

---

# AVR103: Using the EEPROM Programming Modes

## Features

- EEPROM access explained
- Up to 50% reduced EEPROM update times
- EEPROM driver implementation
- Backwards compatible code

## Introduction

This application note implements a driver utilizing the programming modes available for the EEPROM in some AVR parts, like ATmega48/88/168, ATtiny13, ATtiny2313 and ATmega256x. The driver saves both power and time by always using the most efficient programming mode when writing data to EEPROM memory. It ensures that EEPROM cells are only erased and programmed when it is required.

A detailed description of the EEPROM erase and program operations are given to understand the programming modes and their benefits. With some planning, using the driver in this application note can significantly increase performance in many applications, e.g. flag and parameter storage in EEPROM.



8-bit **AVR**<sup>®</sup>  
Microcontrollers

Application Note

**PRELIMINARY**

Rev. 2578A-AVR-03/05





## Theory of Operation

### EEPROM Access Explained

EEPROM memory consists of independent cells each representing a single bit. The cells are based on floating-gate transistor technology: An electrical charge trapped on the transistor gate determines the logic level of the cell. Slightly simplified, the way that the cell works can be described as follows: When erasing a cell, a charge is placed on the gate and the cell is read as logic one. Programming a cell is equivalent to discharging the gate, bringing the logic value to zero. It is only possible to program (discharge) a cell that has been erased (charged).

Note that even if EEPROM memory is accessed in bytes, bits can be programmed individually. Since only the bits being programmed are discharged (zero), the remaining unprogrammed bits are still charged (one). Any unprogrammed bit can be programmed at a later stage. Therefore, programming a byte that is already programmed, without erasing in between, will result in a bit-wise AND between the old value and the new value.

### EEPROM Programming Modes

In AVR devices that don't have selectable EEPROM programming modes, the 'write' operation is actually a combined erase and program operation. The memory byte to be written is always erased before programming the new value, even if no bits need to be erased. For instance, when programming the value 0x00 to an EEPROM byte already containing the value 0x0F, there is no need to erase the byte first. No bits need to be changed from 0 to 1. The erase operation in this case has no effect except consuming time and power. The combined erase and program operation takes twice the time of a simple program operation.

When designing with AVR devices that do have selectable EEPROM programming modes, e.g. ATmega48/88/168, it is possible to split the EEPROM write access into separate erase and program operations. This will reduce EEPROM update time by 50% in cases where an erase is not required. The available programming modes are listed in Table 1 below.

In this document the word 'program' is used to describe update without erase, and 'update' is used for any operation that changes EEPROM contents.

**Table 1.** EEPROM Programming Modes

Programming Mode	Resulting operation	Typical update time
0	Combined erase and program	3.4 ms
1	Erase only	1.8 ms
2	Program only	1.8 ms

The programming mode is set using the EEPM1 and EEPM0 bits of the EEPROM Control Register (EECR). The default mode at startup is mode 0, and the register is therefore compatible with software not using EEPROM programming modes.

### Efficient EEPROM Access

To benefit from the programming modes, the EEPROM software should use the modes so that only the required operation is performed. For instance, when updating an EEPROM byte, and no bits need to be changed from 0 to 1, an erase is not necessary.

When updating EEPROM bytes, the software should compare the bits of the existing value against the bits of the new value and decide which programming mode to use. Depending on the values written, the extra CPU cycles used to decide which mode to use is outweighed by the reduction in average update time.

## Implementation

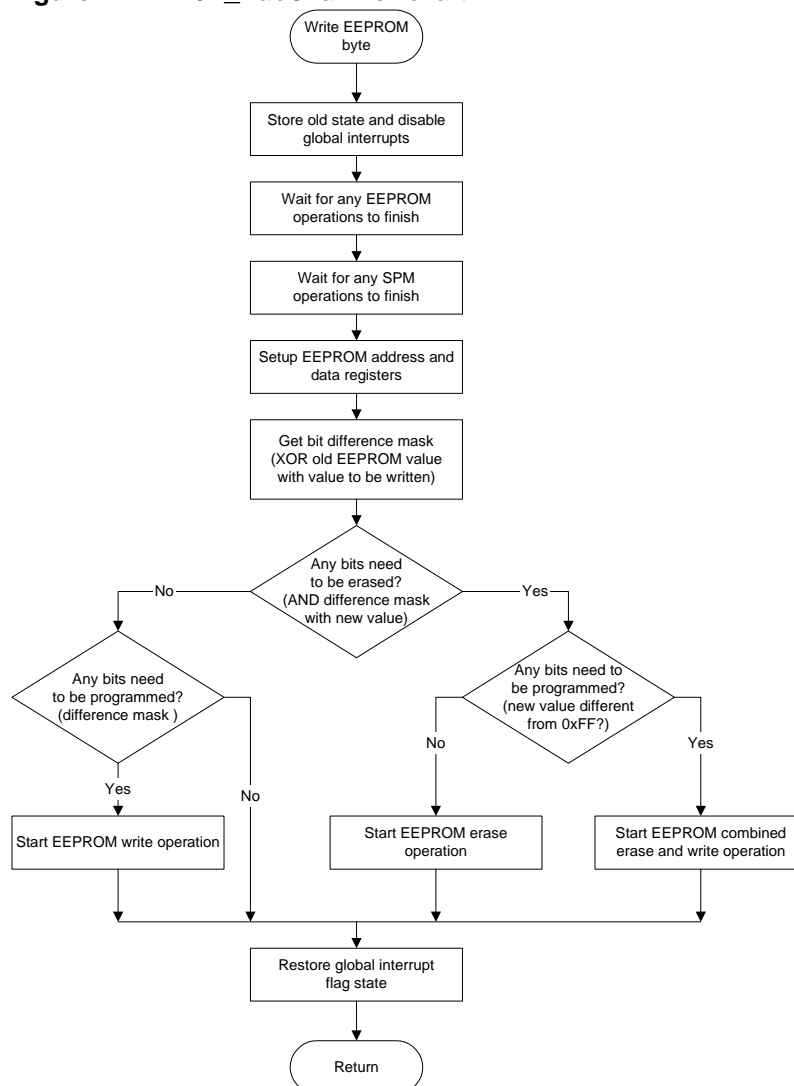
This application note provides a small test application to show how to use the programming modes. The source code implements EEPROM read and update functions and executes a number of EEPROM updates to demonstrate the different programming modes. Only the EEPROM update function is described in detail.

