reset (PORST) will NOT cause this issue if masked (UTEST_MISC[MASK_PORST] = 1, default: masked)

From power-up to DRUN there is a window (50 ns typical) at which time if any of the Low Voltage Detects (LVDs) are asserted, the mode transition will not complete. (Window #1)

From DRUN/RUNx to STANDBY transition there is a window (50 ns typical) at which time if the External Reset (EXR) is asserted, the mode transition will not complete. The window occurs in the STANDBY entry transition period (60 µs typical) - this period is from the mode transition request at the Mode Control Register (MC_ME_MCTL) to a toggle of the External Regulator Control pin (EXTREGC). The EXTREG pin signals the low power transition is complete. (Window #2)

Upon wake-up at exit from STANDBY there is a single window (50 ns typical) at which time if any of the LVDs are asserted, the mode transition will not complete. This window occurs in the STANDBY exit transition period (12 µs typical) immediately after assertion of the wake-up signal. (Window #3)

For the case the mode transition does not complete the MCU will be stuck in reset (Window #1) or stuck in STANDBY (Window #2 and #3) and will only recover via a power-cycle of VDDE_A.

Workaround: Ensure that any Low Voltage Detect or Power-On Reset (LVD / PORST) is not triggered during the windows of susceptibility at Power-up, at the entry to STANDBY mode or at the exit of STANDBY (window #1, window #2 and window #3 respectively).

ERR010377: MC_ME: Mode transition from DRUN/RUN mode to STANDBY will not complete if a wakeup event is triggered in a 50ns window

Description: While transitioning from DRUN/RUNx to STANDBY there is a susceptibility window (50 ns typical) at which if a wake-up event occurs the mode transition will not complete. The window occurs in the STANDBY entry transition period (60 µs typical) - this period is from the mode transition request at the Mode Control Register (MC_ME_MCTL) to a toggle of the External Regulator Control pin (EXTREGC). The EXTREGC pin signals the low power transition is complete.

For the case the mode transition does not complete the MCU will be stuck in STANDBY and will only recover via a power-cycle of VDDE_A.

Workaround: Ensure wake-ups are not triggered in the STANDBY entry transition period during the DRUN to STANDBY mode transition by adhering to all of the following guidelines:

1. If the application cannot guarantee to avoid triggering a wake-up event during the STANDBY entry transition period, prior to entering STANDBY mode all external wake-ups must be disabled.

2. Application SW should use a periodic wake-up (e.g. Real Time Clock Autonomous Periodic Interrupt) and poll the Wake-up Status register (WKPU_WISR). If a wake-up is recorded at WKPU_WISR this signals to the application SW that an external wake-up has occurred whilst in STANDBY mode.

3. The RTC-API timer must not timeout during the STANDBY entry transition period.

ERR010265: MC_ME: Wakeup event while transitioning to a low power mode may cause the Mode Transition Status bit to remain asserted

Description: While transitioning to a low power mode, if a wakeup interrupt event occurs between writing the Mode Control Register (MC_ME_MCTL) and executing a Wait For Interrupt (WFI) instruction, the low power mode transition will abort but the Mode Transition Status bit in the Global Status Register (MC_ME_GS[S_MTRANS]) will remain asserted.

The Mode Transition Status bit signals when a mode transition is ongoing. If the application software is polling this bit, it can cause an indefinite stall of the application.

Workaround: Workaround 1:

To avoid stalling of the application while waiting on the Mode Transition Status bit to be cleared, the Software Watchdog (SWT) should be enabled before starting the mode transition, this is achieved by setting the Watchdog Enable bit in the Software Watchdog Control Register (SWT_CR[WEN]). The SWT will issue a reset in case the device stalls waiting for the Mode Transition Status bit to be cleared.

Once in low power mode, If an interrupt by the SWT event is not desired, the Slow Internal RC Oscillator (SWT clock source) can be disabled in the target low power mode. In case the target mode is StandBy0 disabling the Slow Internal RC Oscillator is not required since the SWT is disabled in this mode.

Workaround 2:

If a reset is not desired, a timer can be configured to generate a timeout interrupt event, inside the interrupt service routine (ISR),a wait for interrupt (WFI) instruction needs to be executed again, the system will now enter the desired low power mode and the mode transition bit will be cleared.

If the previous wake-up event was to abort low power mode entry, a dummy timeout interrupt can be set prior to second WFI execution to exit low power mode.

ERR009250: PASS: JTAG password not working during reset

Description: The Debug Interface Access cannot be enabled by supplying the JTAG password during reset.

