M480 Series Errata Sheet

Errata Sheet for 32-bit NuMicro® Family

Document Information

Abstract	This errata sheet describes the functional problem known at the release date of this document.
Apply to	M480 Series.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

Table of Contents

1 Overview

Functional Problem	Description
USCI-I²C slave receives invalid data when10-bit addressing mode is enabled	USCI-I2C will receive invalid data in 10-bit addressing mode.
USCI-I²C master sending data failed when 10-bit addressing mode is enabled	Master cannot send data after sending address in 10-bit addressing mode.
Power-down OPA failed	Power-down OPA failed if the duration between OPA disabled and the OPA module clock disabled is shorter than 1 ms.
HSUSBD EPJ, EPK , and EPL data register word write failed	CPU cannot support word write for HSUSBD EPI, EPK, and EPL data registers.
EMAC receives error after system wake-up	The data in the first packet received after Wake-on-Lan (WoL) wake-up is incorrect.
Inrush current when MCU wakes up from DPD mode	MCU wake-up from Deep Power-down (DPD) mode causes inrush current.
Leakage Current in GPIO Quasi-bidirectional mode	Each GPIO consumes 65uA leakage current when the pin state is set to LOW
DPD wake-up failed	DPD wake-up failed if DPD mode wake-up interval from sleep is too short.
PDMA channel reset/enable causes transfer error	PDMA channel reset or enable will cause other channels to shift its data.
PDMA channel configuration causes scatter-	Configuring control registers of other channels while PDMA is loading next table can cause PDMA failed to load the next table.

gather transfer error	
Timer cannot count (M48xGC/M48xG8 Only)	Timer will not count if peripheral clock is faster than PCLK.
Bits in SYS_IPRST2 read as 0 (M48xGC/M48xG8 Only)	QSPI1RST(SYS_IPRST2[4]), CAN2RST(SYS_IPRST2[28]), and EADC1RST (SYS_IPRST2[31]) are always read as 0.
ACMP window latch mode misoperation (M48xGC/M48xG8 Only)	ACMP triggers an interrupt while enabling window latch mode.
ACMP window latch mode misses the first state change (M48xGC/M48xG8 Only)	ACMP misses the first state change in window latch mode.
ACMP clears Interrupt flag sequence (M48xGC/M48xG8 Only)	CPU cannot clear ACMPIFx interrupt flag before WKIFx interrupt flag.
EPWM trigger EADC failed (M48xGC/M48xG8 Only)	EPWM trigger EADC failed in some cases.
HIRC and HIRC48 stop while LVR is disabled (M48xGC/M48xG8 Only)	On chip HIRC and HIRC48M clock source stop while LVR is disabled.
V_{BAT} current leakage (M48xGC/M48xG8 Only)	The V_{BAT} power pin has current leakage if the V_{DD} voltage is in the range from $V_{BAT}/2$ to V_{BAT} – 0.3V.
WDT reset under Power-down mode	System cannot be executed normally if it is reset by WDT from (NPD/FWPD/LLPD) mode.

(Except M48xGC/M48xG8)	
V_{DDIO} current leakage (M48xGC/M48xG8 Only)	The V_{DDIO} domain I/Os (PA.0~PA.5) generate leakage current if V_{DD} - V_{DDIO} > 0.5 V and any of VDDIO domain I/O digital path is enabled and output high.
UART control bit WKIEN is useless	Clearing WKIEN (UART_INTEN[6]) control bit 0 cannot prevent wake-up interrupt to be triggered.
Entering LLPD and SPD mode may fail (Except M48xGC/M48xG8)	System enter LLPD mode failure in some cases.
Memory being flushed by EMAC DMA during packet flooding	System memory is flushed by EMAC DMA while receiving high throughput UDP packets.

2 Functional Problems

2.1 USCI-I²C slave receives invalid data when 10-bit addressing mode is enabled

Description:

USCI-I²C will receive invalid data in 10-bit addressing mode.

Problem:

When a master transmits a high-byte match address and low-byte mismatch address to I²C bus, the slave of USCI-I²C puts the low-byte address into RXDAT (USCI_RXDAT[16:0]) of USCI-I²C.

