AN13566

ECC on i.MX 8 Series

Rev. 0 — 01 April 2022

Application Note

Contents

1 Introduction

1.1 Purpose

The Error Correcting Code (ECC) is a feature available on several i.MX 8 application processors to further enhance the data integrity of the data stored in the DRAM. This document provides information on the ECC functionality to help readers develop applications enabling the available ECC features.

The ECC features are only available on selected devices, such as:

- i.MX 8M Plus
- i.MX 8QuadXPlus / i.MX 8DualXPlus
- i.MX 8XLite (* PRE-PRODUCTION)

Other NXP processor families also have support for ECC, but they are not described in this document. Most DRAM chips include "internal" on-chip error-correction circuits and they are not described in this document.

1.2 Audience

This document is targeted for selected devices in the i.MX 8 family and intended for users who understand:

- ECC functionality available on the SoC
- Specific requirements and impacts of enabling the ECC
- How to start developing an application that utilizes ECC

The audience should understand basic memory architecture concepts and DRAM functionality.

1.3 Acronyms and abbreviations

Table 1. Acronyms and abbreviations

Acronym	Definition
DRAM	Dynamic Random Access Memory
DDRC	DDR (DRAM) Controller
DFI	DDR PHY Interface
ECC	Error Correcting Code
CAM	Content Addressable Memory

Table continues on the next page...

Table 1. Acronyms and abbreviations (continued)

Acronym	Definition
EDAC	Error Detection And Correction
FIT	Failures In Time
MTBF	Mean Time Between Failures
RPA	Register Programming Aid
RMW	Read Modify Write
DM	Data Mask
HD	Hamming Distance
ISI	Inter-Symbol Interference
SBR	ECC Scrubber
SCU	System Controller Unit
SCFW	System Controller Unit Firmware
SER	Soft Error Rate
SECDED	Single-Error Correction and Double-Error Detection
V2X	Vehicle to everything

2 Overview

2.1 Data corruption

In modern SoC designs using external DRAM, several mechanisms can lead to incorrect data being received by the processor:

- Alpha/cosmic particles/radiation
- Signal integrity/ISI/noise
- Retention or coupling faults
- Row hammering

Cosmic rays and other external events can cause corruption in DRAM cells by changing the charge levels in the capacitors. To address this problem, the ECC memory stores extra parity bits next to the data bits to correct these errors.

The DRAM memory may provide increased protection against soft errors by relying on error-correcting code. Such error-correcting memory (ECC) is desirable for systems requiring functional safety compliance (ISO26262), high fault-tolerant applications, and aviation/space applications due to increased radiation.

Because DRAM memories adopt smaller technology nodes, the probability of the Soft Error Rate (SER) increases. SER is the rate at which a device or system encounters (or is predicted to encounter) soft errors. It is typically expressed as the number of Failures In Time (FIT) or the Mean Time Between Failures (MTBF).

Applications requiring higher temperature operation must refresh the DRAM more often to ensure that the charge in the DRAM cell is maintained to prevent data corruption.

2.2 Error correction

Most DRAM chips include "internal" on-chip error-correction circuits, which allow systems with non-ECC memory controllers to still gain most of the benefits of ECC memory. An ECC-capable DDR memory controller can detect and correct errors using an "external" circuit between the CPU and the memory. The ECC data errors can be stored inside additional memory or inside the main DRAM memory array.

The i.MX 8 DDR/DRAM Controller (DDRC) supports Error Detection And Correction (EDAC) with a single-bit Single Error Correction/Double Error Detection Error Correction Code (SEC/DED ECC) for configurations where the DRAM data width is configured to 16 or 32 bits.

ECC can correct "n" bit errors (with n ≥ 1) and detect cases where more than "n" bits have flipped. For this purpose, ECC adds redundant ECC bits to every data word that “checks” the other bits. The combination of the data bits and the ECC bits is called a code word. ECC ensures that if any bit in a valid code word changes, it is no longer a valid code word.

- The Hamming Distance (HD) between any code word is at least 3, meaning that it can detect up to 2-bit errors and correct a single detectable error. However, distinguishing between a word that has a corruption of one bit and a message that has a corruption of two bits is not possible. To remedy this, hamming code can be extended by an extra parity bit. This way it is possible to increase the minimum distance of the hamming code to 4, which allows the decoder to distinguish between 1-bit errors and 2-bit errors. The decoder can detect and correct a single error and at the same time detect (but not correct) a double error.
- SEC/DED: The most common error-correcting code (SEC/DED hamming code) allows a 1-bit error to be corrected and (with an extra parity bit) 2-bit errors to be detected.

3 ECC

3.1 ECC schemes

There are two generally used ECC schemes (Inline ECC and Sideband ECC), which are mutually exclusive on the i.MX processors (see Figure 1).

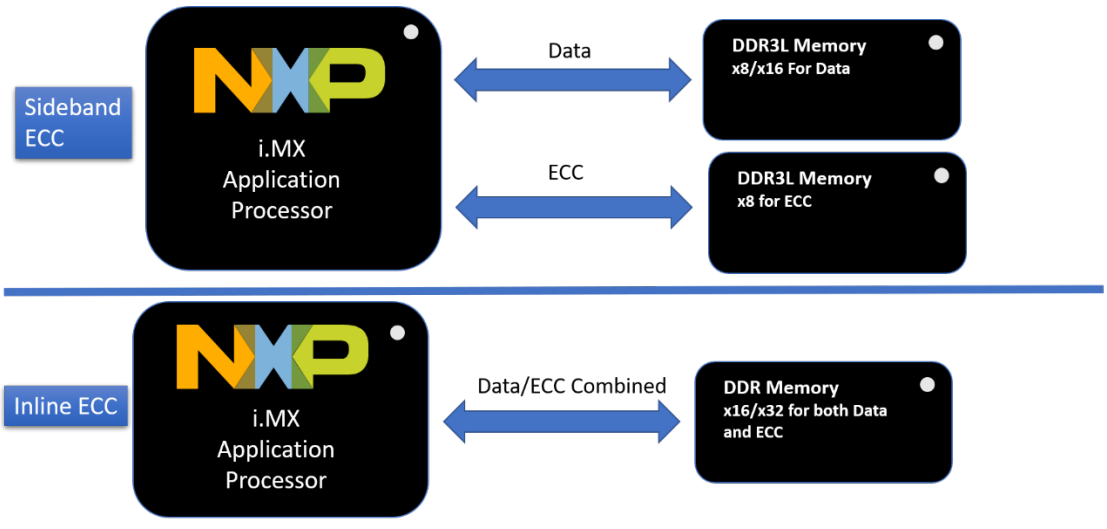

Figure 1. ECC architecture

The Inline ECC is an alternative to the Sideband ECC and it is supported on the following devices:

- i.MX 8XLite – LPDDR4, DDR3L – 16-bit
- i.MX 8M Plus – LPDDR4, DDR4 – 32-bit

The Sideband ECC is supported on the following device and specific memory configuration:

- i.MX 8QuadXPlus and i.MX 8DualXPlus – DDR3L – 40-bit

NOTE

The Sideband ECC is not supported on the i.MX 8DualX processor, because it only supports a 16-bit DDR interface. The 17 x 17 mm 0.8 mm FCPBGA package options do not support the ECC due to the 16-bit DDR interface.

Table 2. Comparing Inline ECC and Sideband ECC

	Sideband ECC	Inline ECC
Supported devices	i.MX 8QuadXPlus DDR3L 40-bit (32 + 8 ECC) i.MX 8DualXPlus DDR3L 40-bit (32 +8 ECC)	i.MX 8XLite (16-bit) LPDDR4, DDR3L i.MX 8M Plus (32-bit) LPDDR4, DDR4
ECC data	Stored in separate DRAM	$1/8^{th}$ density reserved for ECC data on single DRAM
Extra DRAM required	Yes	No
Data-to-ECC ratio	8/1	8/1
Hamming code	64/8 SECDED	64/8 SECDED
Performance impact	Yes (lower impact than Inline ECC)	Yes
Data mask	Optional	Required

In the Inline ECC correction case, extra memory cycles are used to store the ECC data to the existing memory devices. For the Sideband ECC (used on i.MX 8QuadXPlus/i.MX 8DualXPlus DDR3L), the data is stored using separate pins to additional DRAM devices. In both cases, the storage ratio is 8 bits of ECC for every 64 bits of data using a SECDED (Single Error Correct Dual Error Detect) hamming code.

NOTE

The Sideband ECC support is mutually exclusive with the Inline ECC support. i.MX 8 devices that support the Inline ECC do not support the Sideband ECC and vice-versa.

3.2 Inline ECC

The Inline ECC does not require an additional data bus for ECC so the actual DRAM data width is equal to "DRAM_DATA_WIDTH". The ECC parity is stored with the data without using a dedicated sideband memory device.

NOTE

The "DRAM_DATA_WIDTH" term will be used to refer to the bus width used to store actual data (not ECC) in the DRAM memory.

When the Inline ECC is enabled, the 3 highest column bits must be programmed to be mapped to the highest address map position possible. The controller’s flexible address-mapping scheme is constrained so that the highest system address space is reserved for the ECC parity and waste as a single region. For the normal data in all remaining regions, the system address is linear and continuous. The valid data is not the full size of the DRAM.

The following are the features of the Inline ECC:

- ECC parity (code) is stored together with the data without using a dedicated sideband memory device.
- Supported with LPDDR4, DDR4, and DDR3L protocols.
  - The supported memory data widths are 16 and 32.
- 64/8 SECDED hamming code is used:
  - The data-to-ECC ratio is 8/1.
- It requires the Data Mask (DM) to be enabled.
- The RMW command is required when a write access cannot fill one hamming code (64 bits).
- It is suited for LPDDR4 due to device characteristics and device topology (sideband is not practical).

3.2.1 Enabling Inline ECC

Like the whole ECC functionality, this feature is optional and must be specifically enabled by the user. If a user wants to use the Inline ECC feature, they must enable it via the "Register Configuration" tab (Figure 2) of the respective i.MX 8 Register Programming Aid (RPA) tool available from the NXP community pages.

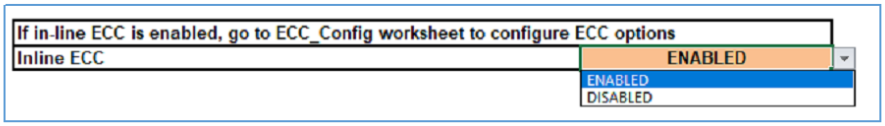

Figure 2. Enabling Inline ECC

When the ECC functionality feature is enabled, it cannot be dynamically disabled by the user. If any of the ECC configuration settings must be modified, a new register programming aid script must be regenerated.

3.2.2 Selecting the ECC regions

When the ECC featured is enabled, further ECC configuration is performed in the "ECC_Config" tab of the respective i.MX 8 Register Programming Aid (RPA) tool.

The ECC can be selected for the critical areas of the memory and deselected for others. The ECC sections are mapped to the top 1/8 of the system address, while the remaining space can be mapped into 7 selectable regions and an “other region” if the selected region size does not cover the whole space. See ECC application considerations to select which regions to protect and for other application-specific optimizations.

For each bank, the lower 7/8 of each row address is data and the upper 1/8 (differentiated by column address) is the ECC space. The data is stored in the lower 7/8 of the memory and the related ECC (1/8 the size) is stored in the equivalent section of the upper 1/8. There is a small region (1/8 of 1/8 - 1/64) of the memory which is wasted (the address range for the memory is allocated to the ECC space).

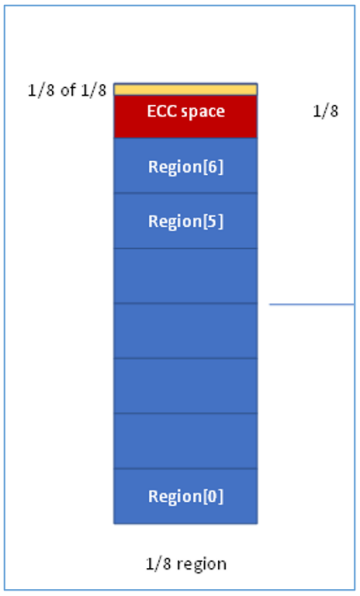

Figure 3. ECC regions

Each region can be 1/8, 1/16, 1/32, or 1/64 of the total memory map. The remaining memory area that is not covered by the 7 regions is referred to as the “other” region, with the exception of the 1/8 granularity, because there is no remaining memory area for this granularity setting. The user has the option to configure the “other” region with or without ECC protection.

