

# Linux Interface Specification Device Driver ECC

User's Manual: Software

RZ/G2L Group, RZ/V2L and RZ/Five

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (<http://www.renesas.com>).

## Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.

7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,  
Koto-ku, Tokyo 135-0061, Japan

[www.renesas.com](http://www.renesas.com)

## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

## Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:

[www.renesas.com/contact/](http://www.renesas.com/contact/)

## General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

### 1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

### 2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

### 3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

### 4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

### 5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

### 6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between  $V_{IL}$  (Max.) and  $V_{IH}$  (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between  $V_{IL}$  (Max.) and  $V_{IH}$  (Min.).

### 7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

### 8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

# How to Use This Manual

## 1. Purpose and Target Readers

This document is designed to provide the user with an understanding of the software development environment for RZ/G2 Group, RZ/V2L Group and RZ/Five Group processors. It is intended for users developing software incorporating the processors. A basic knowledge of software development and Linux systems is necessary in order to use this document.

Particular attention should be paid to the precautionary notes when using the manual. These notes occur within the body of the text, at the end of each section, and in the Usage Notes section.

The revision history summarizes the locations of revisions and additions. It does not list all revisions. Refer to the text of the manual for details.

The following documents apply to the RZ/G2L Group, RZ/V2L Group, and RZ/Five Group. Make sure to refer to the latest versions of these documents. The newest versions of the documents listed may be obtained from the Renesas Electronics Web site.

Document Type	Description	Document Title	Document No.
User's manual for Hardware	Hardware specifications (pin assignments, memory maps, peripheral function specifications, electrical characteristics, timing charts) and operation description  Note: Refer to the application notes for details on using peripheral functions.	RZ/G2L Group User's Manual: Hardware	---
		RZ/V2L Group User's Manual: Hardware	---
		RZ/Five Group User's Manual: Hardware	---
User's manual for Software	Description of ECC Linux Interface Specification	Linux interface Specification - ECC	This User's manual
Application Note	Information on using peripheral functions and application examples Sample programs Information on writing programs in assembly language and C	Available from Renesas Electronics Web site.	
Renesas Technical Update	Product specifications, updates on documents, etc.		

2. Notation of Numbers and Symbols

3. Register Notation

#### 4. List of Abbreviations and Acronyms

Abbreviation	Full Form
ECC	Error Correction Code
CE	Correctable Error
UE	Un-correctable Error

## Table of Contents

1. Overview.....	2
2. ECC Functions .....	3
2.1 Supported ECC Modes .....	3
2.2 Memory Layout .....	4
3. Build Instructions .....	6
3.1 Build Instructions with Yocto .....	6
3.2 Arm-trusted-firmware Build Options.....	8
3.3 U-Boot Build Configurations .....	8
4. How to Use ECC.....	9
4.1 Auto Configure Linux FDT Setting with ECC Configuration in U-boot.....	9
4.2 ECC Functions Supported by Linux Kernel .....	9
4.2.1 Support From EDAC Subsystem .....	9
4.2.2 Panic Control on Un-correctable Error .....	12
4.2.3 Application Notification .....	13
5. Appendix.....	16
5.1 List of ECC Registers.....	16

# 1. Overview

This manual explains how to use ECC on RZ/G2L Series, RZ/V2L and RZ/Five.

Note: Currently, RZ/G2L, RZ/G2LC, RZ/G2UL, RZ/V2L and RZ/Five are supported with kernel information below:

- v5.10: RZ/G2L, RZ/G2LC, RZ/G2UL, RZ/V2L and RZ/Five
- v6.1: RZ/G2L, RZ/G2LC, RZ/G2UL and RZ/V2L



## 2. ECC Functions

ECC (Error Checking and Correction) is the process of detecting bit errors in memory data. This function confirms the accuracy of the data and eliminates or at least recognizes bit errors.

The Memory Controller on the RZ/G2L series boards provides an optional error reporting and correction function that can be used to verify data in memory and correct memory errors as they occur. The memory controller will check all read transactions for data and checksum errors.

