Chip Errata for i.MX RT1064_B

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR050164	BEE: Unaligned access may cause bus error
ERR050235	CCM: Incorrect clock setting for CAN affects UART clock gating
ERR050143	CCM: SoC will enter low power mode before the ARM CPU executes WFI when improper low power sequence is used
ERR011573	Core: Speculative accesses might be performed to memory unmapped in MPU.
ERR011572	Cortex-M7: Write-Through stores and loads may return incorrect data
ERR006223	Failure to resume from WAIT/STOP mode with power gating
ERR009595	FlexCAN: Corrupted frame possible if Freeze Mode or Low Power Mode are entered during a Bus-Off state
ERR005829	FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process.
ERR050246	FlexCAN: Receive Message Buffers may have its Code Field corrupted if the Receive FIFO function is used
ERR009527	FlexCAN: The transmission abort mechanism may not work properly
ERR011377	FlexSPI: DLL lock status bit not accurate due to timing issue
ERR050606	LPSPI: TCR value does not get resampled when polling the register
ERR050607	LPSPI: TCR[FRAMSZ] can be ignored when TCR[TXMSK]=1b1
ERR050130	PIT: Temporary incorrect value reported in LMTR64H register in lifetimer mode
ERR050194	QTMR: Overflow flag and related interrupt cannot be generated when the timer is configured as upward count mode
ERR050144	SAI: Setting FCONT=1 when TMR>0 may not function correctly
ERR050469	SEMC: 8/16bit write to 8bit PSRAM might cause data corruption
ERR011225	SEMC: CPU AXI write to SEMC NAND memory may cause incorrect data programmed into NAND memory
ERR050538	SOC: Potential boot failure on system reset if SJC_DISABLE fuse is blown
ERR010661	USB: There is an vbus leakage if USBOTG1’s vbus is on, and USBOTG2’s vbus from on to off for i.MX series

Table 2. Revision History

Revision	Changes
1.0	Initial revision
1.1	The following errata were removed. • ERR006281 • ERR007265 The following errata were added. • ERR011573 • ERR050143 • ERR050246 • ERR050606 • ERR050607 The following erratum was revised. • ERR010661

ERR050164: BEE: Unaligned access may cause bus error

Description: BEE only supports 16-byte burst access size. This rule may be violated if master sends unaligned burst, because bus fabric may convert unaligned burst into non-16-byte access and cause BEE to send a bus error.

Workaround: To avoid this issue, the general rule is to set start address to BEE 16-byte aligned and total data number a multiple of 16-byte. These settings will mean the bus fabric will always make 16-byte access requests to the BEE. If access master is CPU, enabling cache is recommended to avoid unaligned accesses.

ERR050235: CCM: Incorrect clock setting for CAN affects UART clock gating

Description: When selecting the CCM CAN clock source with CAN_CLK_SEL set to 2, the UART clock gate will not open and CAN_CLK_ROOT will be off. To avoid this issue, set CAN_CLK_SEL to 0 or 1 for CAN clock selection, or open the UART clock gate by configuring the CCM_CCGRx register.

Workaround: There are two workarounds for this issue:

- • Set CAN_CLK_SEL to 0 or 1 for CAN clock selection.

Or,

- • If CAN_CLK_SEL is set to 2, then the CCM must open any of UART clock gate by configuring the CCM_CCGRx register.

ERR050143: CCM: SoC will enter low power mode before the ARM CPU executes WFI when improper low power sequence is used

Description: When software tries to enter the low power mode with the following sequence, SoC enters the low power mode before the ARM CPU executes the WFI instructions.

• Set CCM_CLPCR[1:0] to 2'b00

• ARM CPU enters WFI

• ARM CPU wakes up from an interrupt event, which is masked by GPC or not visible to GPC, such as an interrupt from local timer.

• Set CCM_CLPCR[1:0] to 2'b01 or 2'b10

• ARM CPU executes WFI

Before the last step, SoC enters the WAIT mode if CCM_CLPCR[1:0] is set to 2'b01, or enters the STOP mode if CCM_CLPCR[1:0] is set to 2'b10.

