Mask Set Errata for Mask 2N27E

This report applies to mask 2N27E for these products:

- MC56F84789
- MC56F84786
- MC56F84783
- MC56F84769
- MC56F84766
- MC56F84763
- KC56F84763
- KC56F84769
- SC56F84766

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
e9432	eFlexPWMA: eFlexPWMA output pins may be with an output of high state (logic 1) instead of low state (logic 0) when Fractional delay block is powered up.
e10487	eFlexPWMB: Possible missed output edges when using dithering under certain conditions.
e10031	ADC12: Possible incorrect readings on second ADC conversion when sampling two ADC channels and first channel input voltage is less than VREFL or greater than VREFH.
e7725	ADC12: SCTRL register bits for converter B cannot be used with convertor B.
e6032	FlexCAN: A frame with wrong ID or payload is transmitted into the CAN bus when the Message Buffer under transmission is either aborted or deactivated while the CAN bus is in Bus Idle state.
e9729	FlexCAN: Corrupted frame possible if Freeze Mode or Low Power Mode are entered during a Bus-Off state.
e5829	FlexCAN: FlexCAN does not transmit a message that is enabled to be transmitted in a specific moment during the arbitration process.
e9732	FlexCAN: The transmission abort mechanism may not work properly.
e11484	POR: Residual voltage on VDD may cause POR unsuccessful.
e50194	QTMR: Overflow flag and related interrupt cannot be generated when the timer is configured as upward count mode.
e50273	eFlexPWM: Clock sources which are asynchronous to IPBus/ipg clock cannot be used as EXT_CLK of eFlexPWM.
e50308	ADC: ADCB may malfunction when independent parallel mode is used and differential mode is enabled for any channel of ADCB.

Table 2. Revision History

Revision	Changes
17 SEP 2014	No changes in this revision
19 JUN 2015	The following errata were added. • e9432
13 JAN 2016	The following errata were added. • e6032 • e5829 • e9729 • e10031 • e9732
12 MAY 2016	The following errata were added. • e7725
01 OCT 2016	The following errata were added. • e10487
01 OCT 2016	The following errata were added. • e10487
Feb 2020	New errata added: • e11484 • e50194 • e50273 • e50308
Dec 2020	Workaround description updated in the following errata: • e11484
Feb 2022	Added MC56F84783 new part, in the cover page.

e9432: eFlexPWMA: eFlexPWMA output pins may be with an output of high state (logic 1) instead of low state (logic 0) when Fractional delay block is powered up.

Description: eFlexPWMA outputs may be set to 1 upon powering up the fractional delay block. Because there is no reset signal to the circuit in the fractional delay block to force a specific reset state, the output must be cleared by creating a pulse on the PWM. This issue only occurs when using the fractional delay block and only lasts until the first time that PWM channel transitions.

Workaround: After powering up the fractional delay block of the eFlexPWMA by setting FRCTRL[FRAC_PU] and waiting the required power up time, program the VAL2-5 registers in all sub-modules to create a PWM pulse (>0% duty cycle) and run for at least one PWM period to clear the state of the registers in the fractional delay block. This can be done prior to enabling the PWM outputs so that external circuitry is not affected (no PWM pulse can be observed on the PWM output pins).

e10487: eFlexPWMB: Possible missed output edges when using dithering under certain conditions

Description: In PWM submodules that do not have a fractional delay block and support dithering instead, the PWM will fail to create falling/rising edges on its output under certain special conditions.

1. Submodule does not have a fractional delay block but is instead using dithering mode (FRCTRL[FRAC*_EN]==1) and
2. Submodule is using the master sync signal from submod0 as its initialization source (CTRL2[INIT_SEL]==10) and
3. Submodule VAL1==VAL3 or VAL1==VAL5 and
4. FRACVAL3 (or FRACVAL5) does not equal 0

Then over the course of the dithering, the submodule's output failling edge will occasionally be scheduled at cycle VAL3+1 in order to create the dithering. Since VAL3+1 is greater than VAL1 (since VAL1==VAL3), the falling edge is scheduled beyond the submodule's "normal" period as defined by INIT and VAL1. Since the submodule is using the master sync signal, the submodule counter will continue past VAL1 but the logic in the submodule will use the VAL1 timing to recalculate the edge placement, which results the falling edge will be lost.