ECC works by creating unique "checksums" which are mathematical descriptions of the information in aligned data segments called "ECC Data Words". The checksum is always associated with the entire ECC data word and can be used inside the memory controller for all memory reads to control the accuracy of the data." The "online" ECC stores these checksums in memory and does not input or output them from the user interface.

The memory controller supports 64-bit ECC data word sizes. Each 64-bit memory region has an 8-bit checksum. ECC word boundaries are located on each 8-byte address (0xN0, 0xN8).

The memory controller defines ECC errors as either "correctable" or "non-correctable". A correctable error is a single bit error in the checksum or data. The controller uses syndrome to determine which bit is in error and can correct the error. Uncorrectable errors are double-bit errors in the checksum and/or data. In this case, the controller can recognize 2 bits in the checksum and/or data that are incorrect but cannot determine exactly which 2 bits are in error and therefore cannot correct the error.

ECC is enabled by programming the **ecc\_enable** parameter to a non-zero value.

### 2.1 Supported ECC Modes

ECC function in controller is controlled by a parameter named **ecc\_enable**. This parameter enables ECC and sets the reporting and correcting behavior. For details, please refer to **Table 2-1**.

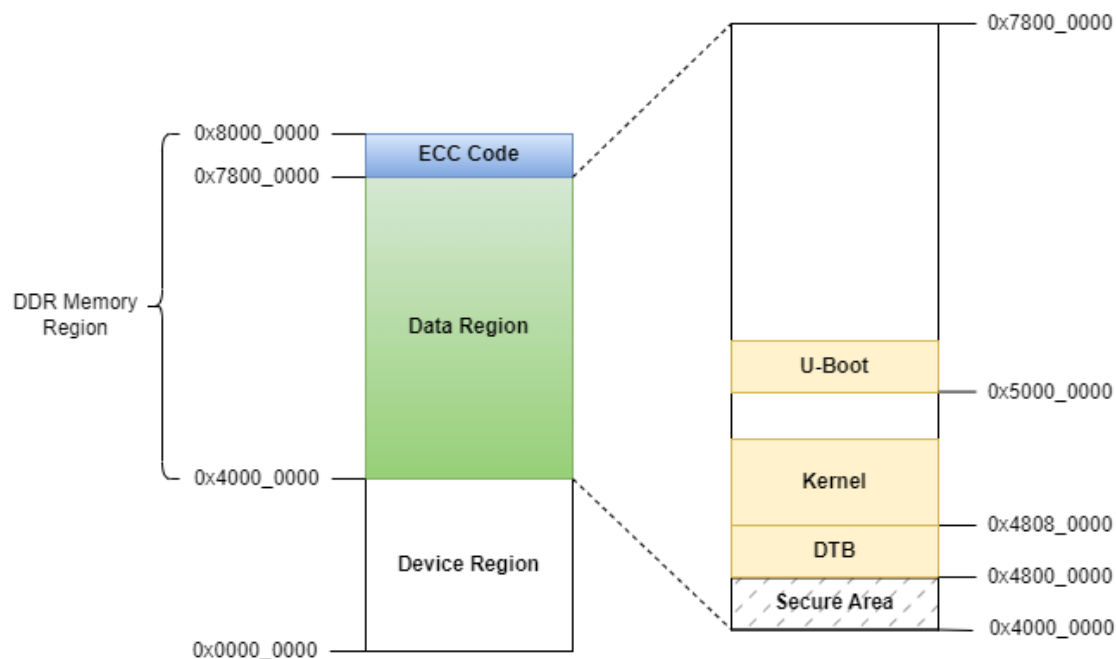
Table 2-1 Details for the meaning of parameter `ecc_enable`