Workaround: Software workaround

1) Trigger IRQ #41 (IOMUX), which is always pending by setting IOMUX_GPR1_GINT bit

2) Unmask IRQ #41 in GPC before setting the CCM low power mode

3) Mask IRQ #41 right after the CCM low power mode is set (set bit0-1 of CCM_CLPCR)

ERR011573: Core: Speculative accesses might be performed to memory unmapped in MPU.

Description: Arm errata 1013783-B

Cortex-M7 can perform speculative memory accesses to Normal memory for various reasons. All other types of memory should never be subject to speculative accesses.

The memory attributes for a given address are defined by the settings of the MPU when it is enabled. Regions that are not mapped in the MPU do not have any explicit attributes and should not be subject to any speculative accesses.

Because of this erratum, Cortex-M7 can incorrectly perform speculative accesses to such unmapped regions.

Conditions:

To trigger this erratum, the data cache must be enabled and the MPU must be enabled with the default memory map disabled. That is:

• CCR.DC = 1; data cache is enabled.

• MPU_CTRL.ENABLE = 1; MPU is enabled.

• If MPU_CTRL.PRIVDEFNA = 1, then this erratum cannot occur from privileged mode.

• If MPU_CTRL.HFNMIENA = 1, then this erratum cannot occur from the NMI or HF handlers or exception handlers when FAULTMASK = 1.

In these situations, a PLD instruction targeting an unmapped region might result in an incorrect speculative access. The PLD instruction itself could be speculative because of branch prediction. Even a literal data value that corresponds to a PLD encoding could theoretically cause this issue. This makes it difficult to scan code to check if these conditions apply.

Therefore, Arm recommends that any software with the MPU and data cache configured as mentioned in the conditions above uses the workaround below.

Implications:

Processor execution is not directly affected by this erratum. The data returned from the speculative access is never used and if the access is inferred by the program, then an abort will be taken as required.

The only implications of this erratum are the access itself which should not have been performed. This might have an impact on memory regions with side-effects on reads or on memory which never returns a response on the bus.

Workaround: Instead of leaving memory unmapped, software should use MPU region 0 to cover all unmapped memory and make this region execute-never and inaccessible. That is, MPU_RASR0 should be programmed with:

- MPU_RASR0.ENABLE = 1; MPU region 0 enable.
- MPU_RASR0.SIZE = b11111; MPU region 0 size = 2^32 bytes to cover entire memory.
- MPU_RASR0.SRD = b00000000; All sub-regions enabled.
- MPU_RASR0.XN = 1; Execute-never to prevent instruction fetch.
- MPU_RASR0.AP = b000; No read or write access for any privilege level.
- MPU_RASR0.TEX = b000; Attributes = Strongly-ordered.
- MPU_RASR0.C = b0; Attributes = Strongly-ordered.
- MPU_RASR0.B = b0; Attributes = Strongly-ordered.

Note that the MPU supports addressing hitting in multiple regions with the highest numbered region taking priority.

Therefore, use of MPU region 0 in this way does not affect the existing organization and use of MPU regions.

ERR011572: Cortex-M7: Write-Through stores and loads may return incorrect data

Description: Arm errata 1259864

If a particular sequence of stores and loads is performed on the Cortex-M7 core to Write-Through memory, and some timing-based internal conditions are met, then a load may not get the last data stored to that address.

This erratum can only occur if the loads and stores are to Write-Through memory. The following methods enable write-through mode of the cache:

1. The Memory Protection Unit (MPU) has been programmed to set this address as Write-Through.

2. The default memory map is being used, and this address is Write-Through in the default memory map.

3. The memory is cacheable, and the CM7_CACR.FORCEWT bit is set.

4. The memory is cacheable, shared, and the CM7_CACR.SIWT bit is set.

The following sequence is required for this erratum to occur:

1. The address of interest must be in the data cache.

2. A Write-Through store is performed to the same double-word as the address of interest.

3. One of the following:

• A linefill is started (to a different cacheline to the address of interest) that allocates to the same

set and way as the address of interest.