Workaround: Set VAL1 in the submodule to a value greater than VAL3 or VAL5 but less than the period of the master submodule 0.

e10031: ADC12: Possible incorrect readings on second ADC conversion when sampling two ADC channels and first channel input voltage is less than VREFL or greater than VREFH

Description: When sampling two cyclic ADC channels that are listed consecutively by time and from the same ADC block, if the first channel input voltage is slightly less than VREFL or slightly greater than VREFH, then incorrect readings on the second conversion may result.

Workaround: For the ADC pins used during conversions, ensure that the voltages are between the values VREFL + 0.1 V and VREFH – 0.1 V.

e7725: ADC12: SCTRL register bits for converter B cannot be used with convertor B

Description: The ADC12 (cyclic ADC) is unable to use SCTRL register values for convertor B when the scan is set to one of the parallel modes with SIMULT being set to zero (parallel scanning and Converter A and Converter B operating independently). If bits 4~7 and bits 12~15 of SCTRL register, which corresponds to convertor B, are set to one, then the ADC conversion results are unpredictable.

Workaround: In parallel mode with SIMULT being set to zero, bits 4~7 and bits 12~15 of the SCTRL register must be set to zero.

e6032: FlexCAN: A frame with wrong ID or payload is transmitted into the CAN bus when the Message Buffer under transmission is either aborted or deactivated while the CAN bus is in Bus Idle state.

Description: The FlexCAN module may transmit an incorrect message if one or more Message Buffers (MBs) are configured for transmission while FlexCAN is in Bus Idle state, and the MB selected for transmission is either aborted or deactivated at the exact moment it starts to be transmitted. This will cause FlexCAN to transmit a syntactically correct message, but with either incorrect ID or data field. The CRC information will be calculated over the incorrect data (in case data is affected) and all other fields of the frame will be correct.

The probability of the problem occurring is limited to one CAN bit during the transmission of one frame, however under a very specific combination of simultaneous events:

a) Bug event may take place in one specific CAN bit per frame.

b) The CAN bus must be in Bus Idle state.

c) The CPU must be triggered to configure one or more MBs for transmission while in Bus Idle state.

d) The CPU must be triggered to remove the MBs just configured, by abortion or deactivation, in a short period after starting the configuration in step c).

In summary, the probability of occurrence is very low, in the order of 1 per 10 million. Moreover, the procedure of configuring a MB followed by abortion or deactivation of the same MB in a short interval is unlikely to occur in normal applications.

In practice, there is no issue if the CPU guarantees that any MB configured for transmission will be aborted or deactivated just in the next frame. Said differently, there is no issue if any MB configured for transmission lasts active for a minimum of 1 CAN frame.

Workaround: The user can avoid the error by preventing to make Message Buffer (MB) configurations for transmission when the CAN bus is in the Bus Idle state.

To do so, there are bits in a FlexCAN debug register that can be used to determine when the CAN bus is in Idle state.

This debug register is located at:

FlexCAN Debug 1 Register (CAN_DBG1) - Base + 0x0058

The CAN Finite State Machine (CFSM) bits of CAN_DBG1 register monitor the FlexCAN's internal state. The CFSM is the 6 least significant bits of the CAN_DBG1 register. The CAN Bit Number (CBN) is the 5 bits long field at bit positions 3 to 7 in the CAN_DBG1 register that indicates the current bit number in a given CFSM state value.

CAN_DBG1.CFSM = 0x0000_003F

CAN_DBG1.CBN = 0x1F00_0000

There are several internal states values that need to be looked for, listed below with their corresponding CFSM value.

RXINTERMISSION – 0x2F

TXINTERMISSION – 0x14

BUSIDLE – 0x02

The following procedure must be performed to configure a MB for transmission:

1) Disable all interrupts.

2) Read CAN_DBG1.CFSM and CAN_DBG1.CBN fields.

3) Check if CFSM value is either BUSIDLE, RXINTERMISSION or TXINTERMISSION. For the later two values, also check if CBN value is 3, to determine the paired conditions RXINTERMISSION bit 3 or TXINTERMISSION bit 3, and proceed as described below.

