FEE alignments to reduce data loss through ECC

Version 1.0
2014-08-06
Application Note AN-ISC-8-1161

Author(s)	Goß, Michael

Restrictions	Customer confidential - Vector decides

Abstract	Alignments and ECC handling of flash devices have to be considered with the MICROSAR FEE configuration

Table of Contents

1.0 Overview

This application note addresses a hardware dependent topic of microcontrollers which may lead to a corruption of data entities in the flash memory. This document describes alignment attributes and ECC (Error Correction Code) error handling of the flash memory hardware and how the MICROSAR Flash EEPROM Emulation (FEE) has to be configured in order to avoid problems.

In section 2 the problem is illustrated on the basis of a particular microcontroller family, which is affected by this behavior.

Section 3 describes the MICROSAR FEE alignment characteristics and points out configuration details, which have to be taken into account.

In section 4 microcontroller families are listed which are known to show the described behavior.

2.0 Hardware Alignment Problem

The following hardware dependent problem is illustrated on basis of Freescale MPC560xB (Bolero) microcontroller family. Generally the characteristics of every microcontroller should be examined to the effect that the described problem is prevented.

In some microcontrollers, for instance in Freescale's MPC560xB family, the hardware specific write alignment differs from the read alignment. Particularly this can cause an issue if the read alignment is greater than the write alignment. In this case the treatment of ECC (Error Correction Code) errors by the microcontroller can result in corruption of data entities.

By way of illustration and for further considerations the alignment properties of MPC560xB family are depicted in the following table:

Write alignment	8 byte (64 Bit)
Read alignment	16 byte (128 Bit)

Table 1 – Write and read alignment of MPC560xB microcontrollers

In figure 1 the addressing schemes of this hardware is described in accordance with read- or write-accesses to the flash memory.

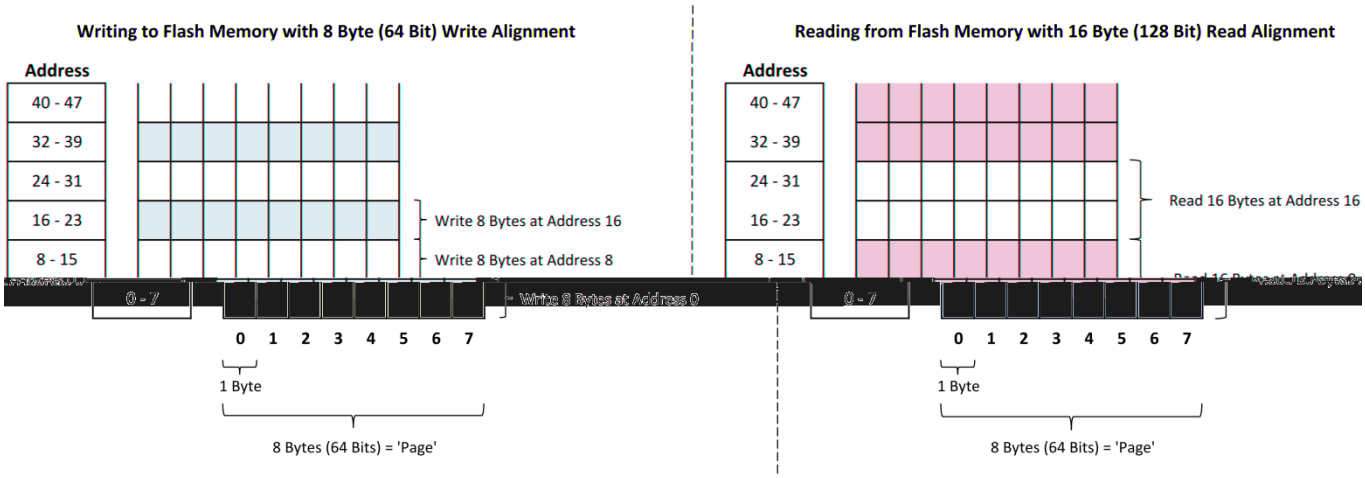

Figure 1 – Memory addressing

For further considerations: The smallest writable unit in this flash is called a page with a size of 8 byte.