Workaround:

Perform a dummy read from RXDAT (USCI_RXDAT[16:0]) when STORIF (USCI_PROSTS[9]) occurs. Below is an example that performs a dummy read after STORIF interrupt occurs.

```
void UI2C_SlaveTRx(uint32_t u32Status)
{
    …
    } else if((u32Status & UI2C_PROTSTS_STORIF_Msk) == UI2C_PROTSTS_STORIF_Msk) {
        /* Clear STOP INT Flag */
        UI2C_CLR_PROT_INT_FLAG(UI2C0, UI2C_PROTSTS_STORIF_Msk);

        /* Dummy Read */
        u32Dummy = UI2C_GET_DATA(UI2C0);

        /* Event process */
        g_u8SlvDataLen = 0;
        s_Event = SLAVE_H_WR_ADDRESS_ACK;

        UI2C_SET_CONTROL_REG(UI2C0, (UI2C_CTL_PTRG | UI2C_CTL_AA));
    }
}
```

2.2 USCI-I²C master sending data failed when 10-bit addressing mode is enabled

Description:

Master cannot send data after sending address.

Problem:

When ADDR10EN is enabled, USCI_I²C will check if AA (USCI_PROTCTL[1]) bit is set or not. If AA (USCI_PROTCTL[1]) bit is cleared, USCI_I²C will become 'read' transfer in this transmission. On the contrast, if AA (USCI_PROTCTL[1]) bit is set, USCI_I²C becomes 'write' transfer in this transmission after sending slave address

Workaround:

When master sends START signal, it also enables AA (USCI_PROTCTL[1]) bit. Below is an example that sets AA bit along with the START signal.

```
void UI2C_SlaveTRx(uint32_t u32Status)
{
    …
        if(m_Event == MASTER_SEND_ADDRESS) {
            /* SLA+W has been transmitted and NACK has been received */
            m_Event = MASTER_SEND_START;
            /* Send START signal */
            UI2C_SET_CONTROL_REG(UI2C0,
                                 (UI2C_CTL_PTRG | UI2C_CTL_STA | UI2C_CTL_AA));
        }
    …
}
```

2.3 Power-down OPA failed

Description:

Power-down OPA failed if the duration between OPA disabled and the OPA module clock disabled is shorter than 1 ms, where OPA disabled indicates that OPENx(OPA_CTL[2:0]) bit is cleared; the OPA module clock disabled indicates that OPACKEN(CLK_APBCLK1[30]) bit is cleared.

Problem:

An OPA finite state machine transition is controlled by a 1 kHz clock that comes from HIRC 12MHz for sampling. Thus, it takes 1ms for the finite state machine to transit to idle state after OPA is disabled. OPA analog macro cannot be disabled and will keep consuming power if the OPA module clock is disabled before the finite state machine transits to idle state.

Workaround:

Add an OPA software module reset between OPA disabled and OPA module clock disabled. The reset can force the OPA finite state machine to transit to idle state and to disable the OPA analog macro without sampling the 1 kHz clock. Below is an example using the SYS_ResetModule() function to reset the OPA module after OPA is disabled and before OPA module clock is disabed.

```
void OPA_POWER_DOWN(OPA_T *opa, uint32_t u32OpaNum)
{
  (opa)->CTL &= ~(1UL<<(OPA_CTL_OPEN0_Pos+(u32OpaNum)));
  SYS_ResetModule(OPA_RST);
  CLK_DisableModuleClock(OPA_MODULE);
}
```

2.4 HSUSBD EPJ, EPK, and EPL data registers word write failed

Description:

High-speed USBD (HSUSBD) endpoint data registers support both byte access and word access. However, endpoint J (EPJ) data register, endpoint K (EPK) data register, and endpoint L (EPL) data register cannot support CPU word write.

Problem:

A CPU word write to EPK or EPL data registers only send two least significant bytes into their memory buffer. A CPU word write to EPJ will corrupt the data in EPK and EPL memory buffer.

Workaround:

Several methods listed below can be used to avoid this problem.

1. Use CPU byte write instead of CPU word write while accessing these endpoints' data register.
2. Use DMA word write instead of CPU word write while accessing these endpoints' data register.
3. Use these endpoints for IN transfer only, so there is no need to write endpoints' data register.