Workaround: To enable the Debug Interface Access, supply the JTAG password after reset.

ERR050130: PIT: Temporary incorrect value reported in LMTR64H register in lifetimer mode

Description: When the Programmable interrupt timer (PIT) module is used in lifetimer mode, timer 0 and timer 1 are chained and the timer load start value (LDVAL0[TSV] and LDVAL1[TSV]) are set according to the application need for both timers. When timer 0 current time value (CVAL0[TVL]) reaches 0x0 and subsequently reloads to LDVAL0[TSV], then timer 1 CVAL1[TVL] should decrement by 0x1.

However this decrement does not occur until one cycle later, therefore a read of the PIT upper lifetime timer register (LTMR64H) is followed by a read of the PIT lower lifetime timer register (LTMR64L) at the instant when timer 0 has reloaded to LDVAL0[TSV] and timer 1 is yet to be decremented in next cycle then an incorrect timer value in LTMR64H[LTH] is expected.

Workaround: In lifetimer mode if the read value of LTMR64L[LTL] is equal to LDVAL0[TSV] then read both LTMR64H and LTMR64L registers one additional time to obtain the correct lifetime value.

ERR010590: PLL: Might remain unlocked when enabled after Standby or power on

Description: When enabled after an event where the following conditions are met, the PLL might fail to lock:

1.- VDD_LV (1.2 V) was disabled

2.- A Pre-divider of 1 (PLLDIG_PLLDV[PREDIV] = 0 or 1) is used

3.- The PLL is clocked from the fast internal oscillator (FXOSC)

Common situations in which VDD_LV might be disabled are Standby and power-off of the device.

Workaround: There are three ways to avoid this:

1.- Avoid using a pre-divider of 1 (PLLDIG_PLLDV[PREDIV] = 0 or 1) and instead use a value of 2.

2.- During Standby leave the supply to VDD_LV enabled.

3.- Initialize the PLL using the fast internal oscillator (FIRC) as source clock and then switch to use the fast external oscillator (FXOSC).

ERR010170: QuadSPI: Insufficient read data may be received in the RX Data Buffer register

Description: Data read from flash through QuadSPI using Peripheral Bus Interface (IPS) may return insufficient data in the RX Buffer Data register (QuadSPI_RBDRn) when the read data size of a flash transaction is programmed to be greater than 32 bytes.

Workaround: For data size greater than 32 bytes, program the IP data transfer size in the IP configuration register (QuadSPI_IPCR[IDATSZ]) to be in multiples of 8 bytes.

ERR009461: QuadSPI: Read data errors may occur with data learning in 4x sampling method

Description: Data learning using 4x Sampling method may select a sampling point which is marginal. A marginal sampling point occurs when the sampling point is located on the edge of the valid sampling window. A marginal sampling point may return a positive comparison of the data learning pattern but small variations in voltage and temperature during the same read transaction may result in data errors, since the sampling point is not properly located inside the valid sampling window.

Workaround: There are two options:

- Internal DQS method allows to perform data learning as described on the Reference Manual.
- If 4x Sampling method is used, data learning should not be used and a fixed sampling point must be selected.

ERR008950: RLE_DEC: eDMA transaction may not finish correctly on certain conditions

Description: The following programming scenarios of Run Length Encoding Decoder (RLE_DEC) and eDMA may result in improper transactions:

Case 1:

When eDMA.TCD[NBYTES] is equal to eDMA.TCD[SSIZE] (in bytes), the RLE_DEC will trigger the first request but subsequent eDMA requests will not occur and the decoder will hang.

Case 2:

When THRESHOLD is equal to NBYTES, the RLE_DEC will trigger extra DMA request at the end of eDMA transaction which might set RXFIF or TXEIF flags for receiver RX and transmitter TX channels respectively.

Workaround: The following 2 conditions should be maintained on the RLE_DEC and eDMA configuration:

Condition 1:

eDMA.TCD[NBYTES] >= 2 x eDMA.TCD[SSIZE] (in bytes)

For example: eDMA.TCD[NBYTES] = 4 and eDMA.TCD[SSIZE] = 1 (16-bit) is a correct configuration.

Condition 2:

RLE_DEC.MCR[RX/TX_FIFO_THRESHOLD] >= eDMA.TCD[NBYTES] + eDMA.TCD[SSIZE] (in bytes)

For example: RLE_DEC.MCR[THRESHOLD] = 6 and eDMA.TCD[NBYTES] = 4 and eDMA.TCD[SSIZE] = 1 (16-bit) is a correct configuration.