The handling of programming (write) accesses differs from read accesses due to the alignment sizes. On the one hand, programming the flash memory is possible with a page size of 8 byte (64 Bit) and therefore the write addresses have to be aligned to 8 byte boundaries. On the other hand reading from the flash memory is hardware specifically aligned to 16 byte boundaries. In consequence, the data being read from the flash memory contains two pages of written data. It is required by AUTOSAR to provide memory abstraction and non-aligned memory access. Nevertheless when performing read jobs, this flash memory controller always accesses 16 byte-sized address spaces due to internal constraints. Therefore even if only one byte is requested by the application, the flash memory controller accesses 16 bytes of data. As a consequence misaligned read accesses to hardware may be accomplished by several 16 byte reads. The requested data then is extracted from the received packages and assembled correctly before passing it to the upper layer.

For each written page (8 byte) one ECC is calculated to add some redundancy to a data set, which can be used to check its consistency, and to recover data determined to be corrupted. The ECC implemented within the flash memory module will correct single bit failures and detect double bit failures. Correction of single bit errors takes place within the flash memory controller whereas the flash cell content remains unchanged. Depending on the used platform, e.g. MPC560xB, ECC error handling varies. For further considerations the ECC error handling is evaluated with regard to the microcontroller family MPC560xB.

If an uncorrectable ECC error occurs, an error response is signaled and the requested access is terminated with an error. This implies that an ECC error in one page results in an erroneous read operation which actually accesses two pages of written data. Hence reading from an address aligned to the 16 byte boundary fails if just one page therein contains an uncorrectable ECC error. As a result the second potentially error-free page within this read boundary will be treated as corrupted thereby as well.

The reason for an uncorrectable ECC error in a flash page is for example an aborted programming access due to a reset. Figure 2 illustrates how one ECC error affects read operations.

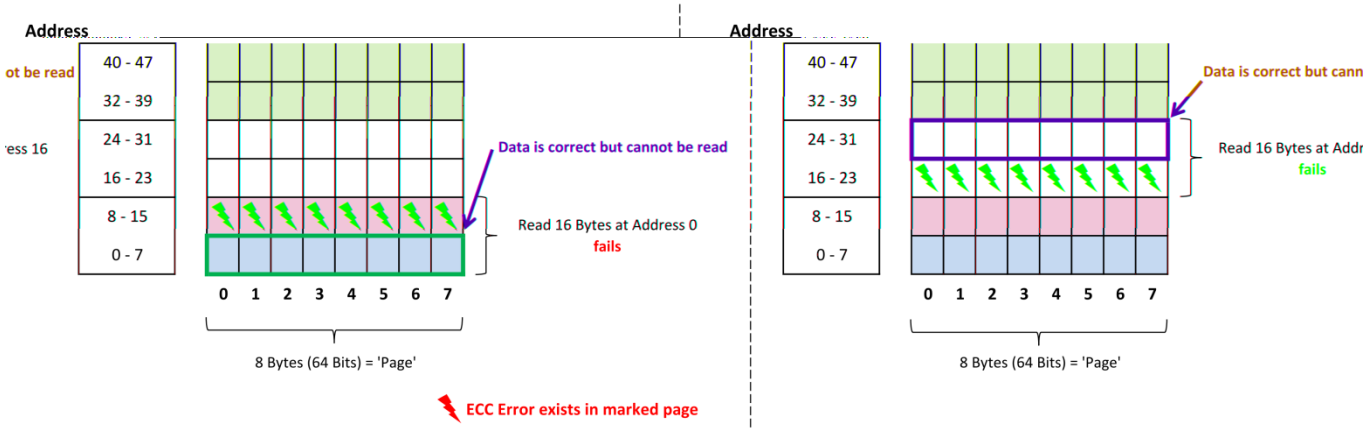

Figure 2 – ECC error affects read operation

Figure 2 depicts that one ECC error within a 16 byte address boundary leads to an unsuccessful reading. If one ECC error is received during read access, the flash memory controller does not update the read buffer. Thus reading from this address fails even if one of the two pages is consistent.