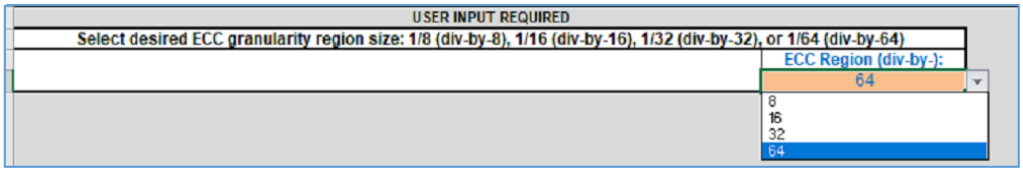

Figure 4. Selecting ECC granularity

The 7 regions (the address is based on the number of regions starting from the lowest address) may be configured to have or not to have the ECC protection. Without ECC protection, the performance overhead is not encountered. No memory space is recovered, it is simply not used. In the example shown in Figure 5, region 0 is configured to be ECC-protected at the beginning of the DRAM address range.

Main Memory Region	Start Address of each Main Memory Region	Density of each Main Memory Region	User Input ECC Protection Configuration for each Main Memory Region
Other Region	0x087000000	784MB	UNPROTECTED
Region6	0x086000000	16MB	UNPROTECTED
Region5	0x085000000	16MB	UNPROTECTED
Region4	0x084000000	16MB	UNPROTECTED
Region3	0x083000000	16MB	UNPROTECTED
Region2	0x082000000	16MB	UNPROTECTED
Region1	0x081000000	16MB	UNPROTECTED
Region0	0x080000000	16MB	PROTECTED
Total DRAM density:	1024MB (8Gb)		

Figure 5. Selecting ECC regions

The corresponding protected ECC parity sections for this example memory (as used on the i.MX8 DualXLite EVK) are as follows:

ECC Parity Region Section	Start Address of each ECC Parity Region Section	Density of each ECC Parity Region Section	ECC Parity Region Section memory attributes
ECC Parity Region 0 Section	0x0BFE00000	2MB	ACCESSIBLE
ECC Parity Region 1 Section	0x0BFC00000	2MB	ACCESSIBLE
ECC Parity Region 2 Section	0x0BFA00000	2MB	ACCESSIBLE
ECC Parity Region 3 Section	0x0BF800000	2MB	ACCESSIBLE
ECC Parity Region 4 Section	0x0BF600000	2MB	ACCESSIBLE
ECC Parity Region 5 Section	0x0BF400000	2MB	ACCESSIBLE
ECC Parity Region 6 Section	0x0BF200000	2MB	INACCESSIBLE
Other Region ECC Parity Region Section	0x0B9000000	98MB	ACCESSIBLE
Always user accessible	0x0B8000000	16MB	ACCESSIBLE

Figure 6. Corresponding ECC parity sections

Each ECC parity section memory may be user-accessible, depending on the corresponding region's ECC protection scheme. The inaccessible ECC parity section 0 is always at the top of the DRAM memory map, as shown in Figure 6. This configuration has the advantage of only reserving the topmost memory, leaving the rest accessible to other applications in a contiguous manner.

NOTE

The user software must ensure that no applications access the regions mapped as inaccessible. Attempting to access an inaccessible section results in a data abort.

3.2.3 Region locks

When the ECC is enabled, the data parity regions (up to 1/8 of the total DRAM density) are inaccessible to user software. The ECC region is locked by programming the "ECC_REGION_PARITY_LOCK" register bit. This bit defaults to a locked position when the ECC is enabled. When set, it locks the parity section of the ECC region (hole), which is the highest system-address part of the memory (ECC parity protected region).

The waste region can also be locked to ensure that waste regions are not user-accessible. The "ECC_REGION_WASTE_LOCK" register bit locks the remaining waste parts of the ECC region (hole) that are not locked by "ECC_REGION_PARITY_LOCK".

The ECC regions are normally locked and protected by the controller to avoid accesses. However, if it is necessary to inject errors (to test the mechanism) it may be done by unlocking and then writing the ECC region directly.

The recommended LOCK configurations are as follows:

`ECC_REGION_PARITY_LOCK = 1` - to prevent access to the ECC parity region

`ECC_REGION_WASTE_LOCK = 0` - to allow access to the ECC waste region

NOTE

To maximize memory usage when the ECC is enabled, users may access the waste region by unlocking it. However, users will be responsible for their software applications to not access the reserved (inaccessible) regions. By default, the RPA enables access to the waste region (this waste region is unlocked by default).

3.2.4 Non-binary density

A non-binary memory is a memory with densities of 6, 12, 24, or 48 Gb. When such densities are used, a special address-mapping register must be programmed in the DDRC. For ECC, when using a DRAM memory with a density of 6, 12, 24, or 48 Gb, special considerations are required and a separate "ECC config" tab must be selected by the user.

`ADDRMAP6.LPDDR4_6gb_12gb_24gb = 0` - use binary-aligned worksheet

`ADDRMAP6.LPDDR4_6gb_12gb_24gb = 1,2,3` - use non-binary-aligned worksheet

NOTE

The RPA asserts a warning if an incorrect worksheet is used, based on the `ADDRMAP6.LPDDR4_6gb_12gb_24gb` setting. See Binary/non-binary density for more details and recommendations on this topic.

3.2.5 Operation flow

The ECC generation and checking are handled automatically by the controller, generating separate commands to store and read the ECC value and data.

For the "read" operations, the ECC is read and stored internally. The address is calculated from the main read data. If the relevant ECC data is already loaded, no read is required. The read data is scheduled after the read ECC operation so that the read data can be corrected using the already present ECC data before being returned to the requestor.

For writes, the ECC is calculated when it is written to the memory. After the write occurs, the write ECC data (at the calculated address) is written. For cases where less than a full write for ECC is required, the ECC is written via a masked write (which must be enabled) rather than a read/modify/write operation on the ECC data. If partial writes on the data are required (if only part of a double word is written), then a read/modify/write operation is required on the data to ensure that full double words are written for the calculated ECC to be correct.

