AN2440
Application note

Embedded ECC in F8H process automotive EEPROM: device architecture and related application guidelines

Introduction

The Error Code Correction (ECC) is a specific logic, embedded in certain EEPROM devices, able to correct a single bit error when reading a binary word.

This embedded ECC algorithm is transparent for the Master bus and increases the data read robustness. However, in some specific cases, an EEPROM device featuring ECC behaves differently when compared to another without ECC.

The purpose of this application note is to describe the STMicroelectronics EEPROM ECC architecture in order to understand the possible differences in certain specific user cases.

This application note applies to the automotive F8H process products listed in *Table 1*.

Table 1. F8H process automotive products

Type	Serial interface	Root Part Numbers
Serial EEPROM	SPI bus	M95320-A125/A145 M95640-A125/A145 M95128-A125/A145 M95256-A125/A145 M95512-A125/A145 M95M01-A125/A145 M95M02-A125
	I^2C bus	M24128-A125 M24256-A125 M24512-A125 M24C64-A125 M24C32-A125 M24M01-A125 M24M02-A125

Contents

List of tables

List of figures

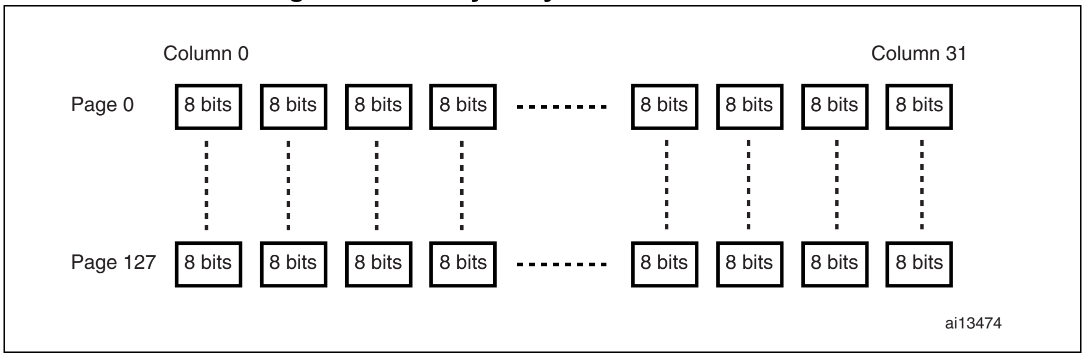
1 Standard EEPROMs

1.1 Memory array architecture

ST EEPROMs without the ECC feature are organized as pages (or rows) of 16, 32, 64, 128 or 256 bytes, depending on the maximum page size of the memory. For instance, a 32-Kbit EEPROM is organized as 128 pages of 32 bytes each.

Each single byte of the memory array can be changed independently from the others.

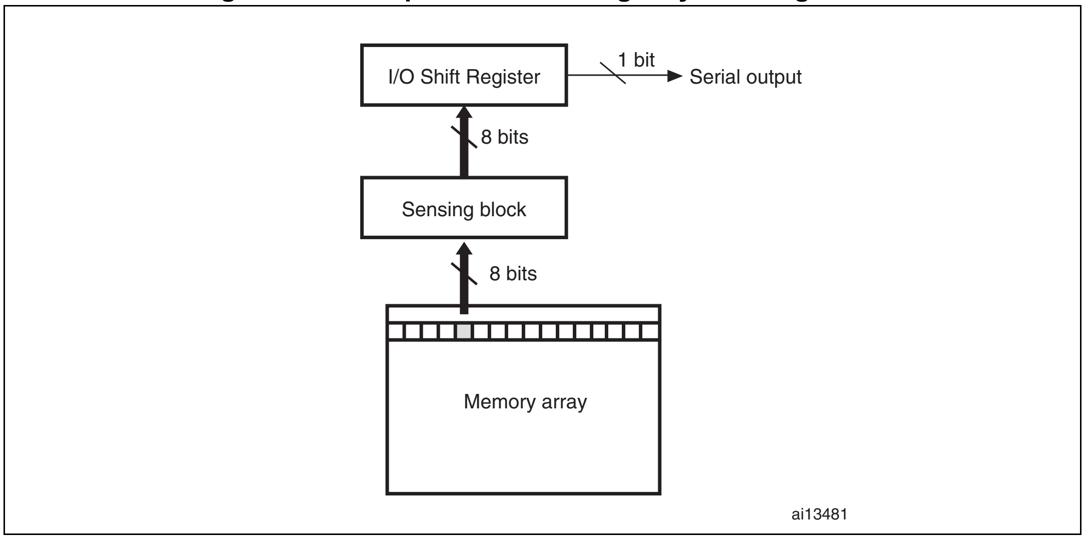
Figure 1. Memory array without ECC feature