This means the occurrence of one uncorrectable ECC error in a page treats the neighboring page within the 16 byte address boundary also as corrupted. This situation is particularly problematic if the start addresses of two neighboring data blocks are not aligned to 16 byte address boundaries, as shown in figure 3.

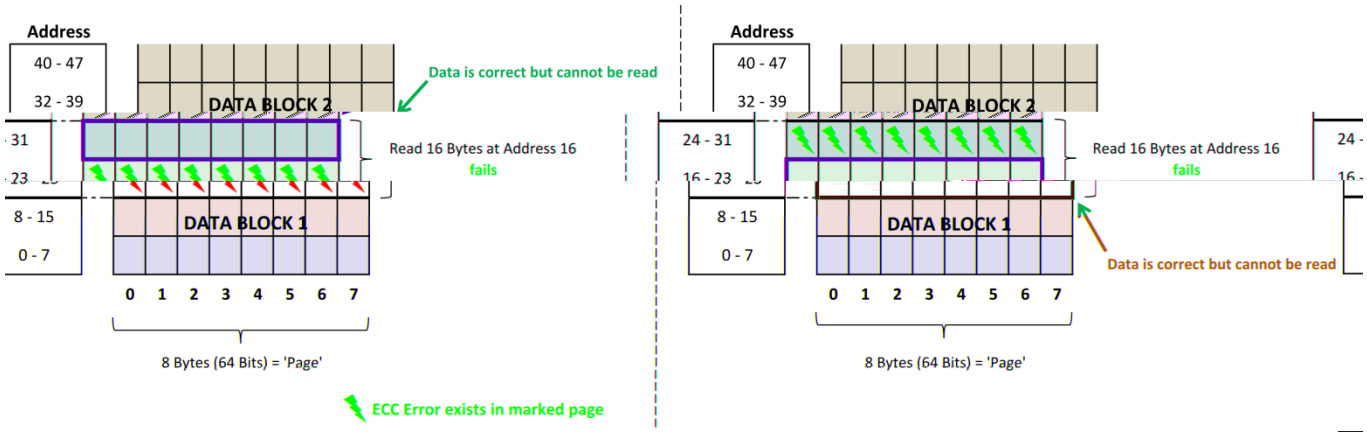

Figure 3 – Two data blocks not aligned to 16 Byte (128-bit) address boundaries

As depicted in figure 3, an ECC error in one data block may result in the corruption of another data block. Either the first page of data block 2 (left diagram) or the last page of data block 1 (right diagram) cannot be read in case of an ECC error in the other block. Usually the first and the last pages of a data block contain information (e.g. block id, block length) which is used for management purposes. If these pages cannot be read the entire data block may be corrupted.

An application using the MICROSAR FEE with such insufficient alignment settings may run in data consistency issues as described in the following example:

A (simplified) configuration contains one block with multiple instances. After writing this block several times, the programming access is aborted due to a reset while writing the first page of the block. In consequence the block with latest data was not written successfully and an uncorrectable ECC error occurs. Generally, when reading data of a specific block from flash memory, the MICROSAR FEE returns the content of the last written data. In case the

last written instance is corrupt, e.g. due to a write abort (reset), the MICROSAR FEE returns the second latest data. In this scenario the uncorrectable ECC error in the latest instance corrupts the second latest instance and thus the MICROSAR FEE returns the data of the third latest instance to the application. Thus it cannot be assured that the returned data is up-to-date.

A worst case scenario would be a completely corrupted flash image caused by an uncorrectable ECC error at particular positions. As a result, neither read nor write accesses to the flash memory would be possible.

Regarding the impact of this behavior on an application, if reading data from the flash memory the following results are possible:

- Successful reading, no ECC error occurred, last written data is returned to the application
- Successful reading, ECC error occurred, outdated data is returned to the application
- Unsuccessful reading, ECC error occurred, no data is returned to the application

3.0 FEE Alignment Configuration

The MICROSAR FEE (Flash EEPROM Emulation) implementation is capable of addressing this hardware behavior by providing two different alignment settings. It is distinguished between:

- Write Alignment
- Address Alignment

Following sections describe both FEE alignments' purposes.