2.5 EMAC receives error after system wake-up

Description:

The data in the first packet received after Wake-on-Lan (WoL) wake-up is incorrect.

Problem:

An EMAC internal FIFO pointer is not reset after the CPU is woken up by WoL packet from Power-down mode. Therefore, the first packet received is corrupted.

Workaround:

Driver should drop the first received packet after system wake-up by WoL packet and use upper layer protocol to recover from packet loss, or let other device send multiple WoL packets to wake up the M480 series from Power-down mode.

2.6 Inrush current when MCU wakes up from DPD mode

Description:

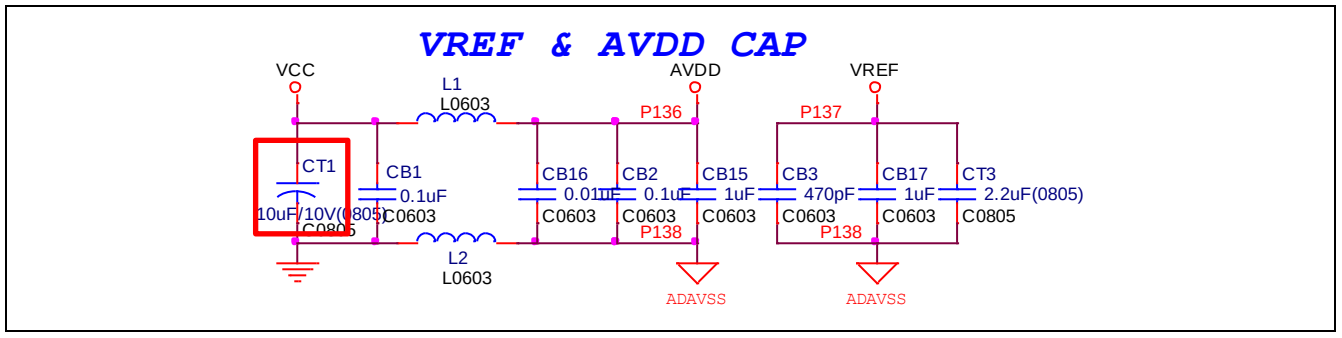
MCU wake-up from Deep Power-down mode (DPD) causes inrush current.

Problem:

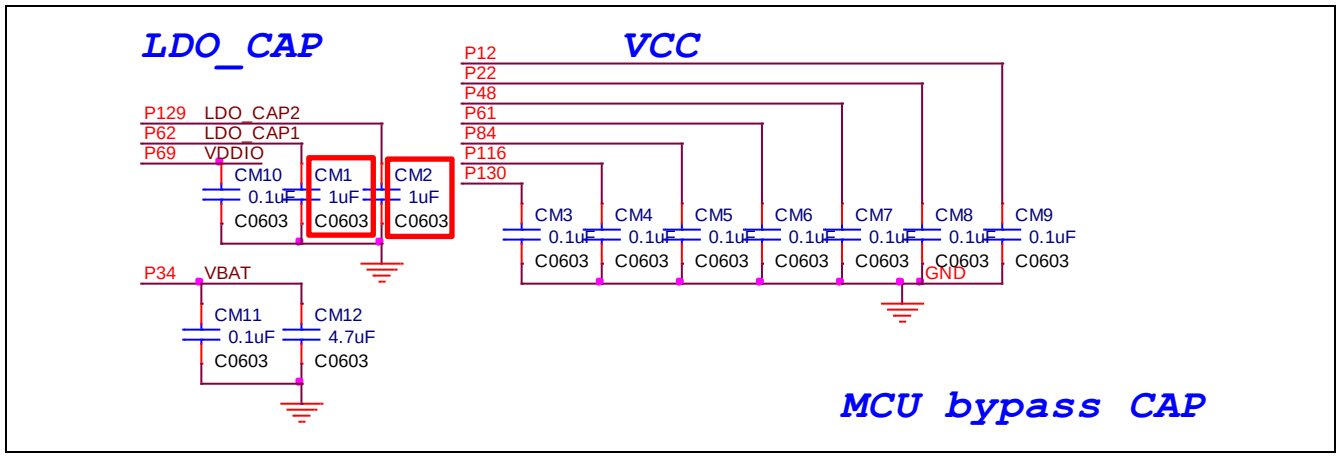
The M480 internal capacitor will be fully discharged when MCU enters DPD mode. When MCU is woken up from DPD mode, it draws current from power supply to recharge the internal capacitors and cause inrush current.

