Document identifier: MPC5604P_1M36W
Rev. JUL2023, 7/2023

MPC5604P_1M36W

Mask Set Errata

Mask Set Errata for Mask 1M36W

Revision History

This report applies to mask 1M36W for these products:

- MPC5604P
-
- MPC5604P_Pictus_082

Table 1. Revision History

Revision	Date	Significant Changes
JUL2023	7/2023	The following errata were added. • ERR051698
SEP2022	10/2022	The following errata were added. • ERR051065 The following errata were revised. • ERR007394
AUG2021	9/2021	The following errata were removed. • ERR002883 • ERR004146 • ERR006967 • ERR003028 The following errata were added. • ERR009976 • ERR010755 • ERR051059 • ERR051110 The following errata were revised. • ERR007394
1	8/2015	The following errata were removed. • ERR003258 • ERR003324 The following errata were added. • ERR006620 • ERR006726 • ERR006239

Table continues on the next page...

Table 1. Revision History (continued)

Revision	Date	Significant Changes
		• ERR008770 • ERR007394 • ERR006082 • ERR004146 • ERR006967 • ERR003442 • ERR005569 • ERR007877 • ERR007804 • ERR006802 • ERR006583 • ERR007322 The following errata were revised. • ERR002897 • ERR005203
2	2/2014	No changes to errata with this revision
1	11/2012	The following errata were added. • ERR003610 The following errata were revised. • ERR002656
0	9/2012	Initial Revision

Errata and Information Summary

Table 2. Errata and Information Summary

Erratum ID	Erratum Title
ERR000575	DSPI: Changing CTARs between frames in continuous PCS mode causes error
ERR001082	DSPI: set up enough ASC time when MTFE=1 and CPHA=1
ERR001103	DSPI: PCS Continuous Selection Format limitation
ERR001388	FlexRay: Incomplete Transmission of Message Frame in Key Slot
ERR002161	NPC: Automatic clock gating does not work for MCKO_DIV = 8
ERR002302	FlexRay: Message Buffer can not be disabled and not locked after CHI command FREEZE
ERR002360	FlexCAN: Global Masks misalignment
ERR002421	FLEXRAY : Message Buffer can not be disabled in POC state INTEGRATION_LISTEN

Table continues on the next page...

Table 2. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR002423	FlexRay: Transmission in a slot n of Channel A in dynamic segment may be corrupted or duplicated on both the channels
ERR002656	FlexCAN: Abort request blocks the CODE field
ERR002685	FlexCAN: Module Disable Mode functionality not described correctly
ERR002813	MC_RGM: Reset source flag not set correctly after previous clear
ERR002894	ADC: Minimum sampling time for ADC at 32MHz
ERR002897	ADC: ADC inaccuracy due to Digital Offset Cancellation Handling mechanism
ERR002958	MC_RGM: Clearing a flag at RGM_DES or RGM_FES register may be prevented by a reset
ERR002963	CMU XOSC Monitoring cannot be guaranteed when RCDIV>0 and FOSC is less than 1.5*Frc/ 2^rcdiv
ERR002964	Register Protection on full CMU_CSR
ERR002966	Serial Boot and Censorship: flash read access
ERR002971	ADC ABORT bit (single conversion)
ERR002972	ADC: Last conversion in chain not aborted
ERR002973	ADC ABORT CHAIN bit
ERR002977	MC_RGM: Long Reset Sequence Occurs on 'Short Functional' Reset Event
ERR002981	FMPLL: Do not poll flag FMPLL_CR[PLL_FAIL]
ERR002982	MCM: MRSR does not report Power On Reset event
ERR002997	ADC: Injected conversion not executed during scan mode.
ERR002999	MC_CGM and MC_PCU: A data storage exception is not generated on an access to MC_CGM or MC_PCU when the respective peripheral is disabled at MC_ME
ERR003001	MC_ME: ME_PSx registers may show '1' in a reserved bit field
ERR003010	ADC: conversion chain failing after ABORT chain
ERR003021	LINFlex: Unexpected LIN timeout in slave mode
ERR003022	SWT: Watchdog is disabled during BAM execution
ERR003032	Auto clock off feature does not work for adclksel=1
ERR003049	MC_RGM: External Reset not asserted if Short Reset enabled
ERR003060	MC_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared
ERR003069	ADC: ADC_DMAE[DCLR] set to 1 clears the DMA request incorrectly
ERR003094	MC_ME: Peripherals in unknown state after SAFE mode exit
ERR003110	Debugging functionality could be lost when unsecuring a secured device.
ERR003114	FLASH: Erroneous update of the ADR register in case of multiple ECC errors
ERR003164	FCU: Timeout feature doesn't work correctly.

Table continues on the next page...

Table 2. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR003165	BAM: Code download via FlexCAN not functioning in a CAN network
ERR003219	MC_CGM: System clock may stop for case when target clock source stops during clock switching transition
ERR003248	ADC:Abort request during last sampling cycle corrupts the data register of next channel conversion
ERR003256	eTimer: Configuring an eTimer counter channel in GATED-COUNT mode or in SIGNED-COUNT mode may cause a possible incorrect counting of 1 tick.
ERR003257	FlexPWM: Configuring the count value to set PWMA/PWMB first low and then high in the same cycle the output signals are low.
ERR003269	MC_ME: Peripheral clocks may get incorrectly disabled or enabled after entering debug mode
ERR003310	FlexPWM: PWM signals are improperly synced when using Master Sync
ERR003335	FlexPWM: Incorrect PWM operation when mixing DMA and non-DMA controlled channels
ERR003407	FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1
ERR003442	CMU monitor: FXOSC/FIRC and FMPLL/FIRC relation
ERR003511	FlexPWM : Incorrect PWM operation when IPOL is set.
ERR003579	Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge
ERR003583	MC_RGM: A non-monotonic ramp on the VDD_HV_REG supply can cause the RGM module to clear all flags in the DES register.
ERR003584	MC_ME: Possibility of Machine Check on Low-Power Mode Exit
ERR003610	FlexCAN: Wrong data transmission exiting from STOP mode in case EXTAL frequency is greater than IRC
ERR005113	ADC: triggering an ABORT or ABORTCHAIN before the conversion starts
ERR005203	ADC: "Abort command" aborts the ongoing injected channel as well as the upcoming normal channel.
ERR005569	ADC: The channel sequence order will be corrupted when a new normal conversion chain is started prior to completion of a pending normal conversion chain
ERR006082	LINFlexD : LINS bits in LIN Status Register(LINSR) are not usable in UART mode.
ERR006239	eTimer: When counter is not enabled, capture flags can be set but capture register values are not updated.
ERR006583	eTimer: Incorrect updating of the Hold Register
ERR006620	FLASH: ECC error reporting is disabled for Address Pipelining Control (APC) field greater than Read Wait-State Control (RWSC) field.
ERR006726	NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled
ERR006802	eTimer: Extra input capture events can set unwanted DMA requests

Table continues on the next page...

Table 2. Errata and Information Summary (continued)

Erratum ID	Erratum Title
ERR007322	FlexCAN: Bus Off Interrupt bit is erroneously asserted when soft reset is performed while FlexCAN is in Bus Off state
ERR007394	MC_ME: Incorrect mode may be entered on low-power mode exit.
ERR007804	LINFlex: Consecutive headers received by LIN Slave triggers error interrupt
ERR007877	FlexPWM: do not enable the fault filter
ERR008770	FlexRAY: Missing TX frames on Channel B when in dual channel mode and Channel A is disabled
ERR009976	DSPI: Incorrect data received by master with Modified transfer format enabled when using Continuous serial communication clock mode
ERR010755	DSPI: Transmit and Receive FIFO fill flags in status register is not cleared when DMA is improperly configured
ERR051059	DSPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event
ERR051065	SARADC: Interrupted Normal conversion chain may not be restored properly
ERR051110	FlexPWM: Half cycle automatic fault clearing does not work in PWM submodule 0
ERR051698	CTU : Double buffer reload mechanism is blocked when master reload pulse is not generated by Software

Known Errata

ERR000575: DSPI: Changing CTARs between frames in continuous PCS mode causes error

Description

Erroneous data could be transmitted if multiple Clock and Transfer Attribute Registers (CTAR) are used while using the Continuous Peripheral Chip Select mode (DSPIx_PUSHR[CONT=1]). The conditions that can generate an error are:

1) If DSPIx_CTARn[CPHA]=1 and DSPIx_MCR[CONT_SCKE = 0] and DSPIx_CTARn[CPOL, CPHA, PCSSCK or PBR] change between frames.

2) If DSPIx_CTARn[CPHA]=0 or DSPIx_MCR[CONT_SCKE = 1] and any bit field of DSPIx_CTARn changes between frames except DSPIx_CTARn[PBR].

Workaround

When generating DSPI bit frames in continuous PCS mode, adhere to the aforementioned conditions when changing DSPIx_CTARn bit fields between frames.

ERR001082: DSPI: set up enough ASC time when MTFE=1 and CPHA=1

Description

When the DSPI is being used in the Modified Transfer Format mode (DSPI_MCR[MTFE]=1) with the clock phase set for Data changing on the leading edge of the clock and captured on the following edge in the DSPI Clock and Transfer Attributes Register (DSPI_CTARn[CPHA]=1), if the After SCK delay scaler (ASC) time is set to less than 1/2 SCK clock period the DSPI may not complete the transaction - the TCF flag will not be set, serial data will not received, and last transmitted bit can be truncated.