3.2.6 Scrubber

The i.MX DDR Controller (DDRC) provides a comprehensive solution to automatically correct 1-bit errors in the DRAM. The ECC Scrubber (SBR) is a block that initializes the ECC values of the protected DRAM regions and then initiates periodic background read commands to the DDRC. The scrub command is a read of 1 memory burst (for example, BL8), which is sent to the DDRC periodically with the lowest priority.

NOTE

ECC Scrubber (SBR) is different from the ECC Scrub feature, which is supported only in the Sideband ECC configurations.

The scrubber's function is to ensure that a single ECC error does not accumulate by correcting the data and writing back to the memory. There is a difference in the mechanism for the Inline and Sideband ECC mechanisms.

In the Inline ECC configurations, the ECC Scrub feature of the controller is not supported and it cannot scrub for each correctable read error. For this reason, the Read/Modify/Write (RMW) command is initiated by the scrubber itself for every single bit of the ECC error detected.

In the Inline ECC mode, ECC Scrubber (SBR) generates addresses only within protected regions. It automatically skips the unprotected and ECC regions. The ECC Scrubber (SBR) does not send transactions to invalid addresses, but it skips to the next valid address in the next cycle. In hardware-controlled low-power modes, the SBR continues to operate automatically without any software intervention. When using the Inline ECC mode, the RPA ensures that the SBR is configured to cover all protected regions.

For our purposes, when an ECC single-bit error is detected, the supplied data is corrected and sent to the requestor. However, this data is not written back to the memory. Because the SBR is constantly running over the entire protected space, reading the data and ECC and performing a check, it will eventually come across the erroneous data word. When the scrubber detects a correctable error, then a single RMW operation is scheduled with no valid data. It reads the memory checks, corrects the data, and performs a write back to the memory with the data corrected. This runs periodically with a programmable time between the reads and covers a specified address range. When "Scrub_Burst" is programmed, the SBR automatically ensures these "back to back" transactions followed by a long waiting time. It performs "n" transactions and waits for "n" intervals. This is helpful so that SBR does not constantly interrupt the system traffic.

SBR programming

The scrubber range, SBR burst interval, and other parameters are pre-configured in the RPA tool for optimal operation. Details of these programmable registers will be included in the respective SoC reference manuals.

SBRCTL - Scrubber Control Register – used to program the scrub interval, scrub burst count, and to enable the scrub function in different modes.

SBRSTAT - Scrubber Status Register - used to check on the status of scrub commands and reports busy and done on the initiated scrub functions.

SBRWDATA0 - the ECC Scrubber writes the data pattern for the ECC initialization of the data bus[31:0].

SBRWDATA1 - the ECC Scrubber writes the data pattern for the ECC initialization of the data bus[63:32].

NOTE

Additional scrub start and range functions are only for debugging purposes and not required for normal operation.

The scrubbing mechanism occurs during the DDR initialization and periodically, as described below.

Scrubber performed during DDR initialization

When the ECC is enabled, the scrubber is performed during DDR initialization to initialize protected regions of the DDR with valid data and ECC. The scrubber is performed only on ECC-protected regions. This is automatically configured based on the DDR RPA tools for a particular SoC.

- In the case of i.MX 8XLite and i.MX 8QuadXPlus/i.MX 8DualXPlus, the Register Programming Aid (RPA) tool automatically populates scrubber register writes based on which regions are protected.
- In the case of i.MX 8MPlus, the DDR Stress Test tool generates a "*.c" file that contains the function calls to scrub the protected regions.

In either case, no user interaction is needed as these mechanisms are configured automatically, based on the DRAM parameters and ECC-protected region configuration in the respective RPA tool.

Scrubber performed periodically

When the ECC is enabled, the scrubber periodically performs background read commands, but only on the ECC-protected regions. This mechanism is enabled by default (in the RPA tool) when the ECC option is enabled. In addition, other fields such as the scrub interval are also pre-configured for optimal operation. No further user configuration is required.

3.2.7 Reporting ECC errors

The DDR controller provides an ECC error-reporting mechanism using interrupts. There are several errors related to the Inline ECC mechanism that are mapped:

- `ECC_NCORRECT_INT`: An uncorrectable error is detected.
- `ECC_CORRECT_INT`: A correctable error is detected.
- `ECC_AP_ERR_INT`: An uncorrectable error leading to an address-protection fault is detected.

This is distinguished from the previous uncorrectable error because the number of errors is larger (greater than *ECCCFG0.ecc_ap_err_threshold*) indicating a data mismatch.

The ECC interrupts are mapped and can be configured through the SoC Global Interrupt Controller (GIC).

The application software in the respective interrupt-handling routines must decide the specific actions to be taken on the ECC interrupts being set. See Error reporting and actions. The example interrupt mapping is shown below for the i.MX 8XLite and i.MX 8MPlus SoCs.

Interrupt Name	Description	Interrupt Request (IRQ) Number
ECC_CORRECT_INT	A correctable ECC error has been detected	100
ECC_NCORRECT_INT	A non-correctable ECC error has been detected	101
ECC_AP_ERR_INT	A non-correctable error such that an address channel error is suspected	104

Figure 7. i.MX 8XLite ECC Interrupt Map

IRQ	Module	Logic	Interrupt Description
147	DDR	OR	DRAM Controller Error Interrupt for address protection fault.
147	DDR	OR	DRAM Controller Error Interrupt for correctable ECC error detected
147	DDR	OR	DRAM Controller Error Interrupt for uncorrectable ECC error detected

Figure 8. i.MX 8M Plus ECC Interrupt Map

NOTE

The assignment of the interrupts varies by SoC and users should see the respective reference manual for further information.