1.2 Read operation with single byte management

The Read process is carried out on a byte-by-byte basis. *Figure 2* shows how each byte in the memory array is read individually.

Figure 2. Read operation with single byte management

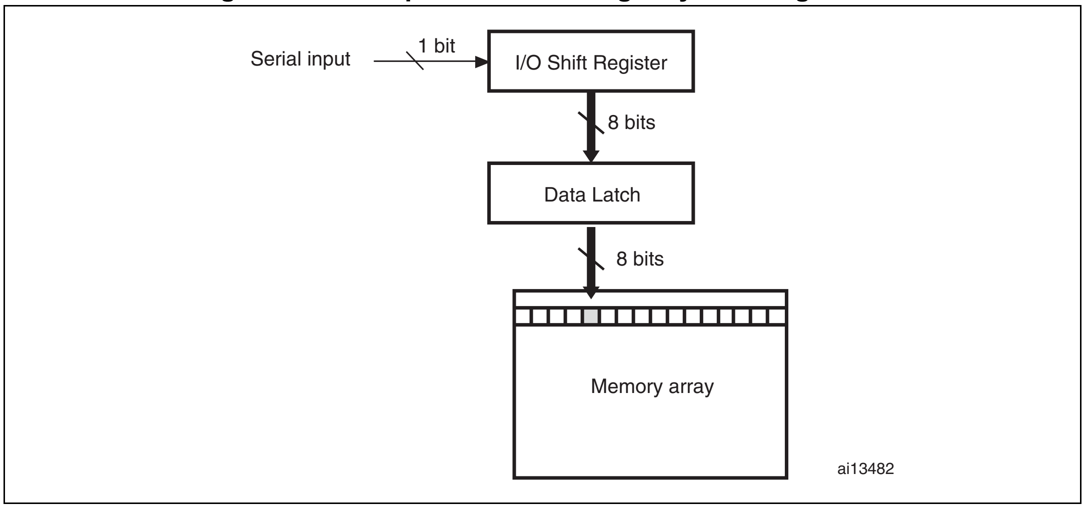


When a Read command is decoded, the addressed byte is pointed, sensed and shifted out. No other data byte is involved in the process.

1.3 Write operation with single byte management

The Write process is carried out on a byte-by-byte basis. *Figure 3* shows how each individual byte in the memory array is written.

Figure 3. Write operation with single byte management



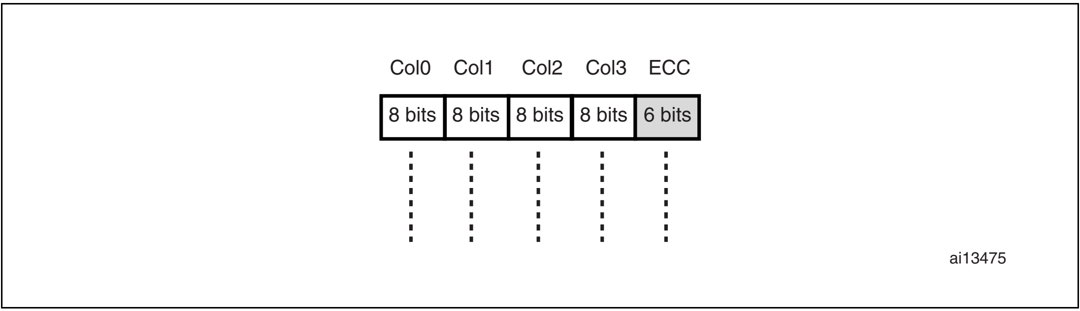
When a Write command is sent, the data byte to be written is serially shifted in, transferred to a data byte latch, and then programmed to the targeted byte location in the memory array. No other memory data byte is accessed during the process.