Workaround

If the Modified Transfer Format mode is required DSPI_MCR[MTFE]=1 with the clock phase set for serial data changing on the leading edge of the clock and captured on the following edge in the SCK clock (Transfer Attributes Register (DSPI_CTARn[CPHA]=1) make sure that the ASC time is set to be longer than half SCK clock period.

ERR001103: DSPI: PCS Continuous Selection Format limitation

Description

When the DSPI module has more than one entry in the TX FIFO and only one entry is written and that entry has the CONT bit set, and continuous SCK clock selected the PCS levels may change between transfer complete and write of the next data to the DSPI_PUSHR register.

For example:

If the CONT bit is set with the first PUSHR write, the PCS de-asserts after the transfer because the configuration data for the next frame has already been fetched from the next (empty) fifo entry. This behavior continues till the buffer is filled once and all CONT bits are one.

Workaround

To insure PCS stability during data transmission in Continious Selection Format and Continious SCK clock enabled make sure that the data with reset CONT bit is written to DSPI_PUSHR register before previous data sub-frame (with CONT bit set) transfer is over.

ERR001388: FlexRay: Incomplete Transmission of Message Frame in Key Slot

Description

The FlexRay module will transmit an incomplete message in the key slot under the following circumstances:

1.) The transmit message buffer n assigned to the key slot is located in the message buffer segment 2, i.e. FR_MBSSUTR[MB_LAST_SEG1] < n, and

2.) the data size of the message buffer segment 1 is smaller than the static payload length, i.e. FR_MBDSR[MBSEG1DS] < PCR19[payload_length_static].

In this case, the FlexRay module will transmit only FR_MBDSR[MBSEG1DS] payload words from message buffer n. The remaining words are padded with 0's.

Workaround

The transmit message buffer assigned to key slot must be located in message buffer segment 1.

ERR002161: NPC: Automatic clock gating does not work for MCKO_DIV = 8

Description

If the Nexus clock divider (NPC_PCR[MCKO_DIV]) in the Nexus Port Controller Port Configuration Register is set to 8 and the Nexus clock gating control (NPC_PCR[MCKO_GT]) is enabled, the nexus clock (MCKO) will be disabled prior to the completion of transmission of the Nexus message data.

Workaround

Do not enable the automatic clock gating mode when the Nexus clock divider is set to 8. If Nexus clock gating is required, use a divide value of 1, 2 or 4 (set NPC_PCR[MCKO_GT]=0b1 and NPC_PCR[MCKO_DIV]=0b000, 0b001, or 0b011). However, MCKO must be kept below the maximum Nexus clock rate as defined in the device data sheet. If divide by 8 clock divider is required, then do not enable clock gating (set NPC_PCR[MCKO_GT]=0b0 and NPC_PCR[MCKO_DIV]=0b111).

ERR002302: FlexRay: Message Buffer can not be disabled and not locked after CHI command FREEZE

Description

If a complete message was transmitted from a transmit message buffer or received into a message buffer and the controller host interface (CHI) command FREEZE is issued by the application before the end of the current slot, then this message buffer can not be disabled and locked until the module has entered the protocol state normal active.

Consequently, this message buffer can not be disabled and locked by the application in the protocol config state, which prevents the application from clearing the commit bit CMT and the module from clearing the status bits. The configuration bits in the Message Buffer Configuration, Control, Status Registers (MBCCSRn) and the message buffer configuration registers MBCCFRn, MBFIDRn, and MBIDXRn are not affected.

At most one message buffer per channel is affected.

Workaround

There are two types of workaround.

1) The application should not send the CHI command FREEZE and use the CHI command HALT instead.

2) Before sending the CHI command FREEZE the application should repeatedly try to disable all message buffers until all message buffers are disabled. This maximum duration of this task is three static or three dynamic slots.

ERR002360: FlexCAN: Global Masks misalignment

Description

Convention: MSB=0.

During CAN messages reception by FlexCAN, the RXGMASK (Rx Global Mask) is used as acceptance mask for most of the Rx Message Buffers (MB). When the FIFO Enable bit in the FlexCAN Module Configuration Register (CANx_MCR[FEN], bit 2) is set, the RXGMASK also applies to most of the elements of the ID filter table. However there is a misalignment between the position of the ID field in the Rx MB and in RXIDA, RXIDB and RXIDC fields of the ID Tables. In fact RXIDA filter in the ID Tables is shifted one bit to the left from Rx MBs ID position as shown below:

Rx MB ID = bits 3-31 of ID word corresponding to message ID bits 0-28

RXIDA = bits 2-30 of ID Table corresponding to message ID bits 0-28

Note that the mask bits one-to-one correspondence occurs with the filters bits, not with the incoming message ID bits.

This leads the RXGMASK to affect Rx MB and Rx FIFO filtering in different ways.

For example, if the user intends to mask out the bit 24 of the ID filter of Message Buffers then the RXGMASK will be configured as 0xffff_ffef. As result, bit 24 of the ID field of the incoming message will be ignored during filtering process for Message Buffers. This very same configuration of RXGMASK would lead bit 24 of RXIDA to be "don't care" and thus bit 25 of the ID field of the incoming message would be ignored during filtering process for Rx FIFO.

Similarly, both RXIDB and RXIDC filters have multiple misalignments with regards to position of ID field in Rx MBs, which can lead to erroneous masking during filtering process for either Rx FIFO or MBs.

RX14MASK (Rx 14 Mask) and RX15MASK (Rx 15 Mask) have the same structure as the RXGMASK. This includes the misalignment problem between the position of the ID field in the Rx MBs and in RXIDA, RXIDB and RXIDC fields of the ID Tables.

Workaround

Therefore it is recommended that one of the following actions be taken to avoid problems:

) Do not enable the RxFIFO. If CANx_MCR[FEN]=0 then the Rx FIFO is disabled and thus the masks RXGMASK, RX14MASK and RX15MASK do not affect it. Do not enable the RxFIFO. If CANx_MCR[FEN]=0 then the Rx FIFO is disabled and thus the masks RXGMASK, RX14MASK and RX15MASK do not affect it.

) Enable Rx Individual Mask Registers. If the Backwards Compatibility Configuration bit in the FlexCAN Module Configuration Register (CANx_MCR[BCC], bit 15) is set then the Rx Individual Mask Registers (RXIMR0-63) are enabled and thus the masks RXGMASK, RX14MASK and RX15MASK are not used. Enable Rx Individual Mask Registers. If the Backwards Compatibility Configuration bit in the FlexCAN Module Configuration Register (CANx_MCR[BCC], bit 15) is set then the Rx Individual Mask Registers (RXIMR0-63) are enabled and thus the masks RXGMASK, RX14MASK and RX15MASK are not used.

) Do not use masks RXGMASK, RX14MASK and RX15MASK (i.e. let them in reset value which is 0xffff_ffff) when CANx_MCR[FEN]=1 and CANx_MCR[BCC]=0. In this case, filtering processes for both Rx MBs and Rx FIFO are not affected by those masks. Do not use masks RXGMASK, RX14MASK and RX15MASK (i.e. let them in reset value which is 0xffff_ffff) when CANx_MCR[FEN]=1 and CANx_MCR[BCC]=0. In this case, filtering processes for both Rx MBs and Rx FIFO are not affected by those masks.

) Do not configure any MB as Rx (i.e. let all MBs as either Tx or inactive) when CANx_MCR[FEN]=1 and CANx_MCR[BCC]=0. In this case, the masks RXGMASK, RX14MASK and RX15MASK can be used to affect ID Tables without affecting filtering process for Rx MBs. Do not configure any MB as Rx (i.e. let all MBs as either Tx or inactive) when CANx_MCR[FEN]=1 and CANx_MCR[BCC]=0. In this case, the masks RXGMASK, RX14MASK and RX15MASK can be used to affect ID Tables without affecting filtering process for Rx MBs.

ERR002421: FLEXRAY : Message Buffer can not be disabled in POC state INTEGRATION_LISTEN

Description

If the communication controller is started as a non-coldstart node and configured and enabled message buffers in the POS config state for slot 1, then the message buffer can not be disabled in the INTEGRATION_LISTEN state, which is entered when no communication can be established.

Workaround

A Software work-around is available, which is as follows:

Run a freeze command just before running the message buffer disable for slot 1. This should enable the message buffer disable during the Listen States.

ERR002423: FlexRay: Transmission in a slot n of Channel A in dynamic segment may be corrupted or duplicated on both the channels

Description

If a transmit message buffer is : 1) is assigned to a slot n in the dynamic segment and 2) assigned to both channels A and B; and if the slot ID n in the dynamic segment does not coincide for both channels, the FlexRay module:

a) will transmit the message frame on both channels A and B, or

b) may transmit a corrupted frame on channel A.

Workaround

Assign Message Buffers in the dynamic segment to one channel only. Note, the FlexRay specification (revision 2.1a) states that in the dynamic segment, transmit buffers should only be assigned channel A or channel B, and not to both.

ERR002656: FlexCAN: Abort request blocks the CODE field

Description

An Abort request to a transmit Message Buffer (TxMB) can block any write operation into its CODE field. Therefore, the TxMB cannot be aborted or deactivated until it completes a valid transmission (by winning the CAN bus arbitration and transmitting the contents of the TxMB).

Workaround

Instead of aborting the transmission, use deactivation instead.