Error mapping:

The 64/8 SECDED hamming code can detect/correct 1-bit errors and detect 2-bit errors within a 64-bit word (8 bytes). The following conditions generate the following Inline ECC error generation:

- Access a 64-bit word with a single erroneous bit – correctable error
  - `ECCSTAT.ecc_corrected_err` will be set to 1
  - `ECCERRCNT.ecc_corr_err_cnt` will indicate the number of correctable ECC errors detected
- Access a 64-bit word with 2 erroneous bits – uncorrectable error and bus fault
  - `ECCSTAT.ecc_uncorrected_err` will be set to 1
  - `ECCERRCNT.ecc_uncorr_err_cnt` will indicate the number of uncorrectable ECC errors detected
- Access a 64-bit word with more than two erroneous bits. In this case, the number of errors exceeds the 64/8 SECDED hamming code and the result is indeterministic. Any of the following may be reported: correctable error, uncorrectable error, or no error.
- Access a 64-bit word with two erroneous bits and there are more than `ECCCFG0.ecc_ap_err_threshold` uncorrectable errors for the corresponding burst – AP error, uncorrectable error, and bus fault.
  - `ECCAPSTAT.ecc_ap_err` = 1
  - `ECCSTAT.ecc_uncorrected_err` = 1

Burst granularity:

The ECC engine checks all words in a burst. If the DRC client accesses a correct 64-bit word but there is an incorrect 64-bit word in the same burst, an error will be reported for the incorrect word, even though this is not the word being read.

If the error is uncorrectable, no bus fault will be reported because the actual data being read is not corrupted. If more than `ECCCFG0.ecc_ap_err_threshold` uncorrectable errors are found in a burst but those errors do not affect the 64-bit word being accessed, an AP error and an uncorrectable error will be generated, but no bus fault.

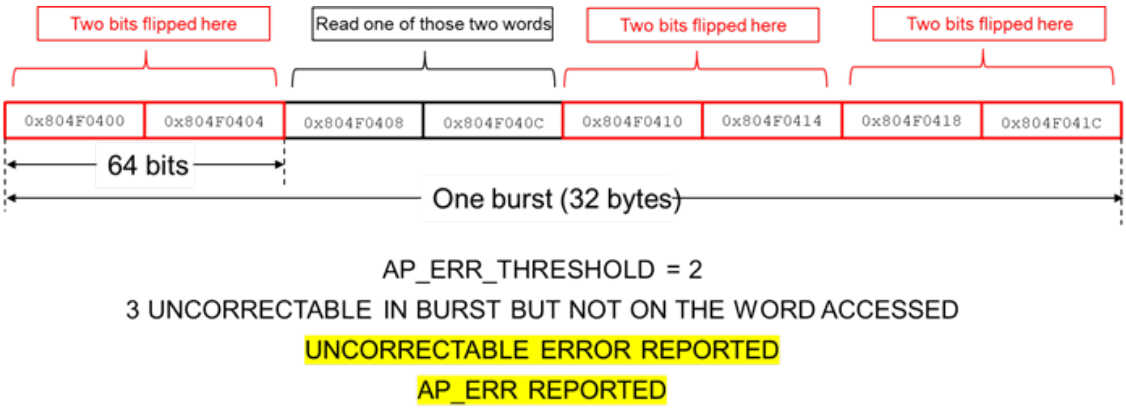

Figure 9. Example mapping shown for an ECC AP error

The following section provides more details on the ECC AP errors.

3.2.7.1 Reporting ECC AP errors

In addition to reporting ECC correctable and uncorrectable errors, the i.MX 8 DDR controller supports a third error-reporting mechanism (ECC AP error). In this condition, if the number of 64-bit words with 1-bit or 2-bit errors exceeds the value programmed in `ECCCFG0.ecc_ap_err_threshold` within a burst, then the `ECCAPSTAT.ecc_ap_err` flag will set. To determine how to configure `ECCCFG0.ecc_ap_err_threshold`, the users must first ascertain the number of 64-bit words within a burst (total number of ECC checks within 1 burst). There are two parameters needed to determine this:

- DRAM data width: 16-bit or 32-bit
- DRAM Burst Length (DRAM BL)
  - LPDDR4 uses a burst length of 16
  - DDR3L uses a burst length of 8

The equation to determine the total number of ECC checks within one burst:

- The total number of ECC checks within one burst = (DRAM Data width x DRAM BL)/64.

The following table shows the “Total number of ECC checks within one burst” for each supported memory and DRAM Data width:

Table 3. Total number of ECC checks within one burst based on memory type and DRAM data bus width

Memory type and burst length	16-bit data bus width	32-bit data bus width
LPDDR4 (BL16)	4	8
DDR3L (BL8)	2	4

To detect an ECC AP error, it is recommended to set `ECCCFG0.ecc_ap_err_threshold` as the total number of ECC checks within one burst – 1. The following table shows the recommended setting for `ECCCFG0.ecc_ap_err_threshold` based on memory type and DRAM data width:

Table 4. Recommended setting for ECCCFG0.ecc_ap_err_threshold based on memory type and DRAM data bus width

Memory type and burst length	16-bit data bus width	32-bit data bus width
LPDDR4 (BL16)	3	7
DDR3L (BL8)	1	3

NOTE

The SoC-specific RPA tool preconfigures the `ECCCFG0.ecc_ap_err_threshold` field to the recommended values, so no further user interaction is required.

3.2.7.2 ECC error reporting examples

This section provides illustrative examples of the various ECC error-reporting mechanisms.

The following examples are based on the following system conditions:

- DRAM type: LPDDR4
- DRAM data width: 16 bits
- DRAM Burst Length (BL): 16
  - LPDDR4 burst length = 16
- Total number of ECC checks within 1 burst: 4
  - (DRAM data width x DRAM BL)/64 = (16 x 16)/64 = 4
- Recommended `ECCCFG0.ecc_ap_err_threshold` setting: 3
  - Total number of ECC checks within 1 burst: 1

Example 1: ECC correctable error detected (1-bit error)

The following ECC status bits will be set:

- `ECCSTAT.ecc_corrected_err` = 1

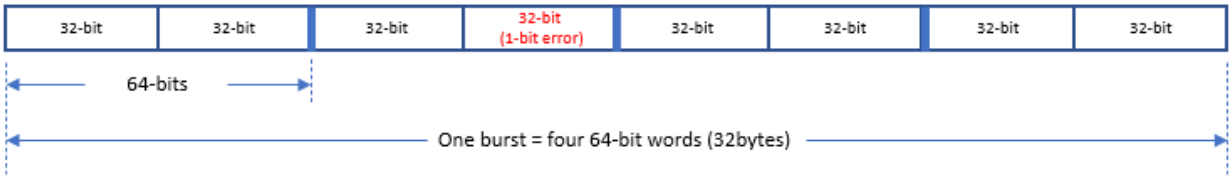

Figure 10. ECC correctable error detected (1-bit error)