Workaround:

This issue can be fixed by a correct capacitor. Using correct capacitor can decrease inrush current impact. Here is the block diagram of V_{REF} & AV_{DD} capacitors. Change the CT1 capacitor from 10uF to 22uF if the system needs smaller inrush current.



Below is the block diagram of LDO_CAP. There are two capacitors with a value added up to 2.2uF. For the packages with two LDO_CAP pins, both CM1 and CM2 should be 1uF. As to the packages with one LDO_CAP pin, CM1 or CM2 must be 2.2uF.



Please refer to the table below for experimental results on the package with one LDO_CAP pin

(CM2) only.

CT1 (uF)	CM2 (uF)	Inrush current after wake-up from DPD mod (mA)
22	2.2	390
10	2.2	450

Note: It is suggested using the default capacitor value if the system does not need to enter DPD mode.

2.7 Leakage current in GPIO Quasi-bidirectional mode

Description:

In Quasi-bidirectional mode, the I/O pin supports digital output and input function at the same time but the source current is only up to hundreds uA. If the corresponding data output value DOUT (Px_DOUT[n]) bit is 0, the pin drives a low output. If the corresponding data output value DOUT (Px_DOUT[n]) bit is 1, and pin state is high, no action is taken. Otherwise, the pin will drive strong high for 2 clock cycles and then disable the strong output drive and use an internal pull-up resistor to keep the pin status.

Problem:

When a GPIO pin is configured as Quasi-bidirectional mode, there will be a 65uA leakage current consumed by the pull-up resister as long as the pin state is low.

Workaround:

Disable Quasi-bidirectional mode to reduce leakage current when the I/O pin state is low if the state is drive low externally, and the GPIO should be configured as Input mode or Open-drain mode. If the I/O pin drives a low output, the GPIO should be configured as Output mode.

2.8 DPD wake-up failed

Description:

The DPD (Deep Power-Down) mode can minimize memory power consumption by shutting down the internal power supply generator and suspending SRAM refresh operations. The M480 cannot wake up from DPD mode if the power-down and wake-up interval is too short. For example, if RTC tick is less than 1/8 sec or Timer interval is less than 1/128 sec, the DPD wake-up will be failed.

Problem:

Upon DPD mode wake-up, a circuit will trigger POR (Power on Reset) automatically to make the system resume normal operation. However, one-time POR function can block DPD to wake up POR if the power-down and wake-up interval is too short.

Workaround:

Application can disable one-time POR by writing 0 to SYS_PORCTL register before entering DPD mode, thus the one-time POR function cannot prevent the circuit to trigger POR. After wake-up, the application can enable one-time POR by writing 0x5AA5 to SYS_PORCTL

Disable one-time POR:

```
SYS->PORCTL = 0;
```

Enable one-time POR:

```
SYS->PORCTL = 0x00005AA5;
```

Please note that SYS_PORCTL is a protected register and can only be accessed when the register protection is disabled. The sequence to disable register protection is writing the data “59h”, “16h” “88h” to the register SYS_REGLCTL continuously.

2.9 PDMA channel reset/enable causes transfer error

Description:

The PDMA controller supports multiple channels. Sometimes the application needs to reset a PDMA channel due to transfer error caused by buffer overflow or underflow, or needs to enable a PDMA channel while other channels are active.

Transfer errors may occur on the active channels while other channel is reset or enabled.

Problem:

The reset signal of other channels will affect the channels that are loading a descriptor first time and cause the affected channels to shift its data. For example, on a peripheral to memory channel, the correct data “0x30, 0x31, 0x32, 0x33, 0x34, 0x35…” becomes “0x31, 0x32, 0x33, 0x34, 0x35…” where 0x30 is overwritten by 0x31, 0x31 overwritten by 0x32… and so on. On a memory to peripheral channel, the correct data “0x30, 0x31, 0x32, 0x33, 0x34, 0x35…” becomes “0x30, 0x30, 0x31, 0x32, 0x33…” where the first data is sent twice.