Note that there is a chance that the deactivated TxMB can be transmitted without setting IFLAG and updating the CODE field if it is deactivated.

ERR002685: FlexCAN: Module Disable Mode functionality not described correctly

Description

Module Disable Mode functionality is described as the FlexCAN block is directly responsible for shutting down the clocks for both CAN Protocol Interface (CPI) and Message Buffer Management (MBM) sub-modules. In fact, FlexCAN requests this action to an external logic.

Workaround

In FlexCAN documentation chapter:

Section "Modes of Operation", bullet "Module Disable Mode":

Where is written:

"This low power mode is entered when the MDIS bit in the MCR Register is asserted. When disabled, the module shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules.."

The correct description is:

"This low power mode is entered when the MDIS bit in the MCR Register is asserted by the CPU. When disabled, the module requests to disable the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules."

Section "Modes of Operation Details", Sub-section "Module Disable Mode":

Where is written:

"This low power mode is entered when the MDIS bit in the MCR Register is asserted. If the module is disabled during Freeze Mode, it shuts down the clocks to the CPI and MBM sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit.."

The correct description is:

"This low power mode is entered when the MDIS bit in the MCR Register is asserted. If the module is disabled during Freeze Mode, it requests to disable the clocks to the CAN Protocol Interface (CPI) and Message Buffer Management (MBM) sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit."

ERR002813: MC_RGM: Reset source flag not set correctly after previous clear

Description

Bits in the RGM_FES and RGM_DES registers may not show the correct status if a reset event occurs after its corresponding flag has previously been cleared. This error occurs only if no other MC_RGM register access has taken place after clearing the status flag and before the reset event.

Workaround

Perform a MC_RGM register access (e.g. a 'dummy' read) directly after each write access of the RGM_FES and RGM_DES registers.

ERR002894: ADC: Minimum sampling time for ADC at 32MHz

Description

When ADC is running at 32MHz the minimum sampling time of 135ns specified is not met.

Workaround

At 32MHZ the minimum sampling time must be at least 180ns.

ERR002897: ADC: ADC inaccuracy due to Digital Offset Cancellation Handling mechanism

Description

The intrinsic ADC precision is better when used without the Digital Offset Cancellation Mechanism. For this reason, reference to the Digital Offset Cancellation (DOC) was removed from the reference manual. However, there is a possible side effect that may result in a small but potentially significant offset up to +/- 8 least significant bits. To remove this possible offset, the following cancellation steps should be followed.

Workaround

The following three step approach should be used to eliminate any potential offset add-on.

1. Run the offset cancellation routine

Steps:

a. Check that the ADC is in Power Down Mode in the ADC Status register MSR[ADCSTATUS] == 001

b. Configure ADC Conversion Time in the ADC Conversion time register – The value of this register depends on the ADC clock.

c. Enable Offset Cancellation by setting the offset cancellation enable bit in the ADC Module Control Register MCR[OFFCANC]=1

d. Power up ADC by clearing the MCR[PWDN] bit and wait for MSR[ADCSTATUS] == 000 (IDLE state)

e. Wait until ADC Offset Cancellation completes MSR[OFFCANC] == 0 or until the ISR[EOFFSET] is set. If the offset error cancellation occurs, the ISR[EOFFSET] flag is set and the OFFCANC bit of MCR and MSR registers remain at 1. The MCR[OFFCANC] bit can be cleared by software to abort the DOC sequence, while the MSR[OFFCANC] flag cannot be cleared until the next reset.

f. When ISR [EOFFSET] do these: Abort DOC procedure: Clear MCR[OFFCANC], Wait for MSR[ADCSTATUS] == 000 (IDLE state) and clear the ISR[OFFCANC] flag.

2. Overwrite the digital cancellation result to 0 in the Offset Word Register (OFFWR Register)

Steps:

a. Enable Offset Register setting/write OFFWR[OFFSET_LOAD] bit

b. Write OFFWR[OFFSET_WORD] field with 0 with setting OFFWR[OFFSET_LOAD] bit

3. Load the new value of OFFSET_WORD instead of the one calculated from DOC mechanism into internal offset register which is buffered and disable possibility to overwrite the OFFSET_WORD. There is necessary to create one dummy conversion by sampling at least one channel.

a) Enable sampling one channel in the NCMR[0].R.

b) Start conversion by setting MCR[NSTART] or MCR[JSTART] bit.

c) Check if the conversation is finished in the main status register MSR[NSTART].

d) Disable the load of offset register OFFWR[OFFSETLOAD].

Register description

Registers and fields related to Digital Offset Cancellation procedure have been removed from documentation. In the following they are described in order to allow applying the software workaround described.

MCR register

Table:

Bit	Bit Name	Description
28	OFFCANC	0: No offset cancellation phase, 1: Offset cancellation phase before conversion, This bit is reset to zero when the Offset Cancellation ends

MSR register

Table:

Bit	Bit Name	Description

Table continues on the next page...

28	OFFCANC	0: Offset cancellation phase has been completed, 1: Offset cancellation phase is ongoing

OFFSET WORD REGISTER (OFFWR)

Addr: Base +0x00C0

Table:

Bit	Bit Name	Description
0:14	Reserved	Write of any value has no effect, read value is always zero.
15	OFFSET LOAD	0: Disable offset loading, 1: Enable offset loading, that bit should be written before writing OFFSET_WORD field.
16:23	Reserved	Write of any value has no effect, read value is always zero.
24:31	OFFSET_WORD	The offset word coefficient generated at the end of the Digital Offset Cancellation phase is latched into this register. The offset word can also be written by software. In this case, it is loaded into analog ADC and used as offset cancellation word instead of the one calculated using offset cancellation process.

INTERRUPT STATUS REGISTER (ISR)

Table:

Bit	Bit Name	Description

| 25 | OFFCANCOVR | Offset Cancellation Phase Over

When the ADC generates the offset_ok_i to high indication then the offset cancellation phase programmed by the user is over and the offset coefficient is written into the offset register. |

Table:

26	EOFFSET	This interrupt is generated during the offset cancellation phase in case the offset_measure_ok_i pulse is not received.

Note: Both bits are cleared by writing one to their position.

ERR002958: MC_RGM: Clearing a flag at RGM_DES or RGM_FES register may be prevented by a reset

Description

Clearing a flag at RGM_DES and RGM_FES registers requires two clock cycles because of a synchronization mechanism. As a consequence if a reset occurs while clearing is on-going the reset may interrupt the clearing mechanism leaving the flag set.

Note that this failed clearing has no impact on further flag clearing requests.

Workaround

No workaround for all reset sources except SW reset.

Note that in case the application requests a SW reset immediately after clearing a flag in RGM_xES the same issue may occur. To avoid this effect the application must ensure that flag clearing has completed by reading the RGM_xES register before the SW reset is requested.

ERR002963: CMU XOSC Monitoring cannot be guaranteed when RCDIV>0 and FOSC is less than 1.5*Frc/2^rcdiv

Description

A False OLRI (Oscillator frequency less than RC frequency) event may be generated when FOSC is less than 1.5*(FRC / 2^RCDIV) and RCDIV>0, and not FRC/ 2^RCDIV as described in the Reference Manual).

Correct crystal clock monitoring is guaranteed when FOSC is strictly above 1.5*(FRC /2^RCDIV).

Workaround

There are 2 workarounds available :

1. Keep RCDIV = 0

2. When RCDIV > 0, ensure FOSC is greater than FRC/2^(RCDIV -1) to avoid false OLRI (CMU external oscillator failure) event.

ERR002964: Register Protection on full CMU_CSR

Description

The register protection on CMU_CSR of CMU0 works only on the full 32 bit, while it should protect only the bits 24-31.

As a consequence, when register protection is active on CMU_CSR the frequency meter cannot be used anymore.

This issue is also present on CMU1.

Workaround

In order to perform a frequency meter operation, the register protection of the relevant CMU must be disabled first; this workaround would work only when soft lock is active.

ERR002966: Serial Boot and Censorship: flash read access

Description

In a secured device, starting with a serial boot, it is possible to read the content of the four flash locations where the RCHW is stored.

For example if the RCHW is stored at address 0x00000000, the reads at address 0x00000000, 0x00000004, 0x00000008 and 0x0000000C will return a correct value.

Any other flash address is not readable.

Workaround

None

ERR002971: ADC ABORT bit (single conversion)

Description

If the User starts one single ADC conversion and, after, starts a chain of conversions, when the User tries to abort one of the chained conversions, the abort command aborts the chain instead of a single conversion of the chain.

Workaround

- A minimum of two conversions must be programmed.

- A dummy conversion must be started before or after a single conversion.

ERR002972: ADC: Last conversion in chain not aborted

Description

If the user aborts the last ADC conversion of a chain the conversion appears to be aborted but the relevant data register is updated with the conversion data.

Workaround

None

ERR002973: ADC ABORT CHAIN bit

Description

If user aborts a chain of ADC conversions, the current conversion appears as aborted but the relative data register is however updated.

Workaround

None

ERR002977: MC_RGM: Long Reset Sequence Occurs on 'Short Functional' Reset Event

Description

If a 'functional' reset source is configured to initiate a short reset sequence via setting of the appropriate RGM_FESS bit and also configured to assert the external reset pad via setting of the appropriate RGM_FBRE bit, its assertion will not initiate a short reset starting with PHASE3 but rather a long reset sequence starting with PHASE1.