Example 2: ECC uncorrectable error detected (2-bit error)

The following ECC status bits will be set:

- `ECCSTAT.ecc_uncorrected_err` = 1

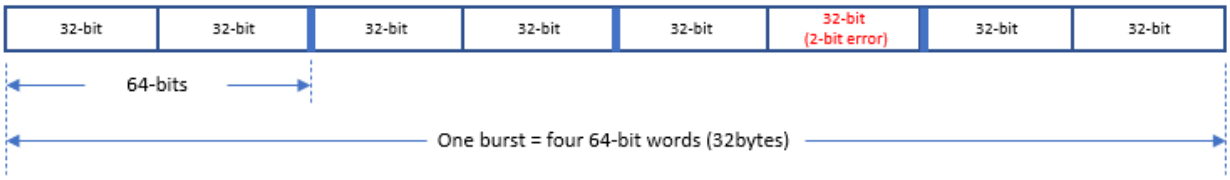

Figure 11. ECC uncorrectable error detected (2-bit error)

Example 3: ECC correctable and uncorrectable errors detected

The following ECC status bits will be set:

- `ECCSTAT.ecc_corrected_err` = 1
- `ECCSTAT.ecc_uncorrected_err` = 1

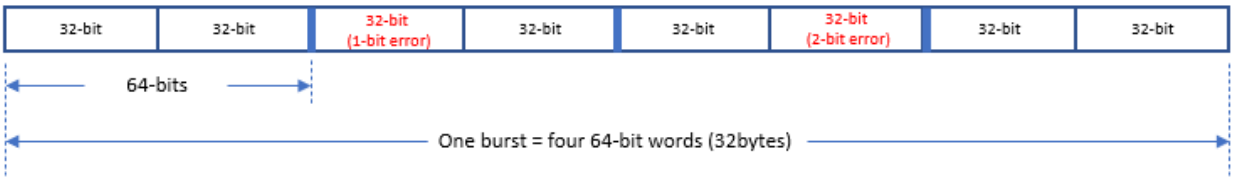

Figure 12. ECC correctable and uncorrectable errors detected

Example 4: ECC AP error detected

The following ECC status bits will be set:

- `ECCSTAT.ecc_corrected_err` = 1
- `ECCSTAT.ecc_uncorrected_err` = 1
- `ECCAPSTAT.ecc_ap_err` = 1 (exceeds `ECCCFG0.ecc_ap_err_threshold` setting: 3)

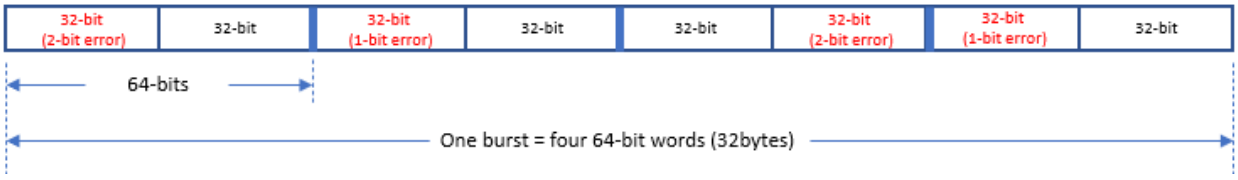

Figure 13. ECC AP error detected

3.2.8 ECC error interrupt configuration

The previous section describes the various ECC errors and their ability to generate interrupts. This section describes how to enable and clear these interrupts within the i.MX 8 DDR controller. This section does not describe how to enable and handle interrupts at the SoC level via the Global Interrupt Controller (GIC).

The ECC Clear Register (ECCCTL) is where the user can:

- Enable the ECC error-reporting interrupts
- Clear the ECC errors and the currently stored ECC error count
- Force an ECC error to test the interrupt-handling mechanism

The following bullets describe how to enable the various ECC error interrupts in the ECCCTL register. For each of the following bullets, set the corresponding bit to enable the desired interrupt and clear the bit to disable the desired interrupt:

- ecc_ap_err_intr_en (bit[10]): interrupt-enable bit for ecc_ap_err_intr
- ecc_uncorrected_err_intr_en (bit[9]): interrupt-enable bit for ecc_uncorrected_err_intr
- ecc_corrected_err_intr_en (bit[8]): interrupt-enable bit for ecc_corrected_err_intr

The following bullets describe how to clear the various ECC interrupts and status mechanisms in the ECCCTL register. For each of the following, set the corresponding bit to clear the desired ECC status:

- ecc_ap_err_intr_clr (bit[4]): interrupt-clear bit for ecc_ap_err
  - When set, ECCAPSTAT.ecc_ap_err is cleared
- ecc_uncorrected_err_clr (bit[1]): interrupt-clear bit for ecc_uncorrected_err
  - When set, ECCSTAT.ecc_uncorrected_err is cleared
- ecc_uncorr_err_cnt_clr (bit[3]): clears the currently stored uncorrected ECC error count
  - When set, ECCERRCNT.ecc_uncorr_err_cnt is cleared
- ecc_corrected_err_clr (bit[0]): interrupt-clear bit for ecc_corrected_err
  - When set, ECCSTAT.ecc_corrected_err is cleared
- ecc_corr_err_cnt_clr (bit[2]): clears the currently stored corrected ECC error count
  - When set, ECCERRCNT.ecc_corr_err_cnt is cleared

There is an optional feature that allows users to force an interrupt to test the interrupt-handling capabilities of their system software. For each of the following bullets, set the corresponding bit in the ECCCTL register to force the desired ECC error interrupt:

- ecc_ap_err_intr_force (bit[18])
- ecc_uncorrected_err_intr_force (bit[17])
- ecc_corrected_err_intr_force (bit[16])

3.2.9 ECC error injection through software

The ECC error injection is a useful optional feature for system-level software validation. Unlike the Sideband ECC, there is no dedicated hardware support for it. However, errors can be injected through the software by unlocking the ECC region through the "ECC_REGION_PARITY_LOCK" register and overriding ECC parity bits. When the corresponding addresses are read from a protected memory region, ECC errors are generated as correctable or uncorrectable, depending on the type of error introduced. Further information on this feature is available on request.