<code>ecc_enable</code>	ECC Enabled	Error Detection	Error Correction	Description
`b00	No	-	-	ECC is disabled. the <code>int_status_ecc</code> parameter and the ECC report parameter will never be updated to reflect any ECC error events. Data written to memory will not contain ECC values, and data returned to the user interface will not be verified for accuracy.
`b01	Yes	No	No	ECC is enabled but not detected or corrected. The <code>int_status_ecc</code> parameter does not reflect ECC errors, and the ECC report parameter will not be updated.
`b10	Yes	Yes	No	ECC is enabled by detection, but correction is not supported. For reading commands, the <code>int_status_ecc</code> parameter and the ECC report parameter will be updated. Error data will be returned to the user on read commands and written to memory on write commands.
`b11	Yes	Yes	Yes	ECC is enabled by detection and correction. For reading commands, the <code>int_status_ecc</code> parameter and the ECC report parameter will be updated. The controller will automatically correct single-bit errors in read and write commands.

## 2.2 Memory Layout

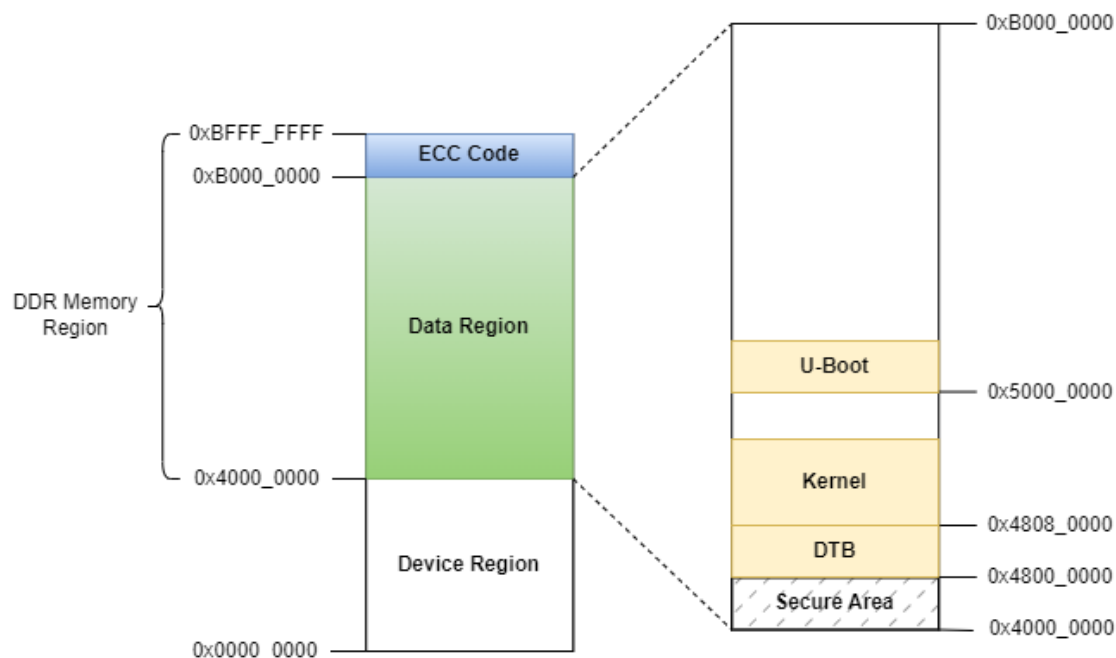
This memory controller supports in-line ECC in which a section (1/8th) of the memory device used to store data is used for ECC storage. When the ECC function was enabled memory controller will assign the top 1/8 memory region to store the ECC check code. And user is not allowed to access to this area.

As an example, when the DDR memory is 1GB, the memory layout will be as below figure when the ECC function is enabled. Assume the DDR memory map to system bus and start with address `0x4000_0000` end with `0x7FFF_FFFF`. The data region to use is from `0x4000_0000` – `0x77FF_FFFF`. The top 1/8 memory region is `0x7800_0000` – `0x7FFF_FFFF` will be used to keep ECC check code.