Workaround

Do not configure 'functional' resets that are configured to initiate a short reset sequence to also assert the external reset pad. In other words, if RGM_FESS[i] is '1', RGM_FBRE[i] should be '0'.

ERR002981: FMPLL: Do not poll flag FMPLL_CR[PLL_FAIL]

Description

For the case when the FMPLL is indicating loss of lock the flag FMPLL_CR[PLL_FAIL] is unpredictable.

Workaround

To avoid reading an incorrect value of FMPLL_CR[PLL_FAIL] only read this flag inside the FMPLL Interrupt Service Routine (ISR). The ISR indicates the flag has been set correctly and at this point the flag can be cleared. Do not poll flag FMPLL_CR[PLL_FAIL] at any other point in the application software.

ERR002982: MCM: MRSR does not report Power On Reset event

Description

The flag POR of MRSR register stays low after Power On Reset event on the device.

Workaround

Do not use MRSR[POR] to determine Power on reset cause. Use RGM_DES instead.

ERR002997: ADC: Injected conversion not executed during scan mode.

Description

When ADC is converting a chain in scan mode -configured using NSTART bit in non-CTU mode

operation; and a injected conversion arrives -triggered by software with JSTART bit or by hardware from eTimer_1 channel 5 (internal connection); the ADC gets stuck in the sampling phase (the triggered conversion is not executed and the chain is not restarted).

Workaround

None

ERR002999: MC_CGM and MC_PCU: A data storage exception is not generated on an access to MC_CGM or MC_PCU when the respective peripheral is disabled at MC_ME

Description

If a peripheral with registers mapped to MC_CGM or MC_PCU address spaces is disabled via the MC_ME any read or write accesses to this peripheral will be ignored without producing a data storage exception.

Workaround

For any mode other than a low-power mode do not disable any peripheral that is mapped to MC_CGM or MC_PCU.

ERR003001: MC_ME: ME_PSx registers may show '1' in a reserved bit field

Description

Some bits in the ME_PSx registers that are defined to always return the value '0' may instead return the value '1'.

Workaround

It is recommended as a general rule that the User should always ignore the read value of reserved bit fields.

ERR003010: ADC: conversion chain failing after ABORT chain

Description

During a chain conversion while the ADC is in scan mode when ADC_MCR[ABORTCHAIN] is asserted the current chain will be aborted as expected. However, in the next scan the first conversion of the chain is performed twice and the last conversion of the chain is not performed.

Workaround

When aborting a chain conversion enable ADC_MCR[ABORTCHAIN] and disable ADC_MCR[START].

ADC_MCR[START] can be enabled when the abort is complete.

ERR003021: LINFlex: Unexpected LIN timeout in slave mode

Description

If the LINFlex is configured in LIN slave mode, an unexpected LIN timeout event (LINESR[OCF]) may occur during LIN Break reception.

Workaround

It is recommended to disable this functionality during LINFlex initialization by clearing LINTCSR[IOT] and LINIER[OCIE] bits, and ignore timeout events.

ERR003022: SWT: Watchdog is disabled during BAM execution

Description

The watchdog is disabled at the start of BAM execution. In the case of an unexpected issue during BAM execution the CPU may be stalled and it will be necessary to generate an external reset to recover.

Workaround

None

ERR003032: Auto clock off feature does not work for adclksel=1

Description

Auto clock off feature in which clock to ADC analog is automatically stopped when in pwdn mode or in IDLE mode with no conversion ongoing does not work when ADC analog clock is same as ADCDigital interface clock i.e. adclksel = 1.

This means a little higher power consumption when this configuration is used.

Workaround

None

ERR003049: MC_RGM: External Reset not asserted if Short Reset enabled

Description

For the case when the External Reset is enabled for a specific reset source at RGM_FBRE and a short reset is requested for the same reset source at RGM_FESS the External Reset is not asserted.

Workaround

None

ERR003060: MC_RGM: SAFE mode exit may be possible even though condition causing the SAFE mode request has not been cleared

Description

A SAFE mode exit should not be possible as long as any condition that caused a SAFE mode entry is still active. However, if the corresponding status flag in the RGM_FES register has been cleared, the SAFE mode exit may incorrectly occur even though the actual condition is still active.

Workaround

Software must clear the SAFE mode request condition at the source before clearing the corresponding RGM_FES flag. This will ensure that the condition is no longer active when the RGM_FES flag is cleared and thus the SAFE mode exit can occur under the correct conditions.

ERR003069: ADC: ADC_DMAE[DCLR] set to 1 clears the DMA request incorrectly

Description

When ADC_DMAE[DCLR] is set the DMA request should be cleared only after the data registers are read. However for this case the DMA request is automatically cleared and will not be recognised by the eDMA.

Workaround

None

ERR003094: MC_ME: Peripherals in unknown state after SAFE mode exit

Description

Peripherals that reside in auxiliary clock domains may be in an unknown state after exiting the SAFE mode to enter the DRUN mode.

Workaround

Execute a software reset while in the SAFE mode if one or more peripherals present in auxiliary clock domains is required for further operation.

ERR003110: Debugging functionality could be lost when unsecuring a secured device.

Description

Providing the backdoor password via JTAG or via serial boot would unsecure the device, but on some devices may leave the Nexus interface and potentially the CPUs in an undetermined state.

Normal operation without a debugger is unaffected, debugging unsecured devices is also unaffected.

Workaround

A second connection attempt may be successful.

Boot in serial mode (using the Flash password), then execute code which unsecures the device.(The JTAG interface needs to be inactive while the unsecure happens.)

Implement a separate backdoor in application software. Once the software detects the custom backdoor sequence it can unlock the device via Flash write.

Leave device unsecured for debugging.

ERR003114: FLASH: Erroneous update of the ADR register in case of multiple ECC errors

Description

An erroneous update of the Address register (ADR) occurs whenever there is a sequence of 3 or more events affecting the ADR (ECC single or double bit errors or RWW error) and both the following conditions apply:

- The priorities are ordered in such a way that only the first event should update ADR.

- The last event although it does not update ADR sets the Read While Write Event Error (RWE) or the ECC Data Correction (EDC) in the Module Configuration Register (MCR).

For this case the ADR is wrongly updated with the address related to one of the intervening events.

Example - If a sequence of two double-bit ECC errors is followed by a single-bit correction without clearing the ECC Event Error flag (EER) in the MCR, then the value found in ADR after the single-bit correction event is the one related to the second double-bit error (instead of the first one, as specified)

Workaround

Always process Flash ECC errors as soon as they are detected.

Clear MCR[RWE] at the end of each flash operation (Program, Erase, Array Integrity Check, etc...).

ERR003164: FCU: Timeout feature doesn't work correctly.

Description

A fault occurs, timeout for this fault is enabled and the fault is not recovered automatically before the timeout :- In this case device will go to ALARM state and waits for timeout, after timeout has elapsed it enters to FAULT state. However, if another fault occurs with timeout enabled the FCU will go directly to FAULT state without waiting for timeout to be elapsed.

If it is desired that FCU will go again to ALARM state for the second fault, a Watchdog reset needs to be asserted after first fault is detected.

Workaround

None

ERR003165: BAM: Code download via FlexCAN not functioning in a CAN network

Description

When the serial download via FlexCAN is selected setting the FAB (Force Alternate Boot) pin, and ABS (Alternate Boot Selector) pins (ABS0 = 1 and ABS1 = 0) and the micro is part of a CAN network, the serial download protocol may unexpectedly stop in case of CAN traffic.

After the code has been downloaded, the BAM tries to disable the FLexCAN module writing the MCR (Module Configuration Register) without waiting for the acknowledge bit LPM_ACK (Low Power Mode Acknowledge) to be set. The FlexCAN cannot enter the low power mode until all current transmissions or receptions have finished. Further writings into any FlexCAN register may cause the low power mode not to be entered and, as consequence, the BAM to stop.

Workaround

Since the higher the traffic, the higher the chance for the BAM to try to disable the FlexCAN module during a CAN frame reception, make sure that no other CAN frame is sent until the code download protocol has been completed.

ERR003219: MC_CGM: System clock may stop for case when target clock source stops during clock switching transition

Description

The clock switching is a two step process. The availability of the target clock is first verified. Then the system clock is switched to the new target clock source within two target clock cycles.

For the case when the FXOSC stops during the required two cycles, the switching process may not complete, causing the system clock to stop and prevent further clock switching. This may happen if one of the following cases occurs while the system clock source is switching to FXOSC:

- FXOSC oscillator failure

- SAFE mode request occurs, as this mode will immediately switch OFF the FXOSC (refer to ME_SAFE_MC register configuration)

Workaround

The device is able to recover through any reset event ('functional', 'destructive', internal or external), so typically either the SWT (internal watchdog) will generate a reset or, in case it is used in the application, the external watchdog will generate an external reset. In all cases the devices will restart properly after reset.

To reduce the probability that this issue occurs in the application, disable SAFE mode transitions when the device is executing a mode transition with the FXOSC as the system clock source in the target mode.

ERR003248: ADC:Abort request during last sampling cycle corrupts the data register of next channel conversion

Description

If the abort pulse is valid in the last cycle of the SAMPLE phase, the current channel is correctly aborted but the data register (CDR[0..15]) of the next channel conversion shows an invalid value.

Workaround

None

ERR003256: eTimer: Configuring an eTimer counter channel in GATED-COUNT mode or in SIGNED-COUNT mode may cause a possible incorrect counting of 1 tick.