Workaround:

Before a PDMA channel is reset or enabled, pause other channels in operation and resume them after reset/enable is complete. Below is an example demonstrating this procedure using PDMA0 channel 0 and 1.

```
/* Pause PDMA0 channel 0 */
PDMA_PAUSE (PDMA0, 0);

/* Wait for PDMA0 channel 0 to finish current transfer */
While (PDMA0->TACTSTS & BIT0) {}

/* Reset PDMA0 channel 1 */
PDMA0->CHRST = BIT1;

/* Enable and re-trigger PDMA0 channel 0 */
PDMA0->CHCTL |= BIT0;
PDMA_Trigger (PDMA0, 0);
```

2.10 PDMA channel configuration causes scatter-gather transfer error

Description:

If there are multiple PDMA channels enabled to transfer data in sequence for application, both Basic and Scatter Gather modes are enabled. For example, when the PDMA channel 0 with Scatter Gather mode is enabled by setting OPMODE (PDMA_DSCT0_CTL[1:0]) = 2 has two linked-list tables, the PDMA controller will load the second table into PDMA_DSCT0_CTL register to execute after the first table is executed. Then, for the PDMA Channel 1 Basic mode is enabled by setting OPMODE (PDMA_DSCT1_CTL[1:0]) = 1.

Problem:

If the control register (PDMA_DSCT1_CTL), the source address register (PDMA_DSCT1_SA), the destination address register (PDMA_DSCT1_DA) or the next scatter-gather table offset address (PDMA_DSCT1_NEXT) of channel 1 is configured when the PDMA controller loads the second table of channel 0 into PDMA_DSCT0_CTL register, the PDMA controller may fail to load the second table into the PDMA_DSCT0_CTL register.

Workaround:

The PDMA channel 0 with Scatter Gather mode enabled needs to pause operation when the PDMA channel 1 configures its control register. Then, the paused channel needs to be re-triggered for continuous operation after the configuring the control register of channel 1 is complete.

```
/* Pause PDMA channel 0 */
PDMA_PAUSE (PDMA0, 0);

/* Wait for PDMA0 channel 0 to finish current transfer */
While (PDMA0->TACTSTS & BIT0) {}

/* Configure control register of PDMA channel 1 */
…

/* Enable and re-trigger PDMA0 channel 0 */
PDMA0->CHCTL |= BIT0;
PDMA_Trigger (PDMA0, 0);
```

2.11 Timer cannot count

Description:

The timer will not count if peripheral clock is slower than PCLK.

Problem:

The timer will not properly sample prescaler counter while peripheral clock is faster than PCLK.

Workaround:

There is no workaround for this issue. Always make sure PCLK0 is not slower than TMR0_CLK/TMR1_CLK and PCLK1 is not slower than TMR2_CLK/TMR3_CLK.

2.12 Bits in SYS_IPRST2 read as 0

Description:

QSPI1RST(SYS_IPRST2[4]), CAN2RST(SYS_IPRST2[28]), and EADC1RST (SYS_IPRST2[31]) are always read as 0 even they have just been set to 1.

Problem:

The register read path of these bits are not connected correctly. Thus, these bits are always read as 0.

Workaround:

There is no point to read back these bits during normal operation. Reset QSPI1, CAN2, EADC1 as follows.

```
/* Reset QSPI1 */
SYS->IPRET2 = SYS_IPRST2_QSPI1RST_Msk;
SYS->IPRET2 = 0;

/* Reset CAN2 */
SYS->IPRET2 = SYS_IPRST2_CAN21RST_Msk;
SYS->IPRET2 = 0;

/* Reset EADC1 */
SYS->IPRET2 = SYS_IPRST2_EADC1RST_Msk;
SYS->IPRET2 = 0;
```

2.13 ACMP window latch mode misoperation

Description:

ACMP always triggers an interrupt while enabling window latch mode if ACMPOx (ACMP_STATUS[4], ACMP_STATUS[5]) equals to 1 regardless of the ACMPx_WLAT pin state.

Problem:

The window latch mode output cannot latch a previous ACMP output status since the current ACMP output status passes window latch mode while enabling window latch mode.