2 EEPROMs with embedded ECCx4 feature

2.1 Memory array architecture

ST EEPROM devices with the ECCx4 feature are organized in pages of 32, 64 128 or 256 bytes (depending on the memory density of the device). Inside a page, the bytes are packed by groups[a], each group being coupled with 6 ECC bits making up a word, as illustrated in *Figure 4*.

Figure 4. Word of 4 bytes with 6 ECC bits



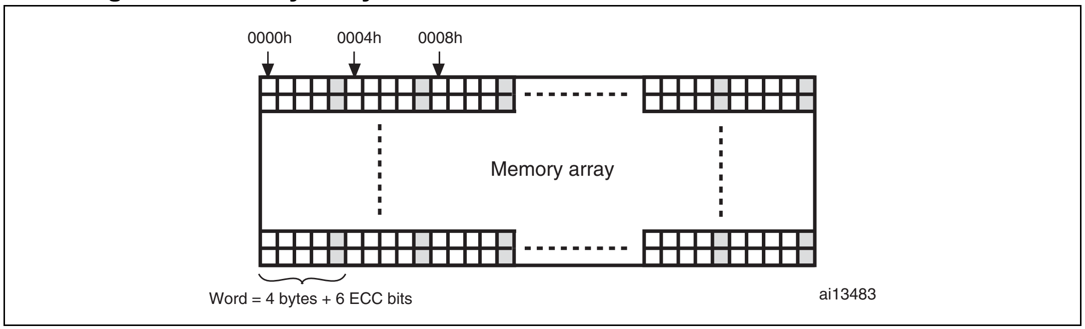
The memory array represented in *Figure 5* shows the memory architecture with embedded ECCx4. The data entity of the array becomes the [4 data bytes + 6 ECC bits]. ECC bits are standard non-volatile EEPROM bits.

The ECC bits are linked to the value of the 4 data bytes in the word; each time a data byte is written inside a word, the corresponding ECC bits are computed and updated. With 6 ECC bits, only one bit error in the 4 data bytes can be corrected.

Note: The ECCx4 cannot correct two or more bit errors, however the probability that 2 bits become weak inside the same word is extremely low (less than 10^{-10}).

The internal data is managed on a word-by-word basis (4 bytes + 6 ECC bits). This internal management is transparent to the Master bus as the device single byte granularity remains unchanged.