Description

This bug affects eTimer in the following counting modes:

- GATED-COUNT mode, the CNTMODE field is '011' (count rising edges of primary source while secondary input is high);

-SIGNED-COUNT mode, the CNTMODE field is '101' (count primary source rising edges, secondary source specifies direction (up/down)).

Delays in the edge detection circuitry lead to a bug where the rising edge on the primary source is compared to the secondary source value one clock later in time. This means that if there is a rising edge on the primary source followed immediately by the secondary source going high, the eTimer logic could see this as a rising primary edge while the secondary is high even though the secondary input was low at the time of the rising primary edge.

► Note: The counter will also increment if the primary source is already high when the secondary source goes high.

The inputs are sampled by MOTC_CLK and a transition detector finds the rising edge.It is not an asynchronous counter.Also, transitions are limited to one rising or falling edge per clock period. The primary and secondary inputs come into the Timer and are sampled. The primary input is checked for a rising edge which takes another clock period. This extra cycle of delay on only the primary input is the problem.

If the primary source transitions high while the secondary is low, then no counting should occur. If the secondary source transitions high in the clock cycle after the primary source transition, then the extra delay to find the rising edge will confuse the counter and it will think , that the primary rising edge occurred while the secondary was high and will increment the counter.

If the transitions of the primary and secondary sources are more than one clock period apart, then there will be no incorrect counting.

► Possibly, the user will accept that if the edges occur this close together, then its okay to consider it as a countable occurrence, this is application dependant and the responsibility of the user.

Workaround

The source selected as the secondary input to the eTimer channel needs to have an additional clock cycle of delay added to it. This can be achieved by using the input filters.

For the primary source set FILT_PER==1 and FILT_CNT==0.

For the secondary source set FILT_PER==1 and FILT_CNT==1.

This will introduce a 5 clock cycle latency on the primary source and a 6 cycle latency on the secondary source which will properly align the two signals for Gated and Signed count modes.

Note: Ensure that the primary source is low before the secondary source goes high to avoid a false count.

Note: This work-around is only valid when the sources are external an pass through the input filter, internal signals have no work-around.

ERR003257: FlexPWM: Configuring the count value to set PWMA/PWMB first low and then high in the same cycle the output signals are low.

Description

The VAL2 and VAL4 registers define the turn-on edge and the VAL3 and VAL5 registers define the turn off edge of the PWMA/PWMB signals respectively. VAL3 cannot be less than VAL2 and VAL5 cannot be less than VAL4. Doing so will cause the PWM signal to turn off at the correct time (VAL3 or VAL5), but it will not turn on at the time defined by VAL2 or VAL4.

This can be an issue during the generation of phase delayed pulses where the PWM signal goes high late in PWM cycle N and remains high across the cycle boundary before going low early in cycle N+1 and goes high again in PWM cycle N+1. This errata will allow that to happen.

VAL3 register must be "greater than or equal to" VAL2 register and VAL5 must be "greater than or equal to" VAL4.

Workaround

None

ERR003269: MC_ME: Peripheral clocks may get incorrectly disabled or enabled after entering debug mode

Description

If ME_RUN_PCx, ME_LP_PCx, ME_PCTLx registers are changed to enable or disable a peripheral, and the device enters debug mode before a subsequent mode transition, the peripheral clock gets enabled or disabled according to the new configuration programmed. Also ME_PSx registers will report incorrect status as the peripheral clock status is not expected to change on debug mode entry.

Workaround

After modifying any of the ME_RUN_PCx, ME_LP_PCx, ME_PCTLx registers, request a mode change and wait for the mode change to be completed before entering debug mode in order to have consistent behaviour on peripheral clock control process and clock status reporting in the ME_PSx registers.

ERR003310: FlexPWM: PWM signals are improperly synced when using Master Sync

Description

If Master Sync signal , originated as the Local Sync from sub-module 0, is selected as the counter initialization signal for sub-modules 1-2-3 (slaves), with a prescaler PRSC < 0x2 on PWM clock, the slave sub-module PWM outputs are delayed approx 2 IP clock against sub-module0.

For PRSC > 0x1 the delay on slave sub-modules disappears.

Workaround

If Master Sync signal is requested, use a prescaler value PRSC >=0x2 to synchronize PWM outputs of Sub-module 0 to slave sub-modules PWM outputs, or don't use the sub-module 0 channel A and B.

ERR003335: FlexPWM: Incorrect PWM operation when mixing DMA and non-DMA controlled channels

Description

When some submodules use DMA to load their VALx registers and other submodules use non-DMA means that means direct writes from the CPU, the LDOK bits for the non-DMA submodules can be incorrectly cleared at the completion of the DMA controlled load cycle. This leads to the non-DMA channels not being properly updated.

Submodules that use DMA to read the input capture registers do not cause a problem for non-DMA submodules.

Workaround

Set the DMA enable bit to 1 also for non-DMA submodules, according to this the DMA will not incorrectly clear the LDOK bit for non-DMA submodules but they will be set to 1 at the end of each DMA cycle. When the CPU has to update the VALx registers of non-DMA submodules, first clear LDOK bit for non-DMA submodules

ERR003407: FlexCAN: CAN Transmitter Stall in case of no Remote Frame in response to Tx packet with RTR=1

Description

FlexCAN does not transmit an expected message when the same node detects an incoming Remote Request message asking for any remote answer.

The issue happens when two specific conditions occur:

1) The Message Buffer (MB) configured for remote answer (with code "a") is the last MB. The last MB is specified by Maximum MB field in the Module Configuration Register (MCR[MAXMB]).

2) The incoming Remote Request message does not match its ID against the last MB ID.

While an incoming Remote Request message is being received, the FlexCAN also scans the transmit (Tx) MBs to select the one with the higher priority for the next bus arbitration. It is expected that by the Intermission field it ends up with a selected candidate (winner). The coincidence of conditions (1) and (2) above creates an internal corner case that cancels the Tx winner and therefore no message will be selected for transmission in the next frame. This gives the appearance that the FlexCAN transmitter is stalled or "stops transmitting".

The problem can be detectable 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:

a) The incoming message matches the remote answer MB with code "a".

b) The MB configured as remote answer with code "a" is not the last one.

c) Any MB (despite of being Tx or Rx) is reconfigured (by writing its CS field) just after the Intermission field.

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

Workaround

Do not configure the last MB as a Remote Answer (with code "a").

ERR003442: CMU monitor: FXOSC/FIRC and FMPLL/FIRC relation

Description

Functional CMU monitoring can only be guaranteed when the following conditions are met:

- FXOSC frequency must be greater than (FIRC / 2^RCDIV) + 0.5MHz in order to guarantee correct FXOSC monitoring

- FMPLL frequency must be greater than (FIRC / 4) + 0.5MHz in order to guarantee correct FMPLL monitoring

Workaround

Refer to description

ERR003511: FlexPWM : Incorrect PWM operation when IPOL is set.

Description

When IPOL bit is set in complementary mode, the source of the PWMi waveform is supposed to be switched from the VAL2 and VAL3 registers to the VAL4 and VAL5 registers. This switch is not happening.

Workaround

Instead of setting IPOL bt, the application can swap the VAL2/3 values with the VAL4/5 values.

ERR003579: Nexus pins may drive an unknown value immediately after power up but before the 1st clock edge

Description

The Nexus Output pins (Message Data outputs 0:3 [MDO] and Message Start/End outputs 0:1 [MSEO]) may drive an unknown value (high or low) immediately after power up but before the 1st clock edge propagates through the device (instead of being

weakly pulled low). This may cause high currents if the pins are tied directly to a supply/ground or any low resistance driver (when used as a general purpose input [GPI] in the application).

Workaround

1. Do not tie the Nexus output pins directly to ground or a power supply.

2. If these pins are used as GPI, limit the current to the ability of the regulator supply to guarantee correct start up of the power supply. Each pin may draw upto few hundred mA current.

If not used, the pins may be left unconnected.

ERR003583: MC_RGM: A non-monotonic ramp on the VDD_HV_REG supply can cause the RGM module to clear all flags in the DES register.

Description

At system power-up a non-monotonic voltage ramp-up or a very slow voltage ramp-up (also known as ‘soft start-up’) can cause incorrect flag setting in the RGM_DES register.

During monotonic power-up, F_POR flag is set when the high voltage regulator supply (VDD_HV_REG) goes above LVD27_VREG low voltage detector threshold and the 1.2V supply (VDD_LV_REGCOR) goes above LVD12_PD0 low voltage detector threshold. Expected behavior POR =1 , LVD27 =0, LVD12=0

During a non-monotonic power-up the VDD_HV_REG may show a non-linearity in the ramp up. When the VDD_HV_REG supply dips below LVD27_VREG threshold, LVD27_VREG low voltage detector is re-fired. If VDD_LV_REGCOR is already above LVD12_PD0 low voltage detector threshold, F_POR flag is reset and F_LVD27_VREG is set. Expected behavior POR=0 , LVD27 =1, LVD12=0

This errata reports behavior when the non-linearity on VDD_HV_REG coincides with the ramp-up of VDD_LV_REGCOR completion and LVD27_VREG is re-fired just after the LVD12_PD0 is released. In this case, neither F_POR flag nor F_LVD27_VREG flag will be set. In this case, application code cannot use the flags to tell if a power-on reset has occurred.