NOTE

ECC data poisoning is not supported by the DDR controller. The reference manual will be updated to remove this functionality.

3.3 Sideband ECC

The Sideband ECC is only supported on a 40-bit (32 + 8) i.MX 8QuadXPlus/8DualXPlus with DDR3L.

When the Sideband ECC is enabled, an additional data bus is used for the ECC. The actual DRAM data width is greater than the current "DRAM_DATA_WIDTH". When enabled, it widens the DDR PHY Interface (DFI) data width to accommodate the extra ECC bytes. 1 ECC byte is added per 1 ECC lane.

NOTE

The Sideband ECC is mutually exclusive with the Inline ECC.

Adding the Sideband ECC to LPDDR4-based systems is difficult, because the JEDEC standard calls for 16-bit LPDDR4 memory devices. In such configuration, more than half of the memory is unused in the Sideband ECC configuration. The ECC byte does not use the full width of the uppermost device.

Adding the Sideband ECC to DDR3L is not so difficult, because 8-bit memory devices are readily available. In the Sideband ECC configurations, the DDR controller can issue RMW commands if 1-bit ECC errors are detected with the "read" commands.

If `ECCCFG0.ecc_mode` is “100”, the 1-bit SECDED ECC is enabled. In this mode, the DDR controller performs the following functions:

- On writes, the ECC is calculated across each ECC lane and the resulting ECC code is written as an additional byte along with the data in the ECC lane. This additional ECC byte is always written to the uppermost byte of the DRAM.
- On reads, the ECC lane (including the ECC byte) is read from the DRAM. This is then decoded. A check is performed to verify that the ECC byte is as expected, based on the data in the ECC lane. If it is correct, the data is sent to the SoC as normal.

3.3.1 ECC Scrub

The ECC Scrub feature is supported only in the Sideband ECC configurations (i.MX 8QuadXPlus and i.MX 8DualXPlus – DDR3L – 40 (32 + 8) bit). It can be enabled by setting `ECCCFG0.dis_scrub` to 0. The same register bit can be used to disable this feature. When this feature is enabled, the DDR controller schedules the ECC Scrub operation when a single-bit error is detected. The scrub is executed as a new RMW operation to the location that resulted in a single-bit error.

When a single-bit error is detected, an RMW operation is scheduled by allocating the entries reserved in the write and read Content Addressable Memory (CAM) with the address that resulted in the single-bit error. Like the regular RMW requests, the "write" part of the scrub RMW is enabled only after the "read" part of the scrub RMW is scheduled and the read data is returned, corrected by the ECC decoder, and written to the appropriate location in the write data buffer.

At any time, only one outstanding ECC Scrub operation is allowed. While a scrub operation is pending, a new single-bit error detected by the read decoder engine does not cause the ECC Scrub operation to be initiated. The controller cannot handle more than one scrub operation at any time. The ECC Scrub RMW operation is initiated by the controller. There is no read response sent out for the scrub RMW operation (because this is not initiated by the SoC core).

NOTE

In the Inline ECC configurations (i.MX 8MPlus), the ECC Scrub functionality is disabled and does not scrub automatically for each correctable read error. (`ECCCFG0.dis_scrub` should be set to 1).

4 ECC application considerations

Although the ECC protection may seem like a win-win proposition to improve data integrity, there are several important considerations when using ECC features. This section briefly describes some important system-level design considerations which may have a varying impact, depending on the end-user application and product goals.

4.1 What to protect

The code and data that need integrity protection are the primary candidates for ECC protection. As the next sections describe, an ECC protecting large sections of DRAM has an impact on boot time and performance. It is highly recommended that users optimize the data to be protected and limit it to the smallest memory footprint possible. A functional safety assessment should be used to identify which code should be protected via the ECC. For such applications, it is also important to ensure that the entire data path is protected.

ECC only guarantees data integrity and should not be used as a mechanism to secure code or achieve data authenticity. ECC can be used to ensure that no data corruption in code is used for security applications, such as Arm Trusted Firmware or other encrypted data. ECC should not be used to protect against other side-channel attacks. ECC and/or increased refresh rates may make some side-channel techniques more difficult to use.

4.2 Error reporting and actions

The DDR controller provides an ECC reporting mechanism using interrupts. Several interrupts can be enabled to indicate when ECC errors are discovered. The end user may develop actions that map to these errors. The actions required depend on the type of ECC error and how the ECC feature is used in the end application.

4.3 Performance

ECC blocks are mapped so that the ECC for data is always mapped to the same page (bank and row) for DDR efficiency (because each data access may have a separate ECC access). ECC accesses are generally optimized along with the bank accesses they are related to. If you have multiple accesses to consecutive data, the ECC operations are merged.

Even with these optimizations, the use of inline ECC will have a performance impact of up to 25 % which will vary depending on the different types of accesses and types of DDR used. It varies depending on single reads vs. burst reads, partial writes vs. burst writes, first read vs. read from a buffer, and so on. For writes we approximate a 90 % efficiency. For reads, there is a latency impact of 4-8 DDR cycles and the data transfer efficiency drops to 90 %.

NOTE

There is no associated memory performance impact for any portion of DDR which does not have ECC enabled. The performance overhead is only associated with ECC areas. The protected regions should be minimized.

Users should expect a performance drop, evaluate if this is acceptable, and characterize the performance based on their specific applications. Similar performance degradation can be expected with the Sideband ECC.

4.3.1 Power

Enabling the ECC may contribute to additional power consumption due to the error-correcting circuitry and more energy consumption for the same workload. The power overhead can be minimized by limiting the ECC regions to be protected.

Low-power entry and exit times may also increase with the Scrubber functionality enabled. Users must have some controls to determine the Scrubber operation in low-power modes. The default configuration uses the automatic hardware control low-power functionality.

4.4 Boot time latency

When enabling the ECC, the ECC Scrubber must be enabled. It performs a mandatory ECC initialization that increases the boot time. Customers with boot time latency requirements should keep in mind that extra latency is incurred when enabling this feature.

ECC protection of the entire memory map may increase the initialization time significantly. NXP recommends limiting the ECC-protected region as much as possible to reduce the time to initialize the DDR when ECC is enabled.