Figure 5. Memory array architecture of EEPROMs with embedded ECCx4



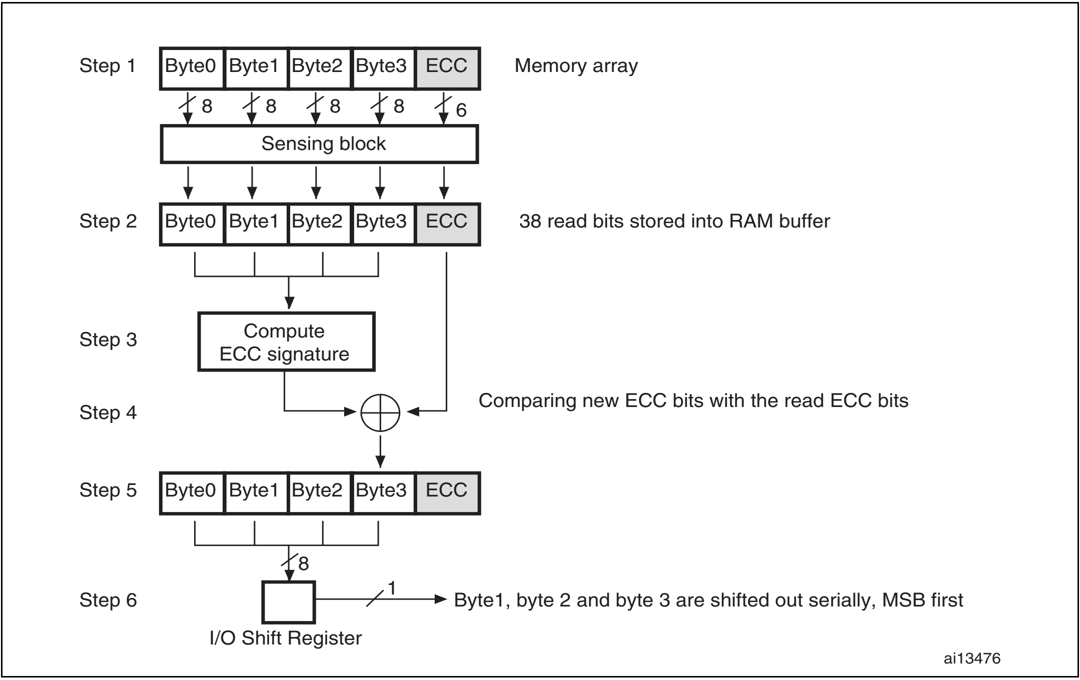
a. A group of four bytes is located at addresses [4*N, 4*N+1, 4*N+2, 4*N+3], where N is an integer.

2.2 Read operation with word management

Besides the additional ECC bits in the memory array, the EEPROM has an additional logic block able to compute the ECCx4 signature, and to detect and correct possible bit errors in the data word.

As the ECC signature (6 ECC bits) is a function of the 4 data bytes in the word, the word is always considered as the smallest entity in the Read process of the EEPROM. *Figure 6* explains how the Read process is managed.

Figure 6. Read process of an EEPROM with embedded ECCx4



The steps of the Read process illustrated in *Figure 6* are described below:

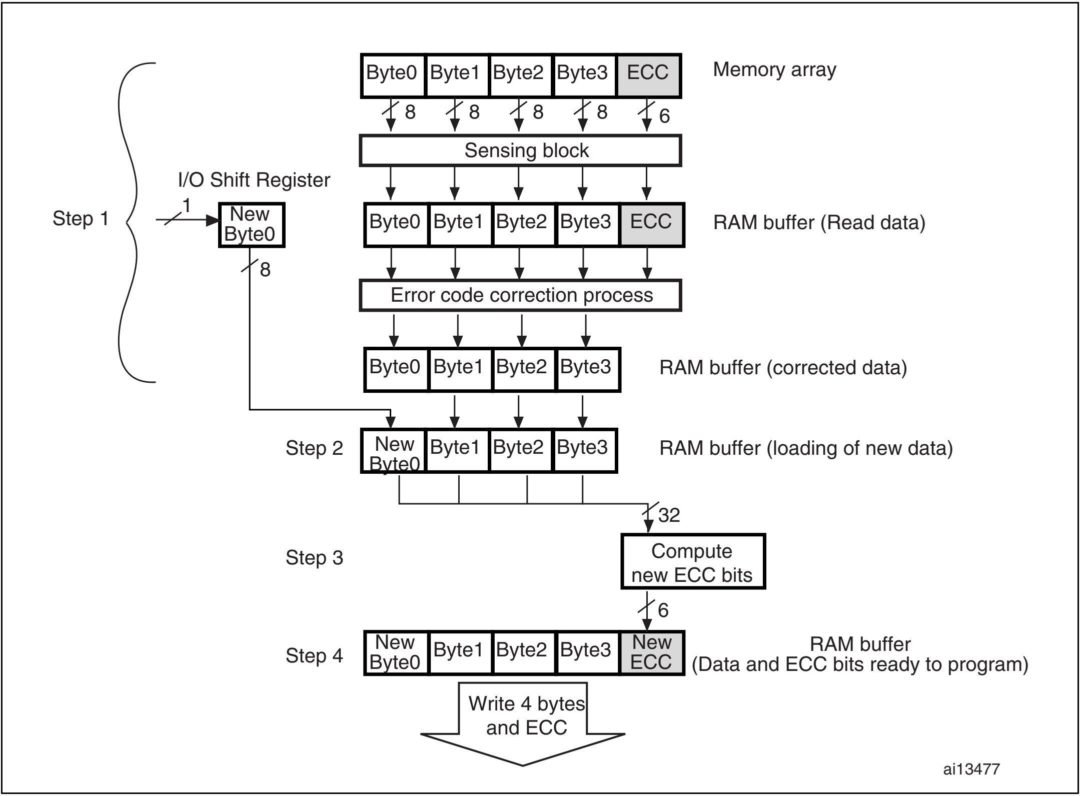
1. The user requests a reading of three bytes: byte 1, byte 2 and byte 3.
2. Internally, the entire word is read (4 data bytes and 6 ECC bits) and stored in a RAM buffer.
3. A new ECC signature is computed from the 4 data bytes read from the memory.
4. The new ECC signature is compared to the ECC bits read from the memory. A difference indicates that there is an error, the bit error is detected and corrected after further calculations.
5. The RAM buffer now contains byte 0, byte 1, byte 2 and byte 3 (byte 3 corrected) and the read ECC bits.
6. The data is shifted out of the memory with the Most Significant Byte first; that is, byte 1 followed by byte 2 and byte 3.