ERR010202: RTC: Software update of Time of Day register (RTC_TOD) might cause the internal counters to sample invalid values

Description: A software write of the Real Time Clock Time of Day register (RTC_TOD) might cause the internal RTC counters (hour, minutes, seconds) to sample invalid values when the write coincides with the RTC counter clock edge. These invalid values are reflected in the RTC_TOD register only on the subsequent second interrupt.

Workaround: This errata can be handled in two ways:

1. Update RTC_TOD with RTC disabled

Disable the RTC to reduce the likelihood of the issue happening and verify if the written value is reflected on the TOD register at the next second interrupt, if not perform the write again.

2. Mimic RTC_TOD register on software

Create a variable to keep track of an offset with respect the Time of Day register. This offset will depend on the desired Time of Day value, for instance if the TOD register holds a value of 1:30:30 and the Time of Day is 2:00:30 an offset of 0:30:00 will be required to adjust the value in the TOD register with respect the actual time. Updates to the Time of Day are performed by adjusting the offset. The issue will never occur since the register will never be written, only the offset variable will be modified..

ERR011306: SAR ADC: Incorrect value of ADC power down exit delay evaluated by the formula given in PDEDR [PDED] field description

Description: The formula in the register field ADC_PDEDR [PDED] provides the delay between the power down bit reset and start of conversion value in number of clock cycles of the ADC module clock, however the given formula of PDED x 1/[ADC_clock_frequency] is incorrect. This gives a calculated value that is too short by 1 cycle of ADC Bus clock and 1 cycle of ADC clock (AD_clk).

Workaround: The correct formula that should be used to calculate the value for the ADC_PDEDR[PDED] register is -

(1/ADC Bus clock) + ((PDED+1) x 1/[ADC_clock_frequency])

Where:

ADC_clock_frequency = Frequency of ADC clock (AD_clk)

ADC Bus clock= Module interface clock for register access (ADC_CLK)

ERR008799: SGM: 8-bit writes to FIFORP register incorrectly clears/flushes all channels

Description: When 8-bit writes are made to to any of the channels (FRPCHn) of the FIFO read pointer register (FIFORP), all 4 channels have their FIFO, read/write pointers and flags cleared instead of just affecting the single channel that was written.

Workaround: To clear the FIFO, read/write pointers and flags of a single channel an 8-bit write to the relevant channel (FWPCHn) field in the FIFO write pointer register (FIFOWP) should be used.

ERR050573: SIRC: Clock output may contain extra clock pulses

Description: The 128 kHz Slow Internal RC (SIRC) oscillator may generate an extra clock pulse per clock period. The occurrence of the potential extra clock pulse may vary for each clock period ranging from no extra clock pulse to one extra clock pulse per period. Factors that affect the occurrence rate are part to part variations, temperature, and core voltage.

The SIRC output clock may be selected as the clock source for the Real Time Clock (RTC) via the RTC_CTRL[CLKSEL] and RTC_CTRL[SCLKSEL] registers, as a clock to monitor in the Clock Monitor Unit (CMU) via the CMU_CSR[CLKSEL1] register, as the clock source for the Software Watchdog Timer (SWT) timers (SWTx) or as the clock source for the LCD via the LCDCR[LCDOSC] register. The RTC, CMU, and SWT counters/timers may get an extra clock per SIRC clock period causing a higher than expected count (which may affect the RTC count/wakeup duration, the CMU measured SIRC frequency, or the SWT time-out period) or may cause a corrupted count value. The SIRC clock may also be selected to the CLKOUT_0 and CLKOUT_1 pins in which the extra pulse may or may not be observable for a divide by 1 configuration and for non-divide by 1 configurations may affect the divided clock duty cycle or may corrupt or stall the divided clock waveform.

Workaround: Select other clock sources for the RTC. The RTC supports other clock sources via the RTC_CTRL[CLKSEL] and RTC_CTRL[SCLKSEL] registers which include selections for the 32 kHz Slow External Crystal Oscillator (SXOSC), the 16 MHz Fast Internal RC Oscillator (FIRC),

or the 8-40 MHz Fast External Crystal Oscillator (FXOSC). If the RTC uses the SIRC clock, then the application needs to manage possible unexpected counter values such as count values that are double the expected value.

If the CMU monitors the SIRC clock, then the application needs to manage possible unexpected measured SIRC frequencies based on possible unexpected counter values such as frequencies that are double the expected value.