3.1) If CAN_DBG1 fields indicate BUSIDLE, wait N CPU clocks.

3.2) Else if CAN_DBG1 fields indicate either RXINTERMISSION bit 3 or TXINTERMISSION bit 3 wait until CFSM is different from either RXINTERMISSION or TXINTERMISSION.

3.3) Check again CAN_DBG1 fields, if they indicate BUSIDLE, wait for DELAY time.

4) Write 0x0 into Code field of CS word.

5) Enable all interrupts.

6) Write the ID word.

7) Write the DATA words.

8) Write 0xC into Code field of CS word.

NOTE

DELAY = {2*(MAXMB+1) + 18 } * peripheral_clock_period + 3 * PE_clock_period + 1 * CAN_bit_period.

The “Number Of The Last Message Buffer” (MAXMB) are the 7 least significant bits in Module Configuration Register (CAN_MCR: base + 0x0).

e9729: 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: To request a Freeze or Low Power Mode during the Bus Off state, the following procedure must be followed:

A) Procedure to enter in 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. Set the Soft Reset bit (SOFTRST) in MCR.

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

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

6. Reconfigure the Module Control Register (MCR)

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

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).

e5829: 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.

The workaround consists of executing two extra steps:

1. 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.

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

NOTE

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

e9732: 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.

e11484: POR: Residual voltage on VDD may cause POR unsuccessful.

Description: When powered on with residual voltage larger than 0.2 V on VDD, there is a chance that POR signal releases when VDD rises to a value around 2.1 V, and VCAP has not reached 1.2 V yet. It may cause system to work abnormally, such as some internal peripherals cannot be reset successfully, or the core cannot run. The residual voltage may be caused by an external clamping diode to VDD with the anode powered up earlier than VDD, or VDD not fully dropped to below 0.2 V from last time powered off.

Workaround: To avoid formalizing a residual voltage, VDD ramp-up shall be monotonic increase, with ramp-up rate during 0 to 0.5 V not slower than the rest of ramp period, and the whole ramp-up time is between 1 ms and 200 ms.

e50194: 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.

e50273: eFlexPWM: Clock sources which are asynchronous to IPBus/ipg clock cannot be used as EXT_CLK of eFlexPWM.

Description: EXT_CLK input of eFlexPWM is not resynchronized. When SMxCTRL2[CLK_SEL] is set to 01b, meaning EXT_CLK is used as the clock for the local prescaler and counter, and a clock source which is asynchronous to the IPBus/ipg clock is routed to EXT_CLK, there can be unpredicted eFlexPWM behavior.

Workaround: When EXT_CLK is enabled, use the clock source which is synchronous to IPBus/ipg clock for EXT_CLK only.

e50308: ADC: ADCB may malfunction when independent parallel mode is used and differential mode is enabled for any channel of ADCB.

Description: ADCB may malfunction with wrong conversion results when two conditions are met:
1. CTRL1[SMODE] is set to once parallel (001b), loop parallel (011b) or triggered parallel (101b) mode, and CTRL2[SIMULT] is set to zero, which means independent mode.
2. There is differential pair enabled among ANB0~ANB7 by setting bit3 or bit2 of CTRL1[CHNCFG_L] or CTRL2[CHNCFG_H].

Workaround: There are two methods to avoid this issue in independent parallel mode:
1. Make sure all the inputs of ADCB are configured as single ended inputs.
2. Make sure the following three conditions are met:
   - Enable sample 0 of ADCA by clearing bit 0 of SDIS register.
   - Disable the additional on-chip sample slots by setting 0xF to SDIS2.
   - Enable full differential mode for sample 8~15 by setting bit2&3 of CTRL1[CHNCFG_L] and CTRL2[CHNCFG_H] while keeping bit2&3 of CTRL3[UPDEN_H] and CTRL3[UPDEN_L] cleared. Or enable unipolar differential mode for sample 8~15 by setting bit2&3 of CTRL1[CHNCFG_L] and CTRL2[CHNCFG_H] while keeping bit2&3 of CTRL3[UPDEN_H] and CTRL3[UPDEN_L] set.

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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2022 NXP B.V.