2.3 Write operation with word management

Besides the additional ECC bits in the memory array, the EEPROM has an additional logic block able to compute the ECC signature, and to detect and correct possible bit errors in the data word.

As the ECC signature (6 ECC bits) is a function of the 4 data bytes in the word, the word is always considered as the smallest entity in the Write process of the EEPROM. Figure 7 explains how the Write process is managed, to ensure byte granularity.

Figure 7. Write process of an EEPROM with embedded ECCx4



1. The user sends a request to write one byte (byte 0) into the memory array.
   a) The new data byte is shifted in the internal Shift Register.
   b) At the same time, the memory word (4 bytes + 6 ECC bits) in which the new data must be updated is read and stored into the RAM buffer (if one data bit is incorrect, the word is corrected as it would be done in a normal Read operation).
2. The new data byte 0 in the Shift Register is loaded at the correct position into the RAM buffer. The RAM buffer is loaded with the new data byte 0.
3. The new ECC bits corresponding to the 4 data bytes in the RAM buffer are calculated and loaded into the RAM buffer and replace the previous ECC bits.
4. The RAM buffer (4 bytes and 6 corresponding ECC bits) contents can now be written into the memory word.

Note: The update of one single byte triggers an internal process refreshing/re-writing the entire word.

3 EEPROM cycling performance

3.1 Standard EEPROMs

In standard EEPROMs (without ECCx4 logic), writing one byte only cycles this byte (without any consequence on contiguous bytes). When defining the application cycling budget, the designer has only to check that each byte in the EEPROM never exceeds the cycling performance specified in the datasheet.

3.2 EEPROMs with ECCx4 logic

In EEPROMs with ECCx4 logic, as the data architecture is based on words (word = 4 bytes[b] + 6 ECC bits), the application cycling budget has to be evaluated:

- If one (or more) byte in one word is cycled independently (single byte Write), we know that the ECC function also writes/cycles the three other bytes located in the same word. As a consequence, the cycling can be distributed over the 4 bytes (bytes 0, 1, 2 and 3) of the word. The cycling seen by the word is the sum of the cycles seen by byte 0, byte 1, byte 2 and byte 3. This sum must not exceed the cycling performance specified in the datasheet.

  <u>Example 1: Cycling equally each byte inside a word</u>

  – For a cycling limit of N cycles, each byte 0, byte 1, byte 2 and byte 3 of a word can be equally cycled N/4 times so that the word cycling budget is:
  N/4+ N/4+ N/4+ N/4 = N cycles.

  – For a cycling limit of 4 million cycles[c], each byte 0, byte 1, byte 2 and byte 3 of a word can be equally cycled 1 million times.