• An Error Correcting Code (ECC) error is observed anywhere in the data cache.

• A data cache maintenance operation without a following Data Synchronization Barrier (DSB).

4. A store to the address of interest.

5. A load to the address of interest.

If certain specific timing conditions are met, the load gets the data from the first store, or from what was in the cache at the start of the sequence instead of the data from the second store.

Under these conditions, a load can return incorrect data.

Workaround: There is no direct workaround for this erratum.

Where possible, Arm recommends that you use the MPU to change the attributes on any Write-Through memory to Write-Back memory. If this is not possible, it might be necessary to disable the cache for sections of code that access Write-Through memory.

ERR006223: Failure to resume from WAIT/STOP mode with power gating

Description: When entering WAIT/STOP mode with power gating of the core(s), if an interrupt arrives during the power down sequence, the system could enter an unexpected state and fail to resume.

Workaround: Use REG_BYPASS_COUNTER (RBC) to hold off interrupts when the PGC unit is in the middle of power down sequence. The counter needs to be set/cleared only when no interrupts pending. To use the work around effectively, the counter needs to be enabled as close to WFI as possible.

Following equation can be used to aid determination of RBC counter value.

RBC_COUNT * (1 / 32K RTC Frequency) >= (25 + PDNSCR_SW2ISO) * (1 / IPG_CLK Frequency)

PDNSCR_ISO2SW = PDNSCR_ISO = 1 (counts in IPG_CLK clock domain)

ERR009595: FlexCAN: Corrupted frame possible if Freeze Mode or Low Power Mode are entered during a Bus-Off state

Description: In the Flexible Controller Area Network (FlexCAN) module, if the Freeze Enable bit (FRZ) of the Module Configuration Register (MCR) is asserted and the Freeze Mode is requested by asserting the Halt bit (HALT) of the MCR register during the Bus Off state, the transmission after exiting the Bus-Off condition will be corrupted. The issue occurs only if a transmission is pending before the freeze mode request. In addition, the same issue can happen if Low-Power Mode is requested instead of Freeze Mode.

Workaround: The workaround depends on whether the bus-off condition occurs prior to requesting Freeze mode or low power mode.

A) Procedure to enter Freeze Mode:

1. Set the Freeze Enable bit (FRZ) 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 Low-Power Mode Acknowledge (LPMACK) bit in MCR is cleared (timeout for software implementation is 2 CAN Bits length).

4. Read the Fault Confinement State (FLTCONF) field in the Error and Status 1 Register (ESR1) to check if FlexCAN is in bus off state. If yes, go to step 5A. Otherwise, go to step 5B.

5A. Set the Soft Reset bit (SOFTRST) in MCR.

6A. Poll the MCR register until the Soft Reset (SOFTRST) bit is cleared (timeout for software implementation is 2 CAN Bits length).

7A. Poll the MCR register until the Freeze Acknowledge (FRZACK) bit is set (timeout for software implementation is 2 CAN Bits length).

8A. Reconfigure the Module Control Register (MCR).

9A. Reconfigure all the Interrupt Mask Registers (IMASKn).

5B. Set the Halt FlexCAN (HALT) bit in MCR.

6B. Poll the MCR register until the Freeze Acknowledge (FRZACK) bit is set (timeout for software implementation is 178 CAN Bits length).

NOTE: The time between step 4 and step 5B must be less than 1353 CAN bit periods.

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 (timeout for software implementation is 2 CAN Bits length).

ERR005829: FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process.

Description: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process. The following conditions are necessary for the issue to occur:

• Only one message buffer is configured to be transmitted

• The write which enables the message buffer to be transmitted (write on Control/Status word) happens during a specific clock during the arbitration process.

• After this arbitration process occurs, the bus goes to the Idle state and no new message is received on the bus.

For example:

1. Message buffer 13 is deactivated on RxIntermission (write 0x0 to the CODE field from the Control/Status word) [First write to CODE]

2. Reconfigure the ID and data fields

3. Enable the message buffer 13 to be transmitted on BusIdle (write 0xC on CODE field) [Second write to CODE]

4. CAN bus keeps in Idle state

5. No write on the Control/Status from any message buffer happens.

During the second write to CODE (step 3), the write must happen one clock before the current message buffer 13 to be scanned by arbitration process. In this case, it does not detect the new code (0xC) and no new arbitration is scheduled.

