Mask Set Errata for Mask 0N23N/1N23N/0P81C/1P81C

This report applies to mask 0N23N/1N23N/0P81C/1P81C for these products:

- S912ZVC64
- S912ZVCA64
- S912ZVC12
- S912ZVC19
- S912ZVCA19

Table 1. Errata and Information Summary

Erratum ID	Erratum Title
ERR008188	ADC: High current in Stop Mode

Table 2. Revision History

Revision	Changes
0	Initial revision
06 MAR 2015	No changes to errata with this revision
28 May 2021	The following erratum was revised. • ERR008188

ERR008188: ADC: High current in Stop Mode

Description: The ADC can take a higher current in Stop Mode (500µA increasing over time to about 1mA per ADC instance) if the ADC is enabled and conversions have been done.

There is a software workaround available.

Workaround: This software workaround has been validated.

The subroutine ADC0_stop_current_workaround takes about 105 bus clock cycles (ECLK) on S12ZVM128.

The subroutine must be executed for each ADC instance performing conversions before Stop Mode entry.

Before the workaround subroutine is executed, the ADC must be initialized (command list pointers and result list pointers have valid addresses).

Please note:

- All the conversion commands of List 0 must be valid (no illegal channel, no illegal SMP value)
- ADCFMT[SRES] value must not be illegal
- The command fetch must not cause an illegal access flag

```
void main(void) {

...

DisableInterrupts; // Shutdown sequence is not interruptible until STOP

...

ADC0_stop_current_workaround(); // Subroutine must be executed with interrupt protection
asm(andcc #0x6f); /* CCW settings: S = 0, I = 0 */ asm(stop); /* MCU enters Stop mode if S =
0 in CCW */ ... }

//Workaround to avoid ADC high current during STOP mode

//ADC will be disabled in this function

//Therefore following register are cleared:

//Address | Register | Description

//----------+----------+---------------------------

//0x02 | ADCSTS | CSL_SEL, RVL_SEL will be restored at the end of the function

//0x08 | ADCEIF | Error interrupt flags

//0x09 | ADCIF | Interrupt flags (SEQAD_IF, CONIF_OIF)

//0x0C-0x0D:| ADCCONIF | Conversion interrupt flags, EOL (end of list) interrupt flag

//0x0E-0x0F:| ADCIMDRI | Intermediate result information (must be stored by the customer
before,

// | | if this information is still needed after the function execution)

//0x10 | ADCEOLRI | EOL result information (must be stored by the customer before,

// | | if this information is still needed after the function execution)

//0x1C | ADCCIDX |

//0x20 | ADCRIDX |

void ADC0_stop_current_workaround (void) {

byte tmp_ADCxCTL_0 = ADC0CTL_0; // Save customer settings, these values will be restored
afterwards byte tmp_ADCxTIM = ADC0TIM; byte tmp_ADCxSTS = ADC0STS; ADC0CTL_0 =
0x00; // Disable ADC ADC0TIM = 0x00; // ADC is set to maximum frequency // Device
specification of allowed frequency is ignored, // the ADC conversion is stopped when reaching
error state // There is no conversion result generated. ADC0CTL_0 = 0x88; // ADC is enabled,
single access mode data bus, restart mode ADC0CTL_0 = 0x88; // Re-do in order to
guarantee ADC is ready for requests before Restart Event occurs ADC0FLWCTL = 0x20; //
```

ADC is restarted, RSTA bit is set: the first command of list 0 is // loaded from memory. The command type does not matter, the conversion // is immediately stopped while (ADC0FLWCTL_RSTA == 1) {} // Wait for restart completion (within a few clock cycles) ADC0FLWCTL = 0x40; // Start conversion (TRIG) ADC0FLWCTL = 0x40; // The second TRIG immediately generates a TRIG_EIF, ERROR state is entered while (ADC0EIF_TRIG_EIF == 0) {} // Wait for trigger error interrupt flag (within a few clock cycles) ADC0CTL_0_ADC_SR = 1; // Execute ADC soft-reset (SR), ADC enters IDLE state while (ADC0STS_READY == 0) {} // Wait for ADC soft-reset done (within a few clock cycles) ADC0CTL_0 = 0x00; // ADC is disabled ADC0TIM = tmp_ADCxTIM; //Restore previous customer settings ADC0STS = tmp_ADCxSTS; ADC0CTL_0 = tmp_ADCxCTL_0; // ADC0CTL_0 is the last one to be restored, in case ADC was enabled before... }

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

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

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

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