This errata only affects the flag circuit and not a the device initialization. Device initializes correctly under all conditions.

Workaround

Hardware Workaround

Ensure that non-linearity on VDD_HV_REG is < 100mV . This is the hysteresis limit of the device. Board regulator should be chosen accordingly.

Software Workaround

The software workaround need only be applied when neither the F_POR, LVD27 or LVD12 flags are set and involves checking SRAM contents and monitoring for ECC errors during this process. In all cases, an ECC error is assumed to signify a power-on reset (POR).

Two suggestions are made for software workarounds. In both cases, if POR is detected all RAM should be initialized otherwise no power-on condition is detected and it is possible to initialize only needed parts of RAM while preserving required information.

Software workaround #1 :

An area of RAM can be reserved by the compiler into which a KEY, such as 0x3EC1_9678, is written. This area can be checked at boot and if the KEY is incorrect or an ECC error occurs, POR can be assumed and the KEY should be set. Use of a KEY increases detection rate to 31 bits(=<10e-9) or 23bits (= <5.10e-6) instead of 7-bit linked to ECC (=<10e-2)

Software workaround #2 :

When runtime data should be retained and RAM only re-initialized in the case of POR, the CRC module should be used to calculate and store a CRC signature when writing data that can be checked at boot time. If CRC signature is incorrect, POR can be assumed.

ERR003584: MC_ME: Possibility of Machine Check on Low-Power Mode Exit

Description

When executing from the flash and entering a Low-Power Mode (LPM) where the flash is in low-power or power-down mode, 2-4 clock cycles exist at the beginning of the RUNx to LPM transition during which a wakeup or interrupt will generate a machine check due to the flash not being available on RUNx mode re-entry. This will cause either a checkstop reset or machine check interrupt.

Workaround

This issue can be handled in one of the following ways. Workaround #1 configures the application to handle the machine check interrupt in RAM dealing with the problem if it occurs. Workaround #2 configures the MCU to avoid the machine check interrupt.

Workaround #1

The application can be configured to handle the machine check interrupt in RAM; when this occurs, the Mode Entry module can be used to bring up the flash normally and resume execution. Before stop mode entry, ensure the following:

1. Enable the Machine Check interrupt at the core MSR[ME] – this prevents a machine check reset occurring

2. Copy IVOR vector table to RAM

3. Point IVPR to vector table in RAM

4. Implement Machine Check interrupt handler in RAM to power-cycle flash to synchronise status of flash between Mode Entry and flash module

The interrupt handler should perform the following steps:

1. Test Machine Check at LPM exit due to wakeup/interrupt event

2. ME_RUNx_MC[CFLAON] = 0b01 (power-down)

3. Re-enter mode RUNx (x = 0,1,2,3) to power down flash

4. Wait for transition to RUNx mode to complete (ME_GS[S_MTRANS] = 1)

5. ME_RUNx_MC[CFLAON] = 0b11 (normal)

6. Re-enter mode RUNx (x = previous x) to power up flash

7. Wait for transition to RUNx mode to complete (ME_GS[S_MTRANS] = 1)

8. On completion, code execution will return to flash (via se_rfci)

Workaround #2

The application can be configured to avoid the machine check interrupt; low-power mode can be entered from a RAM function and Mode Entry configured to have flash off on return to the current RUNx mode. Flash can then be re-enabled by Mode Entry within the RAM function before returning to execution from flash.

1. Prior to LPM mode entry request branch to code execution in RAM while flash is still in normal mode

2. Set ME_RUNx_MC[CFLAON] = 0b01 (power-down) or 0b10 (low-power) for STOP0/HALT0

3. Set ME_STOP0/HALT0_MC[CFLAON] = ME_RUNx_MC[CFLAON]

4. Enter STOP0/HALT0 mode

5. At wakeup or interrupt from STOP0/HALT0, MCU enters RUNx mode executing from RAM with flash in low-power or power-down as per the ME_RUNx_MC configuration from step 2.

6. After the STOP0/HALT0 request, set ME_RUNx_MC[CFLAON] = 0b11 (normal)

7. Enter RUNx mode

8. Wait for transition to RUNx mode to complete (ME_GS[S_MTRANS] = 0)

9. Return to code execution in flash

ERR003610: FlexCAN: Wrong data transmission exiting from STOP mode in case EXTAL frequency is greater than IRC

Description

The FlexCAN module has got a programmable clock source that can be either the system clock (SYS_CLK) or oscillator clock (XOSC_CLK) selected by CLK_SRC bit in the FlexCAN_CTRL register.

In case FlexCAN module has selected the oscillator clock as clock source and XOSC_CLK is bigger than IRC frequency @16MHz and the system clock is PLL_CLK if the device enters STOP mode and the FlexCAN module is in transmission then when device exits from STOP mode the FlexCAN module can transmit wrong data.

This behavior happens because during STOP mode exit, SYS_CLK will be IRC @16MHz till PLL gets locked and if a frame transmission happens during this time itself then there will be a CAN Spec violation.

The FlexCAN module clock source should not be faster than SYS_CLK.

Workaround

Just before entering/requesting the STOP mode, set the “FRZ" and “HALT" bit of CAN_MCR register to ‘1' to request for freeze mode.

During the STOP mode exit, check for the mode transition completion. As mode transition will be over, only when all clock sources are on and the PLL is selected as system clock, unfreeze the CAN by resetting the “FRZ" or “HALT" bit.

ERR005113: ADC: triggering an ABORT or ABORTCHAIN before the conversion starts

Description

When ABORTCHAIN is programmed (ADC_MCR[ABORTCHAIN] = 1) and an injected chain conversion is programmed afterwards, the injected chain is aborted, but neither ADC_ISR[JECH] is set, nor ADC_MCR [ABORTCHAIN] is reset.

When ABORT is programmed (ADC_MCR[ABORT] = 1) and normal/injected chain conversion comes afterwards, the ABORT bit is reset and chain conversion runs without a channel abort.

If ABORT or ABORTCHAIN feature is programmed after the start of the chain conversion, it works properly.

Workaround

Do not program ADC_MCR[ABORT]/ ADC_MCR[ABORTCHAIN] before starting the execution of the chain conversion.

ERR005203: ADC: "Abort command" aborts the ongoing injected channel as well as the upcoming normal channel.

Description

If an Injected chain (jch1,jch2,jch3) is injected over a Normal chain (nch1,nch2,nch3,nch4) the Abort switch does not behave as expected.

Expected behavior:

* Correct Case (without SW Abort on jch3): Nch1 -> Nch2(aborted) -> Jch1 -> Jch2 -> Jch3 -> Nch2(restored) -> Nch3 -> Nch4

* Correct Case(with SW Abort on jch3): Nch1 -> Nch2(aborted) -> Jch1 -> Jch2 -> Jch3(aborted) -> Nch2(restored) -> Nch3 -> Nch4

Observed unexpected behavior:

* Fault1 (without SW abort on jch3): Nch1 -> Nch2(aborted) -> Jch1 -> Jch2 -> Jch3 -> Nch3 -> Nch4 (Nch2 not restored)

* Fault2 (with SW abort on jch3): Nch1 -> Nch2 (aborted) -> Jch1 -> Jch2 -> Jch3(aborted) -> Nch4 (Nch2 not restored & Nch3 conversion skipped)

Workaround

It is possible to detect the unexpected behavior by using the ADC_CDATA[x] register. The ADC_CDATAx[VALID] fields will not be set for a not restored or skipped channel, which indicates this issue has occurred. The ADC_CDATAx[VALID] fields need to be checked before the next Normal chain execution (provided ADC_MCR[OWREN] bit is set in scan mode). The ADC_CDATAx[VALID] fields should be read by every ECH interrupt at the end of every chain execution.

ERR005569: ADC: The channel sequence order will be corrupted when a new normal conversion chain is started prior to completion of a pending normal conversion chain

Description

If One shot mode is configured in the Main Configuration Register (MCR[MODE] = 0) the chained channels are automatically enabled in the Normal Conversion Mask Register 0 (NCMR0). If the programmer initiates a new chain normal conversion, by setting MCR[NSTART] = 0x1, before the previous chain conversion finishes, the new chained normal conversion will not follow the requested sequence of converted channels.

For example, if a chained normal conversion sequence includes three channels in following sequence: channel0, channel1 and channel2, the conversion sequence is started by MCR[NSTART] = 0x1. The software re-starts the next conversion sequence when MCR[NSTART] is set to 0x1 just before the current conversion sequence finishes.

The conversion sequence should be: channel0, channel1, channel2, channel0, channel1, channel2.

However, the conversion sequence observed will be: channel0, channel1, channel2, channel1, channel1, channel2. Channel0 is replaced by channel1 in the second chain conversion and channel1 is converted twice.

Workaround

Ensure a new conversion sequence is not started when a current conversion is ongoing. This can be ensured by issuing the new conversion setting MCR[NSTART] only when MSR[NSTART] = 0.

Note: MSR[NSTART] indicates the present status of conversion. MSR[NSTART] = 1 means that a conversion is ongoing and MSR[NSTART] = 0 means that the previous conversion is finished.

ERR006082: LINFlexD : LINS bits in LIN Status Register(LINSR) are not usable in UART mode.

Description

When the LINFlexD module is used in the Universal Asynchronous Receiver/Transmitter (UART) mode, the LIN state bits (LINS3:0]) in LIN Status Register (LINSR) always indicate the value zero. Therefore, these bits cannot be used to monitor the UART state.