3.1 FEE Write Alignment

The write alignment is used for all write requests the FEE issues to the flash. Both flash addresses and job lengths are always integer multiples of this value. Normally this value corresponds to the page size of the underlying flash device and describes the smallest writeable unit of the flash hardware. Regarding to the microcontroller family MPC560xB programming accesses to the flash are aligned to 8 byte (64 bit) address boundaries.

3.2 FEE Address Alignment

The address alignment influences the way FEE stores data entities in flash memory. Single entities, e.g. FEE sector header, will start at addresses that are an integer multiple of FEE address alignment. In this way data entities cannot influence other entities which are already stored in flash memory, especially if programming accesses get aborted (i.e. ECC error), e.g. due to a reset.

In figure 4 different settings of address alignments are depicted in order to show the effect of this configuration parameter.

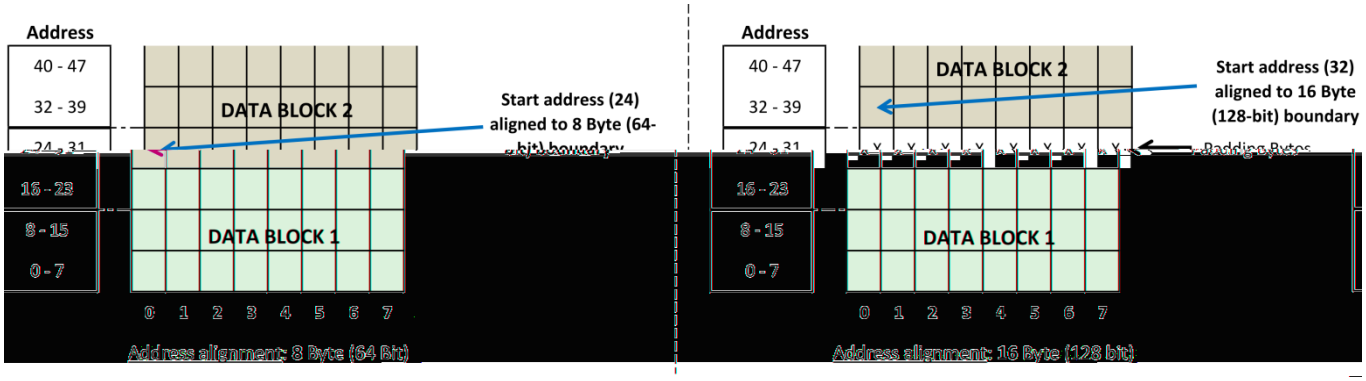

Figure 4 – Effect of different FEE Address Alignment configurations

Left diagram of figure 4 shows that the start address of each data block is aligned to an 8 byte (64 bit) boundary. Whereas the right diagram pictures a configuration in which the start address of each data block is aligned to a 16 byte (128 bit) boundary.

If performing read accesses to the flash memory, these configurations make a difference as soon as ECC errors are considered. As illustrated in figure 3 an ECC error in one page corrupts the neighboring page due to the hardware dependent size of read alignment.

Figure 5 points out that different data blocks have no influence on each other, if the address alignment setting corresponds to the hardware dependent read alignment.