The problem can be detected only if the message traffic ceases and the CAN bus enters into Idle state after the described sequence of events.

There is no issue if any of the conditions below holds:

• Any message buffer (either Tx or Rx) is reconfigured (by writing to its CS field) just after the Intermission field.

• There are other configured message buffers to be transmitted

• A new incoming message sent by any external node starts just after the Intermission field.

Workaround: To transmit a CAN frame, the CPU must prepare a message buffer for transmission by executing the following standard 5-step procedure:

1. Check if the respective interrupt bit is set and clear it.

2. If the message buffer is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control/Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted. If backwards compatibility is desired (MCR[AEN] bit negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the message buffer, but then the pending frame may be transmitted without notification.

3. Write the ID word.

4. Write the data bytes.

5. Write the DLC, Control and CODE fields of the Control/Status word to activate the message buffer.

6. The workaround consists of executing two extra steps:

7. Reserve the first valid mailbox as an inactive mailbox (CODE=0b1000). If RX FIFO is disabled, this mailbox must be message buffer 0. Otherwise, the first valid mailbox can be found using the “RX FIFO filters” table in the FlexCAN chapter of the chip reference manual.

8. Write twice INACTIVE code (0b1000) into the first valid mailbox.

NOTE

The first mailbox cannot be used for reception or transmission process.

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.

ERR009527: FlexCAN: The transmission abort mechanism may not work properly

Description: The Flexible Controller Area NEtwork (FlexCAN) is not able to abort a transmission frame and the abort process may remain pending in the following cases:

a) If a pending abort request occurs while the FlexCAN is receiving a remote frame.

b) When a frame is aborted during an overload frame after a frame reception.

c) When an abort is requested while the FlexCAN has just started a transmission.

d) When Freeze Mode request occurs and the FlexCAN has just started a transmission.

Workaround: Use the Mailbox Inactivation mechanism instead of the transmission abort mechanism. The Abort Enable bit (AEN) of the Module Configuration Register should be kept cleared and the abort code value “0b1001" should not be written into the CODE field of the Message Buffer Control and Status word.

ERR011377: FlexSPI: DLL lock status bit not accurate due to timing issue

Description: After configuring DLL and the lock status bit is set, still may get wrong data if immediately read/write from FLEXSPI based external flash due to timing issue

Workaround: Adding a delay time (equal or more than 512 FlexSPI root clock cycle) after the DLL lock status is set.

ERR050606: LPSPI: TCR value does not get resampled when polling the register

Description: Reading the Transmit Command Register will return the current state of the command register.

Following a write to the TCR (Transmit Command register), if the user continuously reads the TCR (polls the register), then the read content no longer represents the contents of the Transmit Command register if it updates due to internal logic following the first read. The same value shall continue to be read.

Workaround: After reading the Transmit Command Register must always access a different register in between subsequent reads from TCR.

ERR050607: LPSPI: TCR[FRAMSZ] can be ignored when TCR[TXMSK]=1b1

Description: TCR (Transmit Command Register) is used to write new command word to the LPSPI transmit FIFO.

TCR[FRAMESZ] configures the frame size of the data to be transmitted in number of bits equal to (FRAMESZ + 1). When TCR[TXMSK] is set, transmit data is masked (no data is loaded from transmit FIFO and output pin is tristated). In master mode, the Transmit Data Mask bit will initiate a new transfer which cannot be aborted by another command word; the Transmit Data Mask bit will be cleared by hardware at the end of the transfer. TCR[CONTC] controls the continuous transfer mode. TCR[CONTC]=1b1 enables continuous transfer. In master mode, continuous transfer will keep the PCS asserted at the end of the frame size, until a command word is received that starts a new frame.