Select the other clock source for the LCD. Select the 32 KHZ Slow Oscillator (SXOSC) clock source for the LCD via the LCDCR[LCDOSC]] register. If the LCD uses the SIRC clock, then the application needs to manage possible unexpected LCD function.

The application needs to account for the possibility of a quicker SWT timeout for any SWT that is enabled. Other timer sources such as a PIT timer may be used to compliment the SWT counts.

If the CLKOUTx pin selects the SIRC clock source, then the application needs to manage the possible unexpected CLKOUT waveforms.

ERR009658: SPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event

Description: In the Serial Peripheral Interface (SPI) module, when both the receive FIFO and shift register are full (Receive FIFO Overflow Flag bit in Status Register is set (SR [RFOF] = 0b1)) and then the Clear Rx FIFO bit in Module Configuration Register (MCR [CLR_RXF]) is asserted to clear the receive FIFO, shift register data is sometimes loaded into the receive FIFO after the clear operation completes.

Workaround: 1. Avoid a receive FIFO overflow condition (SR[RFOF] should never be 0b1). To do this, monitor the RX FIFO Counter field of the Status Register (SR[RXCTR]) which indicates the number of entries in receive FIFO and clear before the counter equals the FIFO depth.

2. Alternatively, after every receive FIFO clear operation (MCR[CLR_RXF] = 0b1) following a receive FIFO overflow (SR[RFOF] = 0b1) scenario, perform a single read from receive FIFO and discard the read data.

ERR009214: SPI: Only 16 bits are accessible on the SPI_PUSHR_SLAVE register

Description: In the Serial Peripheral Interface module, the PUSH TX FIFO register in Slave Mode (SPI_PUSHR_SLAVE.TXDATA) all [31:0] bits are shown as accessible but actually only [15:0] bits are valid. Bits [31:16] are reserved and will always read as zero. Data written to SPI_PUSHR_SLAVE.TXDATA[15:0] will be directly transferred to TX FIFO. Reading from the SPI_PUSHR_SLAVE.TXDATA register will return the topmost entry of the TX FIFO.

Workaround: Only read or write the SPI_PUSHR_SLAVE.TXDATA[15:0] for valid data.

ERR009892: STCU: System not able to handle PLL Loss of Lock event during offline/ online selftest

Description: PLL Loss of Lock event during STCU offline/online selftest mode, where selftest is configured to run on PLL can cause the system to hang.

Workaround: It is recommended to run system on IRC for both offline/online selftest execution.

ERR050593: XRDC: DERRLOCn register may not capture the PAC and the MRC instance correctly in case of error violation

Description: XRDC provides access protection mechanism to protect all on-chip resources (registers,volatile- and non volatile memory areas and all other on-chip peripherals) against manipulation and/or unauthorized access from external or internal.

A write attempt by a non core master outside the defined ranges shall lead to an exception in case XRDC region is defined to prevent non core master access.The chip will generate bus error when XRDC policies are violated.

When either a memory region controller or a peripheral access controller detects a domain access violation, the address and attribute information of the offending access is captured. The DERRLOCn read-only registers provide additional information by signaling the instance number of the submodule for which the access violation(s) occurred.

It is recommended that the exception handler for XRDC policy violation begins by reading the HWCFG1 register to determine its domainID. Next, it should use the just-retrieved domainID to index into the DERRLOCn array. The resulting DERRLOCn value is then examined to determine

the instance number of the reporting MRC and/or PAC submodule.

XRDC implements access control and signals violations correctly but if there are multiple violation after the first violation successively then the DERRLOCn register is not able to provide the correct instance number of reporting MRC and/or PAC submodule.

Workaround: To ascertain the correct instance reporting MRC and/or PAC submodule that was source of access violation, follow the below software sequence

Read DID from HWCFG1 register to know masters DID

Read all DERR_W1_n registers to determine which all submodules failed (check EST bit)

Read DERR_W0_n registers for which DERR_W1_n[EST] bit is set

Write to DERR_W3_n corresponding to failing submodules to clear error and rearm them for subsequent captures

ERR009114: XRDC:Illegal values written to MRGD_W2[SNUM] and PDAC_W0[SNUM] will result is undefined behavior.

Description: Writes to the reserved bits above the most significant bits of MRGD_W2[SNUM] and PDAC_W0[SNUM] will result in undefined behavior.

Workaround: Avoid writing illegal values to MRGD_W2[SNUM] and PDAC_W0[SNUM].

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.