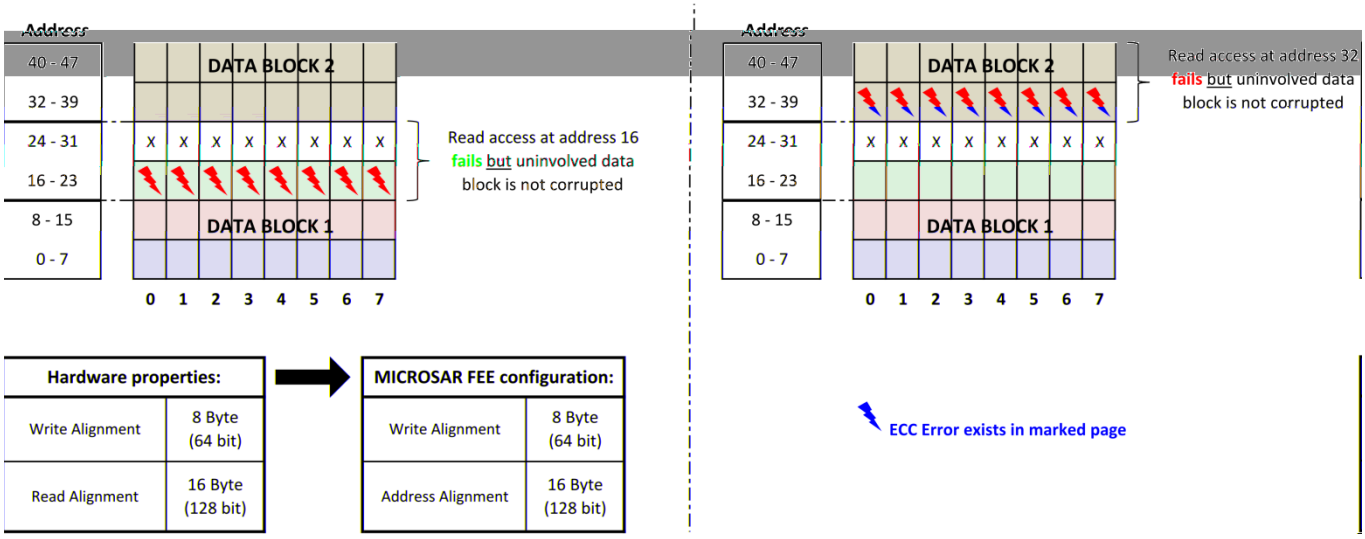

Figure 5 – ECC errors do not affect integrity of neighboring data blocks

In figure 5 two different situations are shown. On the one hand (left diagram) an ECC error in the last page of data block 1 occurred. Reading from this address will fail but the other data block is not affected by this. On the other hand (right diagram) an ECC error in the first page of data block 2 will not impair the integrity of data block 1.

It is shown that an appropriate configuration of FEE address alignment ensures that data blocks do not corrupt each other due to ECC errors. The start address of each data entity is aligned in a way that read requests always access only one block at a time.

3.3 Recommendation for FEE Alignment Configuration

As required by AUTOSAR, the MICROSAR FEE implementation is hardware independent. Still the MICROSAR FEE provides a configurable write alignment and address alignment in order to address hardware dependent characteristics. As it is illustrated using the example of MPC560xB platform, buffer sizes and alignments constraints of read operations may differ from programming operations. In order to configure MICROSAR FEE appropriately, it is recommended to check the hardware manual of the flash device for relevant data. Often this information does not come out of manuals clearly. In case of ambiguity it is recommended to contact the semiconductor manufacturer.

- **Configuring FEE write alignment:**
  The value of FEE write alignment should be set to the page size of the flash device in bytes. Note that FEE does not support write alignments smaller than 8 bytes due to internal reasons.
- **Configuring FEE address alignment:**
  The value of the FEE address alignment should be set to the read alignment of the flash device in bytes. Thus the start address of each data block is aligned in a way that ECC errors in one block do not corrupt

other blocks. Needless to say, FEE address alignment must not be configured smaller than FEE write alignment.

4.0 List of Affected Platforms

The following list makes no claims of being complete. Only platforms are listed which are known to be affected. In order to evaluate the problem regarding a specific non-listed platform, it is necessary to check the hardware manual of the flash memory device for write/read alignments and ECC error handling. To remove ambiguity it may also be useful to check with the semiconductor manufacturer.

List of affected microcontroller families:

- Freescale Bolero MPC560xB/C
- Freescale Pictus MPC560xP
- Freescale Spectrum MPC560xS
- Freescale Komodo MPC567xK
- Freescale Monaco MPC563xM

5.0 Additional Resources

For additional information and how the relevant data for configuration is extracted from HW manuals see reference manual of Freescale MPC5604BCRM microcontroller as an example:

In chapter 27.4.1 Module structure it is described how the flash memory is addressable for program (write) and read accesses.

The chapter 27.8.7 Flash error response operation summarizes the ECC error handling of this specific flash hardware.

VECTOR APPLICATION NOTE
AN-ISC-2-1100 ECC Error Handling on MPC55XX

6.0 Contacts

For a full list with all Vector locations and addresses worldwide, please visit http://vector.com/contact/.