If command word is written with TCR[TXMSK]=1 and TCR[FRAMESZ]>32 and the next command word with TCR[CONTC]=0 is in the FIFO then at the end of any 32-bit word of the first command, the frame will terminate early and negate PCS.

Workaround: There are two workarounds:

1. Do not write a 2nd command word after writing command word with TXMSK=1 and FRAMESZ>32 until the first one has completed.

OR

2. Divide the command word into multiple command words with TXMSK=1 and FRAMESZ=32 (or remainder) using a continuous transfer.

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.

ERR050194: QTMR: Overflow flag and related interrupt cannot be generated when the timer is configured as upward count mode

Description: 1.Overflow flag and related interrupt cannot be generated successfully in upward count mode.

2.When TMR_CTRL[OUTMODE] is set to 110b, OFLAG output is not cleared on counter rollover when the timer counts upward.

Workaround: For item 1, using compare interrupt instead of overflow interrupt by setting compare value to 0xFFFF. The compare interrupt has the same timing effect as overflow interrupt in this way.

For item 2, there is no workaround.

ERR050144: SAI: Setting FCONT=1 when TMR>0 may not function correctly

Description: When FCONT=1 the transmitter will recover after a FIFO error when the FIFO is no longer empty and starting again from the same word in the following frame where the error occurred.

Configuring TMR > 0 will configure one or more words in the frame to be masked (nothing transmitted during that slot). If anything other than the last word(s) in the frame are masked when FCONT=1 and a FIFO Error Flag is set, then the transmitter will not recover and will set FIFO Error Flag during each frame.

Workaround: To avoid this issue, set FCONT in TCR4 to be 0.

ERR050469: SEMC: 8/16bit write to 8bit PSRAM might cause data corruption

Description: When SEMC is configured as 8bit PSRAM port, and the memory type is Normal type in MPU configuration for the PSRAM region, 8bit or 16bit writes to the region might cause data corruption.

Workaround: 8bit PSRAM port of SEMC works with following cases.

1). 32bit aligned write access

2). 8bit or 16bit write access with device memory attribution

ERR011225: SEMC: CPU AXI write to SEMC NAND memory may cause incorrect data programmed into NAND memory

Description: When SEMC NAND memory region is Normal type, non-cacheable, cacheable write-through, or write-back non-allocate and not hit, CM7 AXI write to the region could program incorrect data to the NAND memory.

Workaround: 1) Set SEMC NAND memory region to Device type or Strongly-ordered type in MPU and CPU only perform 32bit write to it or

2) Use eDMA to perform 64bit AXI write to SEMC NAND memory region or

3) Use IP command to program SEMC NAND memory

ERR050538: SOC: Potential boot failure on system reset if SJC_DISABLE fuse is blown

Description: By default, the JTAG/SWD clock is pulled high reset. When the SJC_DISABLE fuse is blown the clock is low. The fuses are reloaded during a system reset, so there is a window between the system reset assertion and fuse loading completion, during which the clock is high. When the fuses are loaded the clock will go low, causing a transition from high to low. There is another clock transition from low to high on a subsequent system reset. The clock toggles can cause the system to think JTAG/SWD is active. This causes a security violation leading to HAB boot failure.

Workaround: Configure the appropriate IOMUXC_SW_PAD_CTL register's PUE and PUS fields to enable a pull resistor on one of the following signals:

- Pull JTAG_TCK/SWD_CLK low
- Pull JTAG_TRST low
- Pull JTAG_TMS/SWD_DIO high

The IOMUXC registers retain state on a system reset, so this only needs to be done one time after each POR.

ERR010661: USB: There is an vbus leakage if USBOTG1's vbus is on, and USBOTG2's vbus from on to off for i.MX series

Description: When two USB ports are used in OTG mode simultaneously, the VBUS voltage for the second port (the port not selected by the ANATOP_HW_ANADIG_REG_3P0[VBUS_SEL] field) will be powered by the first port (the one that is selected by the VBUS_SEL field). voltage will not drop after cable unplug, then port can't detect the cable detach.

Workaround: Only have one port works as OTG, keep the others as host. Set the vbus_sel bit to select the host port.

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.