  <u>Example 2: Cycling unequally each byte inside a word</u>

  – For a cycling limit of N cycles, and for bytes inside the same word, byte 0 can be cycled A times, byte 1 can be cycled B times, byte 2 can be cycled C times and byte 3 can be cycled D times, so that the word cycling budget is A+B+C+D = N cycles.

  – For a cycling limit of 4 million cycles[c], and for bytes inside the same word, byte 0 can be cycled 2 million times, byte 1 can be cycled 1 million times, byte 2 can be cycled 1/2 million times and byte 3 can be cycled 1/2 million times.

- If the application writes data by word(s), the 4 bytes making up the word are always updated at the same time and the number of Write cycles is pulled to the highest cycling possible value. The application designer has only to check that each word in the EEPROM never exceeds the cycling performance specified in the datasheet.

  <u>Example: Cycling by word</u>

  – For a cycling performance of N cycles, the word (four bytes) can be cycled N times so that the word cycling budget is N cycles.

  – For a cycling performance of 4 million cycles[c], the word (four bytes) can be cycled 4 million times.

b. A group of four bytes is located at addresses [4*N, 4*N+1, 4*N+2, 4*N+3], where N is an integer.

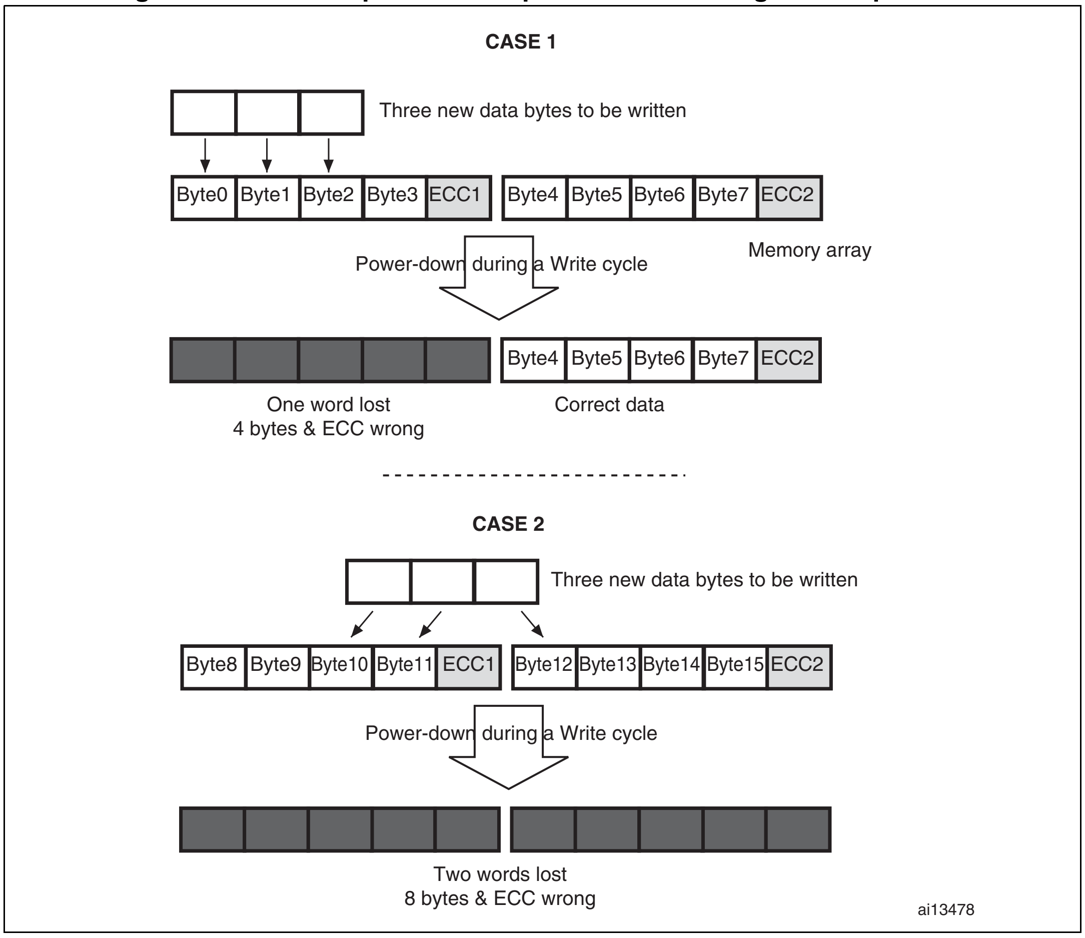
c. The cycling performance is specified in the datasheet.