Workaround

LINS bits should be used only in LIN mode.

ERR006239: eTimer: When counter is not enabled, capture flags can be set but capture register values are not updated.

Description

If counter is not enabled (ETIMER_CHn_CTRL1[CNTMODE]==000) and a capture event happens (as defined by ETIMER_CHn_CCCTRL[CPTxMODE]), capture flags (ETIMER_CHn_STS[ICF1 or ICF2]) are set. But at the same time the capture registers (ETIMER_CHn_CAPTx) are not updated.

Workaround

There is no workaround for this. The user's software should make sure not to initiate anything using the ETIMER_CHn_STS[ICF1] or ETIMER_CHn_STS[ICF2] flags if the eTimer module is disabled.

ERR006583: eTimer: Incorrect updating of the Hold Register

Description

The eTimer's Hold Registers (ETIMER_CHn_HOLD) are incorrectly updated when a Compare and Capture Control Register (ETIMER_CHn_CCCTRL) is read.

Workaround

The eTimer's Hold Registers are supposed to be updated with the Counter Registers (ETIMER_CHn_CNTR) value whenever a Counter Register is read. Recognize that the Hold Registers values will also be updated if a Compare and Capture Control Register is read.

ERR006620: FLASH: ECC error reporting is disabled for Address Pipelining Control (APC) field greater than Read Wait-State Control (RWSC) field.

Description

The reference manual states the following at the Platform flash memory controller Access pipelining functional description.

“The platform flash memory controller does not support access pipelining since this capability is not supported by the flash memory array. As a result, the APC (Address Pipelining Control) field should typically be the same value as the RWSC (Read Wait-State Control) field for best performance, that is, BKn_APC = BKn_RWSC. It cannot be less than the RWSC.”

The reference manual advises that the user must not configure APC to be less than RWSC and typically APC should equal RWSC. However the documentation does not prohibit the configuration of APC greater than RWSC and for this configuration ECC error reporting will be disabled. Flash ECC error reporting will only be enabled for APC = RWSC.

For the case when flash ECC is disabled and data is read from a corrupt location the data will be transferred via the system bus however a bus error will not be asserted and neither a core exception nor an ECSM interrupt will be triggered. For the case of a single-bit ECC error the data will be corrected but for a double-bit error the data will be corrupt.

Notes

1. Both CFlash & DFlash are affected by this issue.

2. For single-bit and double-bit Flash errors neither a core exception nor an ECSM interrupt will be triggered unless APC=RWSC.

3. The Flash Array Integrity Check feature is not affected by this issue and will successfully detect an ECC error for all configurations of APC >= RWSC.

4. For the APC > RWSC configuration other than flash ECC error reporting there will be no other unpredictable behaviour from the flash.

5. The write wait-state control setting at PFCRx[BKn_WWSC] has no affect on the flash. It is recommend to set WWSC = RWSC = APC.

Workaround

PFCRx[BKy_APC] must equal PFCRx PFCRx[BKy_RWSC]. See datasheet for correct setting of RWSC.

ERR006726: NPC: MCKO clock may be gated one clock period early when MCKO frequency is programmed as SYS_CLK/8.and gating is enabled

Description

The Nexus auxiliary message clock (MCKO) may be gated one clock period early when the MCKO frequency is programmed as SYS_CLK/8 in the Nexus Port Controller Port Configuration Register (NPC_PCR[MCKO_DIV]=111) and the MCKO gating function is enabled (NPC_PCR[MCKO_GT]=1). In this case, the last MCKO received by the tool prior to the gating will correspond to the END_MESSAGE state. The tool will not receive an MCKO to indicate the transition to the IDLE state, even though the NPC will transition to the IDLE state internally. Upon re-enabling of MCKO, the first MCKO edge will drive the Message Start/End Output (MSEO=11) and move the tool's state to IDLE.

Workaround

Expect to receive the MCKO edge corresponding to the IDLE state upon re-enabling of MCKO after MCKO has been gated.

ERR006802: eTimer: Extra input capture events can set unwanted DMA requests

Description

When using the DMA to read the eTimer channel capture registers (ETIMER_CHn_CAPTn) and the DMA has completed its programmed number of transfers an extra input capture event will set the eTimers input capture flag bit in the status register (ETIMER_CHn_STS[ICFn]) and also set the internal DMA request signal. While the input capture flag status bits (ICFn) can be cleared by writing a 1 to their bit positions the DMA request can only be cleared by the DMA done signal. This means that when a new DMA transfer is programmed the eTimer will request a DMA read with possibly unwanted data.

This behavior occurs once the DMA requests are disabled on the side of eDMA (DMA_ERQ[ERQn] = 0), but are still enabled in eTimer (ETIMER_CHn_INTDMA[ICFnDE] = 1), and the active edge is detected

Workaround

In cases where extra eTimer input capture events might occur the following procedure can be used to prevent unwanted DMA read requests:

1. Upon completion of the DMA transfer, disable the DMA requests by clearing the ETIMER_CHn_INTDMA[ICFnDE] bits.

2. If ETIMER_CHn_STS[ICFn] bits are clear then there are no extra input capture events and the eTimer is ready for further operation.

3. If the ICFn bits are set then read the ETIMER_CHn_CAPTn registers until the ETIMER_CHn_CTRL3[CnFCNT] fields are both 0 indicating the the capture FIFO's are empty. Then write a 1 to the ICFn bits to clear them. Next, create a dummy DMA read transfer to read the CAPTn registers. The DMA done signal will clear any pending DMA request.

ERR007322: FlexCAN: Bus Off Interrupt bit is erroneously asserted when soft reset is performed while FlexCAN is in Bus Off state

Description

Under normal operation, when FlexCAN enters in Bus Off state, a Bus Off Interrupt is issued to the CPU if the Bus Off Mask bit (CTRL[BOFF_MSK]) in the Control Register is set. In consequence, the CPU services the interrupt and clears the ESR[BOFF_INT] flag in the Error and Status Register to turn off the Bus Off Interrupt.

In continuation, if the CPU performs a soft reset after servicing the bus off interrupt request, by either requesting a global soft reset or by asserting the MCR[SOFT_RST] bit in the Module Configuration Register, once MCR[SOFT_RST] bit transitions from 1 to 0 to acknowledge the soft reset completion, the ESR[BOFF_INT] flag (and therefore the Bus Off Interrupt) is re-asserted.

The defect under consideration is the erroneous value of Bus Off flag after soft reset under the scenario described in the previous paragraph.

The Fault Confinement State (ESR[FLT_CONF] bit field in the Error and Status Register) changes from 0b11 to 0b00 by the soft reset, but gets back to 0b11 again for a short period, resuming after certain time to the expected Error Active state (0b00). However, this late correct state does not reflect the correct ESR[BOFF_INT] flag which stays in a wrong value and in consequence may trigger a new interrupt service.

Workaround

To prevent the occurrence of the erroneous Bus Off flag (and eventual Bus Off Interrupt) the following soft reset procedure must be used:

1. Clear CTRL[BOFF_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).
2. Set MCR[SOFT_RST] bit in the Module Configuration Register.
3. Poll MCR[SOFT_RST] bit in the Module Configuration Register until this bit is cleared.
4. Wait for 4 peripheral clocks.
5. Poll ESR[FLTCONF] bit in the Error and Status Register until this field is equal to 0b00.
6. Write "1" to clear the ESR[BOFF_INT] bit in the Error and Status Register.
7. Set CTRL[BOFF_MSK] bit in the Control Register (optional step in case the Bus Off Interrupt is enabled).

ERR007394: MC_ME: Incorrect mode may be entered on low-power mode exit.

Description

For the case when the Mode Entry (MC_ME) module is transitioning from a run mode (RUN0/1/2/3) to a low power mode (HALT/STOP/STANDBY*) if a wake-up or interrupt is detected one clock cycle after the second write to the Mode Control (ME_MCTL) register, the MC_ME will exit to the mode previous to the run mode that initiated the low power mode transition.

Example correct operation DRUN->RUN1-> RUN3->STOP->RUN3

Example failing operation DRUN->RUN1-> RUN3->STOP->RUN1

Note STANDBY mode is not available on all MPC56xx microcontrollers

Workaround

To ensure the application software returns to the run mode (RUN0/1/2/3) prior to the low power mode (HALT/STOP/STANDBY*) it is required that the RUNx mode prior to the low power mode is entered twice.

The following example code shows RUN3 mode entry prior to a low power mode transition.

```
ME.MCTL.R = 0x70005AF0; /* Enter RUN3 Mode & Key */

ME.MCTL.R = 0x7000A50F; /* Enter RUN3 Mode & Inverted Key */

while (ME.GS.B.S_MTRANS) {} /* Wait for RUN3 mode transition to complete */

ME.MCTL.R = 0x70005AF0; /* Enter RUN3 Mode & Key */

ME.MCTL.R = 0x7000A50F; /* Enter RUN3 Mode & Inverted Key */

while (ME.GS.B.S_MTRANS) {} /* Wait for RUN3 mode transition to complete */

/* Now that run mode has been entered twice can enter low power mode */

/* (HALT/STOP/STANDBY*) when desired. */
```

ERR007804: LINFlex: Consecutive headers received by LIN Slave triggers error interrupt

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.

But in LINFlex IP, 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. Also the LIN Slave remains waiting for the data response corresponding to the first header received.