Workaround:

Clear ACMPEN (ACMP_CTLx[0]) before setting WLATEN (ACMP_CTLx[17]), and then set ACMPEN (ACMP_CTL[0]).

```
/* Clear ACMPEN bit */
ACMP->CLT[0] &= ~ACMP_CTL_ACMPEN_Msk;

/* Set WLATEN bit */
ACMP->CLT[0] |= ACMP_CTL_WLATEN_Msk;

/* Set ACMPEN bit */
ACMP->CLT[0] |= ACMP_CTL_ACMPEN_Msk;
```

2.14 ACMP window latch mode misses the first state change

Description:

ACMP will miss its first falling event if ACMPOx (ACMP_STATUS[4], ACMP_STATUS[5]) equals to 1 while enabling window latch mode.

Problem:

The switch in the window latch mode cannot switch correctly causes the window latch mode output missing the first falling event of ACMPOx.

Workaround:

There is no workaround for this issue. Try to enable window latch mode while ACMPOx is 0.

2.15 ACMP clears Interrupt flag sequence

Description:

CPU cannot clear ACMPIFx (ACMP_STATUS[0], ACMP_STATUS[1]) when WKIFx (ACMP_STATUS[8], ACMP_STATUS[9]) is set.

Problem:

The clear ACMPIFx bit criteria includes WKIFx is 0. Thus, write 1 cannot clear ACMPIFx if WKIFx is 1.

Workaround:

Clear WKIFx and then clear ACMPIFx, or clear them in single instruction.

```
/* Clear both WKIF0 and ACMPIF0 bit */
ACMP->STATUS = ACMP_STATUS_WKIF0_Msk | ACMP_STATUS_ACMPIF0_Msk;

/* Clear WKIF1 before ACMPIF1 bit */
ACMP->STATUS = ACMP_STATUS_WKIF1_Msk;
ACMP->STATUS = ACMP_STATUS_ACMPIF1_Msk;
```

2.16 EPWM trigger EADC failed

Description:

EPWM trigger EADC failed in following cases.

- Enable EPWM channel 0 or 1 trigger EADC, and trigger source select EPWM channel 2 or 4 (up-count or down count) free trigger compare point
- Enable EPWM channel 2 or 3 trigger EADC and trigger source select EPWM channel 0 or 4 (up-count or down count) free trigger compare point
- Enable EPWM channel 4 or 5 trigger EADC and trigger source select EPWM channel 0 or 2 (up-count or down count) free trigger compare point

Problem:

The EPWM channel does not send request to trigger EADC while the channel is not enabled and its complementary pair channel is also not enabled.

Workaround:

- Enable EPWM channel 0 or 1 trigger EADC, and trigger source select EPWM channel 2 or 4 (up-count or down count) free trigger compare point, also enable CNTEN0(EPWM_CNTEN[0] or CNTEN1(EPWM_CNTEN[1]
- Enable EPWM channel 2 or 3 trigger EADC and trigger source select EPWM channel 0 or 4 (up-count or down count) free trigger compare point, also enable CNTEN2(EPWM_CNTEN[2] or CNTEN3(EPWM_CNTEN[3]
- Enable EPWM channel 4 or 5 trigger EADC and trigger source select EPWM channel 0 or 2 (up-count or down count) free trigger compare point, also enable CNTEN4(EPWM_CNTEN[4] or CNTEN5(EPWM_CNTEN[5]

2.17 HIRC and HIRC48 stop while LVR is disabled

Description:

On chip HIRC and HIRC48M clock source stop while LVREN (SYS_BODCTL[7]) is set to 0.

Problem:

Any peripheral that uses HIRC or HIRC48M as clock source cannot work while LVR is disabled. In this condition, CPU will also stop if system clock source is HIRC.

Workaround:

If there is no other concern, user should keep LVR enabled to avoid this condition.

2.18 V_{BAT} current leakage

Description:

The V_{BAT} power pin has current leakage if the V_{DD} voltage is in the range from $V_{BAT}/2$ to V_{BAT} – 0.3V.

Problem:

The maximum V_{BAT} current leakage is 50uA. If the V_{BAT} power source is a battery, the battery will be exhausted in a short time.

Workaround:

For battery application, user should keep V_{DD} not in the range from $V_{BAT}/2$ to V_{BAT} – 0.3V to avoid this current leakage condition.