4 Power loss and data corruption

4.1 Updating data

In EEPROMs without ECCx4, when an internal Write cycle (t_W) is triggered, only the addressed data bytes are accessed and, in case of a supply voltage drop during the internal Write cycle, the data corruption is limited to these addressed data bytes.(d)

Figure 8. Data corruption due to power-down during a Write process


CASE 1
Three new data bytes to be written
Byte0 Byte1 Byte2 Byte3 ECC1
Byte4 Byte5 Byte6 Byte7 ECC2
Memory array
Power-down during a Write cycle
Byte4 Byte5 Byte6 Byte7 ECC2
One word lost
4 bytes & ECC wrong
Correct data
CASE 2
Three new data bytes to be written
Byte8 Byte9 Byte10 Byte11 ECC1
Byte12 Byte13 Byte14 Byte15 ECC2
Power-down during a Write cycle
Two words lost
8 bytes & ECC wrong
ai13478

In EEPROMs with embedded ECCx4, when writing one (or more) byte inside a word, the four bytes making up the word will be written once the internal Write cycle (t_W) is triggered. If the supply voltage (V_{CC}) drops during the internal Write cycle, the four bytes of the word might be incorrectly written but the data corruption is limited to these addressed data bytes(d). *Figure 8* shows two examples of data corruption in an EEPROM with embedded ECCx4:

- Case 1 describes data corruption when the data to update belongs to the same word.
- Case 2 describes data corruption when the data to update belongs to two different words.

d. This is not tested in production but only characterized during the product qualification.

4.2 Read-only parameters

When using an EEPROM with an embedded ECCx4, it is recommended to pay particular attention to the data mapping of read-only bytes and read/write bytes.

Read-only data handle key values for the end application and they must never be changed, even if some uncontrolled events, such as a power-down during a Write cycle, occur.

As it is known that a power-down during a Write cycle can modify the four data bytes inside the addressed word(s), a word should not store both read-only byte(s) and read/write byte(s).

For a robust application design, it is necessary to define two separate families of data:

- Read-only data, which is only read during the application life
- Read/write data, in which the bytes can be updated during the application life

Read-only data must be stored in words different from the words in which the read/write data is stored (read-only data and read/write data must never be stored in a same word).

Thus, an unexpected voltage supply drop during a Write cycle will never corrupt read-only data and will limit the data corruption to only the addressed words(e).

e. This is not tested in production but only characterized during the product qualification.

5 Conclusion

When an EEPROM device embeds an ECCx4 function, the device operates on a word-by-word basis, while an EEPROM device without ECCx4 operates on a byte-by-byte basis.

For the EEPROM devices embedding the ECCx4 function:

1. Read reliability is improved as a 1-bit error (within 4 bytes) can be detected and corrected.
2. Each byte can be independently cycled, provided that the cycling budget for the associated word does not exceed the cycling performance defined in the datasheet.
3. If the supply voltage (VCC) drops during the internal Write cycle, the four bytes of the word might be incorrectly written. As a consequence, read-only parameter bytes must not share the same word with the read/write parameter bytes.

6 Revision history

Table 2. Document revision history

Date	Revision	Changes
24-Oct-2006	1	Initial release.
18-Apr-2012	2	Updated entire application with new information concerning embedded ECC on application cycling times.
10-Jul-2012	3	Modified document title and *Introduction*. Added *Table 1: F8H process automotive products*. Edited step 3 of *Section 5: Conclusion*.
15-Jan-2013	4	Edited *Table 1: F8H process automotive products*.
15-Feb-2016	5	Updated *Introduction* and *Table 1: F8H process automotive products* with introduction of 2 Mbit devices. Minor text edits across the whole document.