Workaround

The following workaround should be applied:

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

2) Set Idle on Timeout in the LINTCSR[IOT] register to '1'.

3) Configure master to wait until the occurrence of the Output Compare flag in LIN Error Status Register (LINESR[OCF]) before sending the next header. This flag causes the LIN Slave to go to an IDLE state before the next header arrives, which will be accepted without any framing error.

ERR007877: FlexPWM: do not enable the fault filter

Description

Operation of the fault pin filter of the Flexible Pulse Width Modulation (FLEX_PWM) may be inconsistent if the Fault Filter is enabled, by setting the Filter Period greater than zero in the Fault Filter register (FFILT[FILT_PER] > 0). The fault filter flag may be set even though the pulse is shorter than the filter time.

Workaround

Do not enable the PWM fault pin filters. Disable the fault pin filters by setting the Fault Filter Period to 0 in the Fault Filter Register (FFILT[FILT_PER] = 0).

ERR008770: FlexRAY: Missing TX frames on Channel B when in dual channel mode and Channel A is disabled

Description

If the FlexRay module is configured in Dual Channel mode, by clearing the Single Channel Device Mode bit (SCM) of the Module Control register (FR_MCR[SCM]=0), and Channel A is disabled, by clearing the Channel A Enable bit (FR_MCR[CHA]=0) and Channel B is enabled, by setting the Channel B enable bit (FR_MCR[CHB]=1), there will be a missing transmit (TX) frame in adjacent minislots (even/odd combinations in Dynamic Segment) on Channel B for certain communication cycles. Which channel handles the Dynamic Segment or Static Segment TX message buffers (MBs) is controlled by the Channel Assignment bits (CHA, CHB) of the Message Buffer Cycle Counter Filter Register (FR_MBCCFRn). The internal Static Segment boundary indicator actually only uses the Channel A slot counter to identify the Static Segment boundary even if the module configures the Static Segment to Channel B (FR_MBCCFRn[CHA]=0 and FR_MBCCFRn[CHB]=1). This results in the Buffer Control Unit waiting for a corresponding data acknowledge signal for minislot:N in the Dynamic Segment and misses the required TX frame transmission within the immediate next minislot:N+1.

Workaround

1. Configure the FlexRay module in Single Channel mode (FR_MCR[SCM]=1) and enable Channel B (FR_MCR[CHB]=1) and disable Channel A (FR_MCR[CHA]=0). In this mode the internal Channel A behaves as FlexRay Channel B. Note that in this mode only the internal channel A and the FlexRay Port A is used. So externally you must connect to FlexRay Port A.

2. Enable both Channel A and Channel B when in Dual Channel mode (FR_MCR[CHA=1] and FR_MCR[CHB]=1). This will allow all configured TX frames to be transmitted correctly on Channel B.

ERR009976: DSPI: Incorrect data received by master with Modified transfer format enabled when using Continuous serial communication clock mode

Description

When the Deserial Serial Peripheral Interface (DSPI) module is configured as follows:

1. Master mode is enabled (Master/Slave Mode Select bit in Module Configuration Register is set (DSPI_MCR [MSTR] = 0b1))

2. Modified transfer format is enabled (Modified Transfer Format Enable bit in Module Configuration Register is set (DSPI_MCR [MTFE] = 0b1))

3. Continuous serial communication clock mode is enabled (Continuous SCK Enable bit in Module Configuration Register is set (DSPI_MCR [CONT_SCKE] = 0b1))

In this configuration if the frame size of the current frame is greater than the frame size of the next received frame, corrupt frames are received in two scenarios:

a) Continuous Peripheral Chip Select Enable bit in PUSH TX FIFO Register is set (DSPI_PUSHR [CONT] = 0b1)

b) DSPI_PUSHR [CONT] = 0b0 and lower significant bit of the frame is transferred first (LSB first bit in Clock and Transfer Attributes Register is set (DSPI_CTAR [LSBFE] =0b1))

Workaround

To receive correct frames:

a) When DSPI_PUSHR [CONT] = 0b1, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).

b) When DSPI_PUSHR [CONT] = 0b0, configure DSPI_CTAR [LSBFE] = 0b0. Alternatively, configure the frame size of the current frame less than or equal to the frame size of the next frame (for all frames).

Make sure that for all received frames, the bits are read equal to their respective frame sizes and any extra bits during POP operation are masked.

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.

ERR051059: DSPI: Inconsistent loading of shift register data into the receive FIFO following an overflow event

Description

In the Deserial Serial Peripheral Interface (DSPI) 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 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.

ERR051065: SARADC: Interrupted Normal conversion chain may not be restored properly

Description

When Normal conversion chain is interrupted by the injected conversion, it is possible that the aborted and remaining conversions of the chain does not get restored in expected sequence or skipped during restoration of the chain.

Vulnerable configurations are:

1) Software Injected chain over a normal chain

2) Cross Trigging Unit (CTU) trigger over a normal chain

3) CTU trigger over an Software injected chain over a normal chain

The following issue can occur:

a. The running channel/sample of normal chain is lost when the injection trigger comes in the conversion phase of running channel

b. Chain restored at an incorrect channel when injection trigger comes during end of the sample phase of a running normal conversion

Moreover to the issues a) and b) above the following additional behavior can occur:

c. Two End of Chain (ECH) interrupts are generated when the injection trigger arrives during conversion phase of last channel of normal chain.

d. If the trigger arrives during the sampling phase of the last channel in the normal chain, an ECH is triggered immediately and another ECH interrupt occurs when the channel is restored.

e. In scan mode, the second ECH is not generated if the injection trigger arrives during the conversion phase.

f. In one-shot mode, an injection trigger in the conversion phase of the last channel of normal chain restarts the whole conversion chain and the next ECH occurs at completion of that chain.

Workaround

The application should check for valid data using the Channel Data Register (CDR) to ensure all expected channels have converted. This can be tested by checking the VALID bit in the CDR registers. Any zero value for the CDR[i].VALID, where i represents all channels in the chain, indicates that a channel has been missed and conversion should be requested again or will be updated in next chain if not interrupted.

Spurious ECH interrupts can be detected by checking the NSTART flags in the SARADC_MSR (Module Status Registers) – if the flag remains set during an ECH interrupt then another interrupt will follow after the restored channel or chain has been sampled and converted.

The spurious ECH workaround above applies to single-shot conversions. In single-shot mode, NSTART changes from 1 to 0. Therefore, the user can rely on checking the NSTART bit to confirm if a spurious ECH has occurred. However, for scan mode, the NSTART bit will remain set during normal operation, so it cannot be relied upon to check for the spurious ECH issue. Consequently, if the CTU is being used in trigger mode, the conversions must be single-shot and not scan mode.

ERR051110: FlexPWM: Half cycle automatic fault clearing does not work in PWM submodule 0

Description

Half cycle automatic fault clearing does not work when the following configuration is used:

a) the EXT_SYNC signal is selected to cause initialization by setting the Submodule 0 Control 2 Register FlexPWM_SUB0_CTRL2[INIT_SEL] = 11 and

b) a specific FAULTx input is associated with the submodule 0 outputs using the Submodule 0 Fault Disable Mapping Register (FlexPWM_SUB0_DISMAP) and

c) the respective bit for that FAULTx is 1 in the FAUTO bitfield of the Fault Control Register FlexPWM_FCTRL,

then the PWM outputs of submodule 0 will only be re-enabled at the cycle boundary (full cycle) and will not be re-enabled at the cycle midpoint (half cycle).

Workaround

When the EXT_SYNC signal is used to cause initialization in submodule 0 and the submodule 0 PWM outputs are disabled by a specific FAULTx input, only full cycle automatic fault clearing is working for the specific FAULTx input.

ERR051698: CTU : Double buffer reload mechanism is blocked when master reload pulse is not generated by Software

Description

CTU operates on two clock domains. The Bus Interface Clock (BIC) domain, used for SW communication and the Module Clock (MC) domain, used for its own functionality. The FGRE bit of CR register is set with first write into double buffered registers in BIC domain and the synchronization logic propagates this set bit into the MC domain. FGRE bit is cleared in both domains when the MR occurs and FGRE bit is equal to GRE bit in the MC domain. Double buffered registers are reloaded only when MR occurs and FGRE and GRE bit are set in the MC domain. But when the MR occurs at the same time as FGRE bit set, generated by first double buffered register write, is propagated into the MC domain the FGRE bits are cleared in both.

Two possible cases can happen:

1) In case of updating one double buffered register the reload doesn’t occur and the CR register is kept in 0x2 value which blocks further double buffered register update.

2) In case of updating more than one double buffered register the reload doesn’t occur and the CR register is kept in 0xA when the delay between first and second double buffered register write is less than the synchronization propagation time which blocks further register update.

Workaround

1) Generate the Master reload pulse by SW (by setting the bit CR[MRS_SG] after setting GRE at the end of update sequence)

2) Master reload pulse is not generated by SW and one double buffered register to be updated:

The register needs to be written twice with the delay between the writes longer than synchronization propagation time.

3) Master reload pulse is not generated by SW and more than one double buffered register to be updated:

The delay between write to first and second double buffered register needs to be longer than synchronization propagation time.

Synchronization propagation time = (6*BIC) cycles + (5*MC) cycles, where BIC is Bus Interface Clock period and MC is Module Clock period.

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 7/2023
Document identifier: MPC5604P_1M36W