2.19 WDT reset under Power-down mode

Description:

System cannot be executed normally if it is reset by WDT from (NPD/FWPD/LLPD) mode.

Problem:

WIC (Wakeup Interrupt Controller) reset signal is not synchronous with CPU reset signal, so the interrupt status in WIC cannot be cleared, causing M480 to be trapped in interrupt handler after system reset.

Workaround:

The workaround is to disable WDT reset function before system enters NPD/FWPD/LLPD mode. There are three scenarios depending on the CWDTEN[2:0] setting in user configuration.

CWDTEN[2:0]	Description
111	With this setting, WDT hardware is disabled after system boot up. Simply clear RSTEN (WDT_CLT[1]) bit before entering Power-down mode.
011	WDT clock is disabled while system enters power mode; no change is required.
Other value	When WDT hardware enable function is active, WDT clock is always on. With this setting, RSTEN cannot be cleared as well. Thus, before entering power mode, WDT peripheral clock needs to be switched to a clock source that is stopped while system is in Power-down mode. For example, switch WDT peripheral clock source to LIRC, and then clear LIRCEN (CLK_PWRCTL[3]) bit before entering Power-down mode. **Note:** WDT timeout event cannot be used as a wake-up source with this workaround.

2.20 Leakage current from V_{DDIO} domain I/O

Description:

The V_{DDIO} domain I/Os (PA.0~PA.5) generate leakage current when setting the conditions below:

1. V_{DD} - 0.5V > V_{DDIO}
2. Any of V_{DDIO} domain I/O digital path is enabled and output high.

Problem:

When the above condition is set, there is about hundreds of micro leakage current from V_{DD} to V_{DDIO}.

Workaround:

Do not set the following conditions:

1. V_{DD} - 0.5V > V_{DDIO}
2. Any of V_{DDIO} domain I/O digital path is enabled and output high.

2.21 UART control bit WKIEN is useless

Description:

Regardless the WKIEN (UART_INTEN[6]) bit status, the wake-up interrupt flag WKIF(UART_INTSTS[6]) always generates the wake-up interrupt WKINT (UART_INTSTS[14]).

Problem:

The WKIEN bit does not affect the wake-up interrupt behavior.

Workaround:

If wake-up system from UART status change is undesirable, clear the UART_WKCTL register to 0 before entering Power-down mode.

2.22 Entering LLPD and SPD mode may fail

Description:

If an interrupt is triggered within certain timing window while M480 enters LLPD/SPD mode, the M480 cannot enter LLPD/SPD state, nor can it return to normal run state.

Problem:

There is a handshake protocol between the PMU (Power Management Unit) and M480 core to make M480 enter LLPD/SPD mode. An interrupt triggered within 16 HCLK cycle after CPU executes WFI instruction will break the protocol and cause the PMU and M480 core stay in different state. Only a power cycle can recover the system form such state.

Workaround:

If an interrupt triggered by an internal event, such as Timer timeout interrupt, is not used as a wake-up source, the interrupt needs to be disabled before entering Power-down mode. Otherwise, avoid the interrupt being trigged within the 16 HCLK window mentioned above. For example, restart the timer counter and make timer interval longer than 16 HCLK clocks before executing the WFI instruction.

```
void tmr1_init(uint32_t u32freq)
{
    TIMER_Open(TIMER1, TIMER_PERIODIC_MODE, u32freq);
    TIMER_EnableInt(TIMER1);
    NVIC_EnableIRQ(TMR1_IRQn);
    TIMER_Start(TIMER1);
}

int main()
{
...

tmr1_init(512000);

...

/* Disable SysTick since it is not used as a wakeup source */
SysTick->CTRL = 0UL;
```

```
/* If TIMER1 is used as wakeup source, reset timer counter and change interval
    to be larger than 16 HCLK clocks before entering LLPD/SPD mode. */
tmr1_init(1250);

/* Put the system to sleep */
__WFI();

/* Wake up from LLPD/SPD mode. Restore timer setting. */
tmr1_init(512000);
```

