AVR1008: Writing EEPROM and Flash in XMEGA A3/D3/256A3B Revision B

8-bit AVR® Microcontrollers

Application Note

Features

- **EEPROM Programming**
- **FLASH Programming**

1 Introduction

This document describes a workaround for the errata “**Writing EEPROM or Flash while reading any of them will not work”** that is present on revision B of XMEGA™ A3 family and the revision B of the ATxmega256A3B. Please refer to the datasheet for details about the errata.

This document does NOT apply to revisions other than revision B of the XMEGA A3 family, neither to other revisions than revision B of the ATxmega256A3B.

2 Finding Revision of ATxmega Devices

- While reading the Signature of the device via JTAGICEmkII, the JTAG ID is in paranthesis and the first number in this shows the revision of the device (if it shows 6 it is revision G).
- It is possible to get the revision of the device by reading the REVID register. Please find this detail from Xmega A Manual --> REVID - MCU Revision ID.
- It is possible to get the device revision by reading the device signature. Please find these details from Xmega A Manual-->Device Identification register-->Version

3 Errata

The EEPROM and Flash cannot be written while reading EEPROM or Flash, or while executing code in Active mode

4 Errata Workaround

Enter IDLE sleep mode within 2.5 uS (Five 2 MHz clock cycles and 80 32 MHz clock cycles) after starting an EEPROM or flash write operation. Wake-up source must either be EEPROM ready or NVM ready interrupt. No other interrupt should have hi level set as if any other interrupts other than EEPROM/NVM wakes up the device from IDLE then the corresponding execution might not be successful.

Alternatively set up a Timer/Counter to give an overflow interrupt 7 ms. after the erase or write operation has started, or 13 ms after atomic erase-and-write operation has started, and then enter IDLE sleep mode.

5 Implementation of Workaround

The implementation of workaround is done in two section one for EEPROM and the other for FLASH. The associated code uses the option to of putting the CPU to SLEEP (IDLE) during a write operation to either EEPROM or FLASH.

5.1 EEPROM Programming

5.1.1 This sequence has to be followed for any of the following commands

- EEPROM ERASE
- EEPROM WRITE
- EEPROM ERASE WRITE

5.1.2 Additional Resources required

- EEPROM ISR has to be defined and inside the ISR the EEPROM interrupt has to be disabled
- Variables for saving SLEEP, PMIC Status, PMIC control, EEPROM Interrupt and SREG has to be declared

5.1.3 Sequence of instructions to be executed

- Load the NVM Command registers with corresponding command (ERASE, WRITE or ERASE WRITE) and NVM Address registers with the Address of the EEPROM where the data has to be written.
- Save the present status of SLEEP register
- Set IDLE mode in SLEEP register
- Save the PMIC Status and control registers
- Enable only the highest level of interrupts
- Save SREG for later use
- Enable global interrupts

- Set SLEEP enable
- Save eeprom interrupt settings
- Write the signature for change enable of protected I/O registers to the CCP register.
- Execute NVM Command
- Enable EEPROM interrupt
- SLEEP (Make sure to execute SLEEP after the Execute NVM Command within 2.5uS)
- After Wakeup by EEPROM interrupt restore the SLEEP register,PMIC Status, PMIC control and SREG

5.2 Flash Programming

5.2.1 This sequence has to be followed for any of the following commands

- ERASE FLASH PAGE
- WRITE FLASH PAGE
- ERASE & WRITE FLASH PAGE
- ERASE APPLICATION SECTION
- ERASE APPLICATION SECTION PAGE
- WRITE APPLICATION SECTION PAGE
- ERASE & WRITE APPLICATION SECTION PAGE
- ERASE BOOT LOADER SECTION PAGE
- WRITE BOOT LOADER SECTION PAGE
- ERASE & WRITE BOOT LOADER SECTION PAGE

5.2.2 Additional Resources required

- Variables for saving SLEEP, PMIC Status, PMIC control, SPM Interrupt and SREG has to be declared.
- A Function to prepare the CPU to sleep has to be defined and this should do the following
- Store the status of SLEEP, PMIC Status, PMIC control, SPM Interrupt and SREG has to be defined
- SPM ISR has to be defined and this should do the following
- SPM interrupt has to be disabled.
- Restore the previous status of SLEEP, PMIC Status, PMIC control, SPM Interrupt and SREG

5.2.3 Sequence of instructions to be executed

- Call the function to Prepare the CPU for sleep
- Load the NVM Command registers with corresponding command (ERASE, WRITE or ERASE WRITE
- Enable only the highest level of interrupts
- Set SLEEP enable

- Write the signature for change enable of protected I/O registers to the CCP register.
- **Execute SPM**
- Enable SPM interrupt
- **SLEEP (Make sure to execute SLEEP after the Execute NVM Command within 2.5uS)**
- After Wakeup the SPM ISR will restore the SLEEP register,PMIC Status, PMIC control and SREG

6 Driver Implementation

This application note includes a source code package with a basic Self-programming driver implemented in assembly with a C interface for ATxmega256A3, similar implementation can be done for other devices with the Errata. Please refer to the driver source code and device datasheet for more details.

6.1 Files

The source code package consists of three files:

- sp_driver.s/sp_driver.s90 – Self-programming driver source file
- sp_driver.h – Self-programming driver header file
- Flash_write_example.c – Example code using the driver

For a complete overview of the available driver interface functions and their use, please refer to the source code documentation.

6.2 Doxygen Documentation

All source code is prepared for automatic documentation generation using Doxygen.

Doxygen is a tool for generating documentation from source code by analyzing the source code and using special keywords. For more details about Doxygen please visit http://www.doxygen.org. Precompiled Doxygen documentation is also supplied with the source code accompanying this application note, available from the *readme.html* file in the source code folder.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN ATMEL'S TERMS AND CONDITIONS OF SALE LOCATED ON ATMEL'S WEB SITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.