Refer to the document “source.doc” supplied with the code for source documentation.

### The EEPROM\_PutChar() function

This is the EEPROM update function. It compares the old and new bytes and uses the most efficient programming mode for the update operation. The flowchart is shown in Figure 1 below.

**Figure 1.** EEPROM\_PutChar flowchart



An alternative approach would be to perform a bitwise AND on the old and new values. If the result is equal to the new value, only a program operation is required. If different, use combined program and erase mode. This would save some CPU cycles



and code size, but would not benefit from the cases when old and new values are equal, or when only an erase operation is required.

## Execution times

For the most general case Flash self-programming operations are considered in the implementation. However, using the C compiler `#define` directive, the part of the code regarding SPM can be left out. Refer to the commented source code for more details.

In Table 2 below, the execution times for the general case is shown. The execution times for the maximum speed optimization settings are also shown. The values include the function call and return instructions and assume previous EEPROM and SPM write was finished. The 'Old EEPROM write' operation refers to the standard EEPROM write routine not using programming modes, which can be found as examples in the datasheets.

**Table 2.** Execution times

Operation	No optimization <sup>(1)(2)</sup>	Max speed optimization <sup>(1)(2)</sup>
Erase only	41	35
Write only	41	35
Erase & write	41	35
No operation	36	30
Old EEPROM write	26	24

Notes: 1: Values assume previous EEPROM and SPM write was finished.  
2: For total execution time refer to values in Table 1.

## Using EEPROM\_PutChar() in your own applications

This implementation of `EEPROM_PutChar` can replace existing implementations that do not use programming modes. The function takes care of selecting the best programming mode internally. For instance, the application note AVR104 "Buffered Interrupt Controlled EEPROM Writes" can be enhanced to use programming modes by replacing the two "Set EEPROM..." blocks in AVR104's Figure 4 with the `EEPROM_PutChar` code from this application note.

In many applications, significant benefits can be gained by carefully planning the way the EEPROM is used. You should try to avoid changing bits from 0 to 1 if you can. For instance, parameter storage in EEPROM can be enhanced with circular buffers in the same way as Flash memory is utilized in application note AVR105 "Power Efficient High Endurance Parameter Storage in Flash Memory". Flash memory cell erasing and programming are very similar to EEPROM cells.

## Literature References

- ATmega48/88/168 Datasheet (Rev. C)
- Application Note AVR104: Buffered Interrupt Controlled EEPROM Writes
- Application Note AVR105: Power Efficient High Endurance Parameter Storage in Flash Memory
- "Application-Specific Integrated Circuits" by Michael John Sebastian Smith



## Atmel Corporation

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

## Regional Headquarters

### Europe

Atmel Sarl  
Route des Arsenalux 41  
Case Postale 80  
CH-1705 Fribourg  
Switzerland  
Tel: (41) 26-426-5555  
Fax: (41) 26-426-5500

### Asia

Room 1219  
Chinachem Golden Plaza  
77 Mody Road Tsimshatsui  
East Kowloon  
Hong Kong  
Tel: (852) 2721-9778  
Fax: (852) 2722-1369

### 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

## Atmel Operations

### Memory

2325 Orchard Parkway  
San Jose, CA 95131, USA  
Tel: 1(408) 441-0311  
Fax: 1(408) 436-4314

### Microcontrollers

2325 Orchard Parkway  
San Jose, CA 95131, USA  
Tel: 1(408) 441-0311  
Fax: 1(408) 436-4314

La Chantreterie  
BP 70602  
44306 Nantes Cedex 3, France  
Tel: (33) 2-40-18-18-18  
Fax: (33) 2-40-18-19-60

### ASIC/ASSP/Smart Cards

Zone Industrielle  
13106 Rousset Cedex, France  
Tel: (33) 4-42-53-60-00  
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906, USA  
Tel: 1(719) 576-3300  
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park  
Maxwell Building  
East Kilbride G75 0QR, Scotland  
Tel: (44) 1355-803-000  
Fax: (44) 1355-242-743

### RF/Automotive

Theresienstrasse 2  
Postfach 3535  
74025 Heilbronn, Germany  
Tel: (49) 71-31-67-0  
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.  
Colorado Springs, CO 80906, USA  
Tel: 1(719) 576-3300  
Fax: 1(719) 540-1759

### Biometrics/Imaging/Hi-Rel MPU/ High Speed Converters/RF Datacom

Avenue de Rochepleine  
BP 123  
38521 Saint-Egreve Cedex, France  
Tel: (33) 4-76-58-30-00  
Fax: (33) 4-76-58-34-80

---

### Literature Requests

[www.atmel.com/literature](http://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. Atmel's products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

© Atmel Corporation 2005. All rights reserved. Atmel®, logo and combinations thereof, AVR®, and AVR Studio® are registered trademarks, and Everywhere You Are<sup>SM</sup> are the trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.