Other interrupts are triggered by external events, for example, GPIO interrupt. Except noise filtering, the GPIO debounce function could be used to postpone interrupt event trigger time if the interrupt source is GPIO rising edge. The following sample code utilizes debounce combined with a GPIO feature that the input is tied to low while digital path is disabled as a workaround. Here the debounce time is set to 2 LIRC clocks which is longer than 16 HCLK clocks. A delay longer than debounce time with the digital path disabled is placed right before the execution of WFI instruction. Thus, a GPIO rising edge event occurs before GPIO_DISABLE_DIGITAL_PATH() will be dropped. And any rising edge event comes in after the GPIO_ENABLE_DIGITAL_PATH() can only trigger the interrupt 2 LIRC clocks later, which falls outside the 16 HCLK clocks window.

```
/* Set debounce time to 2 LIRC */
GPIO_SET_DEBOUNCE_TIME(GPIO_DBCTL_DBCLKSRC_LIRC, GPIO_DBCTL_DBCLKSEL_2);

/* Enable debounce function on input pin, PA.8, PA.3, and PB.0 in this sample */
GPIO_ENABLE_DEBOUNCE(PA, BIT8);
GPIO_ENABLE_DEBOUNCE(PA, BIT3);
GPIO_ENABLE_DEBOUNCE(PB, BIT0);

/* Disable digital path, so input to debounce circuit tied to low */
GPIO_DISABLE_DIGITAL_PATH(PA, BIT8);
GPIO_DISABLE_DIGITAL_PATH(PA, BIT3);
GPIO_DISABLE_DIGITAL_PATH(PB, BIT0);

/* Insert a delay >= 2 LIRC clocks here, assuming HCLK is 12 MHz in this sample */
for(i = 0; i < 1200; i++)
    __NOP();

/* Enable digital path. Debounce circuit can continue to work normally */
```

```
GPIO_ENABLE_DIGITAL_PATH(PA, BIT8);
GPIO_ENABLE_DIGITAL_PATH(PA, BIT3);
GPIO_ENABLE_DIGITAL_PATH(PB, BIT0);

/* Put the system to sleep */
__WFI();

/* Disable debounce function if it’s not enabled before power down procedure. */
GPIO_DISABLE_DEBOUNCE(PA, BIT8);
GPIO_DISABLE_DEBOUNCE(PA, BIT3);
GPIO_DISABLE_DEBOUNCE(PB, BIT0);
```

Although there exists workaround for GPIO interrupt, some of the interrupts triggered by external events do not have a workaround solution. For example, there is no workaround to prevent the TIMER event counting interrupt triggered within the 16 HCLK window. In this case, other Power-down mode such as NPD, FWPD, or DPD should be considered.

2.23 Memory being flushed by EMAC DMA during packet flooding

Description:

System memory is flushed by EMAC DMA while receiving high throughput UDP packets.

Problem:

There is a hardware defect in the M480 EMAC Rx DMA. When receiving high-throughput packets, the DMA may overwrite memory beyond the descriptor region, leading to system crashes.

Workaround:

Approaches to address this issue:

1. Allocate a Block of Memory for Descriptors:
   Initially, allocate a block of memory at the end of SRAM for descriptors. This will serve as a safeguard against potential system memory overwrite by DMA.
2. Embed an Identifier in the Final Unused Descriptor:
   Embed an identifier in the final unused descriptor to serve as a marker. This identifier will help track whether DMA has overwritten system memory.
3. Create a Monitoring Thread:
   Create a dedicated thread to monitor the identifier embedded in the final unused descriptor. This thread will periodically check whether the identifier has been overwritten.
4. Reset EMAC on Identifier Overwrite:
   If the monitoring thread detects that the identifier has been overwritten, trigger a reset of the EMAC to prevent further system crashes.

Revision History

Date	Revision	Description
2017.06.01	1.00	• Initially issued.
2018.03.16	2.00	• Updated for version C.
2018.08.30	2.01	• Added PDMA channel reset/enable limitation.
2019.05.30	2.10	• Added issues for M48xGC/M48xG8.
2021.06.07	2.11	• Added issue for VDDIO leakage. • Added WDT reset limitation under Power-down mode.
2021.11.29	2.12	• Added issue of UART WKIEN bit. • Added issue of LLPD and SPD mode.
2024.04.18	2.13	• Added issue of EMAC DMA.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.