Example:

- Scrubber times are based on the NXP LPDDR4 board running at 1200 MHz.
- Based on the NXP LPDDR4 EVK and 8 Gb (1 GB) configuration, tRFC is 280 ns.

Table 5. Boot time latency

ECC granularity	Size of 1 region (MB)	Scrubber time for 1 region (µs)	Scrubber time for 1 region calculated per MB (µs)
1/8	128	34601	270.32
1/16	64	17301	270.33
1/32	32	8652	270.38
1/64	16	4327	270.44

NOTE

In the above example, the approximate scrubber rate is 270 µs per 1 MB. For an ECC granularity of a 1/64 region with all regions being ECC-protected and with other regions being protected (the protected DDR is 896 MB), the total time is 242 ms.

Because the scrubber time depends on the memory tRFC value, a higher memory density (with a larger tRFC value) has a longer initialization time.

4.5 Memory

Several memory considerations that must be evaluated when enabling the ECC are described below.

4.5.1 Binary/non-binary density

As described in ECC, enabling the ECC with a non-binary-density DDR (such as 3 GB or 6 GB) requires additional considerations. Non-binary density requires multiple sections to be reserved and fragmenting the DDR memory space with non-contiguous memory sections. This requires software applications to navigate around these inaccessible regions (holes). Ensure that your specific applications do not access these regions. Otherwise, you will encounter a data abort.

Memory utilization and inaccessible regions increase, forcing a higher percentage of the DRAM memory to not be utilized. Due to this complexity and reduced memory utilization, it is strongly recommended to use binary memory densities (such as 1 GB, 2 GB, and 4 GB only) to use the Inline ECC feature.

4.5.2 Density

When enabling the Inline ECC, reserve at least 1/8 of the total available DRAM density for the parity data. The amount of memory to reserve varies depending on the ECC-protected region. Options have been presented to minimize the reserved memory regions. Other implications on reducing the reserved region are discussed further on in this document. For the Sideband ECC, a separate DRAM memory is required and the memory density must be adjusted accordingly.

4.5.3 Contiguous memory map

As described in ECC, the configuration of ECC-protected memory sections can impact the DRAM memory address ranges or regions that are inaccessible to application software. Let us consider an example where the user wants to ECC protect a single region of 16 MB. The RPA is configured to protect region 6, starting at address 0x8600 0000.

Main Memory Region	Start Address of each Main Memory Region	Density of each Main Memory Region	User Input ECC Protection Configuration for each Main Memory Region
Other Region	0x087000000	784MB	UNPROTECTED
Region6	0x086000000	16MB	PROTECTED

Figure 14. RPA region

The inaccessible parity regions will be a 2 MB section of memory starting at address 0xBF20 0000.

Main Memory Region	Start Address of each Main Memory Region	Density of each Main Memory Region	User Input ECC Protection Configuration for each Main Memory Region
Other Region	0x087000000	784MB	UNPROTECTED
Region6	0x086000000	16MB	PROTECTED

Figure 15. Inaccessible parity regions

The inaccessible ECC parity region 6 section is in the middle of the memory map and creates a discontinuity-inaccessible memory for the application.

NOTE

For a contiguous memory space, we recommend that the corresponding parity sections at the end of the DRAM memory address range are protected from region 0 and reserved.

In Figure 16, only region 0 is ECC-protected, which only reserves the ECC parity region 0 and leaves the remaining contiguous region available.

ECC Parity Region Section	Start Address of each ECC Parity Region Section	Density of each ECC Parity Region Section	ECC Parity Region Section memory attributes
ECC Parity Region 0 Section	0x0BFE00000	2MB	INACCESSIBLE
ECC Parity Region 1 Section	0x0BFC00000	2MB	ACCESSIBLE
ECC Parity Region 2 Section	0x0BFA00000	2MB	ACCESSIBLE

Figure 16. Configuration example

4.5.4 U-Boot configuration

The users must structure their software and U-Boot configuration carefully to ensure that the application does not access the reserved inaccessible region. If this is not implemented, a data abort happens.

The example configuration is as follows:

- Defining DRAM density
- Reserved ECC section

```
memory@80000000 {
    device_type = "memory";
    - reg = <0x00000000 0x80000000 0 0x40000000>;
    + reg = <0x00000000 0x80000000 0 0x20000000>;
    /* DRAM space - 1, size : 1 GB DRAM */
 };
    @@ -104,6 +104,13 @@
    no-map;
    reg = <0 0x90400000 0 0x1C00000>;
    };
+ /* top 128 MB reserved for ECC */
+ /*
+ ecc_reserved: ecc@0xb8000000 {
    + no-map;
    + reg = <0 0xb8000000 0 0x08000000>;
+ };
+ */
/*global autoconfigured region for contiguous allocations*/
linux,cma {
    compatible = "shared-dma-pool";
```

4.5.5 Reset vector

The i.MX 8 imposes restrictions on the start address for execution in the DDR. The startup code (usually UBOOT or ATF) must be located at the start of the DDR. Users therefore cannot locate their application code to be ECC-protected from this start of the DDR base address. Instead, the first free region after the UBOOT and ATF code can be used.

4.5.6 BOM cost

Depending on the device and ECC scheme, additional Bill of Materials (BOM) costs may be incurred.

- For the Sideband ECC, a separate DRAM device is required. Additional board-space and layout considerations must be included in the system and manufacturing cost.
- For the Inline ECC, a DRAM with a higher density (capacity) may be needed to account for the reserved region that is not accessible. This may require a higher DRAM density (than for non-ECC-based applications), increasing the system BOM cost.

NXP can help users to determine the optimal memory for their specific applications to reduce system BOM costs.

5 References

- i.MX 8M Family DDR Tools Release: https://community.nxp.com/docs/DOC-340179
- i.MX 8/8X Family DDR Tools Release: https://community.nxp.com/docs/DOC-346060
- i.MX 8 Series Reference Manuals available on www.nxp.com

6 Revision history

Table 6. Revision history

Revision number	Date	Substantive changes
0	01 April 2022	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Limited warranty and liability — 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.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer’s applications and products. Customer’s responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer’s applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX,EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org. M, M Mobileye and other Mobileye trademarks or logos appearing herein are trademarks of Mobileye Vision Technologies Ltd. in the United States, the EU and/or other jurisdictions.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 01 April 2022
Document identifier: AN13566