**Figure 2-1 Memory layout when enable ECC with 1GB DDR memory**

When the DDR memory is 2GB, the memory layout will be as below figure when the ECC function is enabled. Assume the DDR memory map to system bus and start with address 0x4000\_0000 end with 0xBFFF\_FFFF. The data region to use is from 0x4000\_0000 – 0xAFFF\_FFFF. The top 1/8 memory region is 0xB000\_0000 – 0xBFFF\_FFFF will be used to keep ECC check code.



**Figure 2-2 Memory layout when enable ECC with 2GB DDR memory**

## 3. Build Instructions

### 3.1 Build Instructions with Yocto

Following Yocto build options are used for the ECC support: they should be set in the file “local.conf”

#### (1) Enable ECC function in local.conf

To use ECC function, add setting as below in **\$WORK/build/conf/local.conf**

```
MACHINE_FEATURES_append = " ecc"
```

This code will enable the ECC support in BSP.

Note: This step is only necessary for Yocto dunfell environment.

#### (2) ECC\_MODE

There are three ECC modes:

- Enable (**ecc\_enable** = `b01)
- Error Detect (**ecc\_enable** = `b10)
- Single bit Error Correct Multibit Error Detects (**ecc\_enable** = `b11)

In case of setting the ECC mode to “Enable” (**ecc\_enable** = `b01), configure as below in **\$WORK/build/conf/local.conf**:

```
ECC_MODE = "ENABLE"
```

This setting will enable ECC but without the error detection and the error correction function.

In case of setting ECC mode to “Error Detect” (**ecc\_enable** = `b10), configure as below in **\$WORK/build/conf/local.conf**:

```
ECC_MODE = "ERR_DETECT"
```

When you set the **ECC\_MODE** = “ERR\_DETECT” and build the BSP, the below build option is applied to arm-trusted-firmware or U-Boot automatically:

- RZ/G2L, RZ/G2LC, RZ/G2UL, RZ/V2L: the option below is applied to arm-trusted-firmware.

```
DDR_ECC_DETECT=1
```

- RZ/Five: the below option is applied to U-Boot.

```
CONFIG_RZF_DDR_ECC_DETECT=y
```

In case of setting ECC mode to Single bit Error Correct Multibit Detect (**ecc\_enable** = `b11), configure as below in **\$WORK/build/conf/local.conf**:

```
ECC_MODE = "ERR_DETECT_CORRECT"
```

When you set the ECC\_MODE = "ERR\_DETECT" and build the BSP, the below build option is applied to arm-trusted-firmware or U-Boot automatically when you build:

- RZ/G2L, RZ/G2LC, RZ/G2UL, RZ/V2L: the option below is applied to arm-trusted-firmware.

```
DDR_ECC_DETECT_CORRECT=1
```

- RZ/Five: the below option is applied to U-Boot.

```
CONFIG_RZF_DDR_ECC_DETECT_CORRECT=y
```

## 3.2 Arm-trusted-firmware Build Options

For the RZ/G2L and platform, the following settings are automatically added to Arm-trusted-firmware when the Yocto build options are set.

### (1) **DDR\_ECC\_ENABLE**

This is the build option to enable ECC feature of DRAM, if not set, it is '0' as default.

### (2) **DDR\_ECC\_DETECT**

This is the build option to choose ECC error detect mode of DRAM, if not set, it is '0' as default.

### (3) **DDR\_ECC\_DETECT\_CORRECT**

This is the build option to enable single bit error correction multibit error detection. If not set, it is '0' as default.

## 3.3 U-Boot Build Configurations

For the RZ/Five platform, the ECC enable, disable, and mode selection are configured in configuration file ".config".

### (1) **CONFIG\_RZF\_DDR\_ECC**

This is the build option to enable ECC feature of DRAM.

### (2) **CONFIG\_RZF\_DDR\_ECC\_DETECT**

This is the build option to choose the ECC mode error detect.

### (3) **CONFIG\_RZF\_DDR\_ECC\_DETECT\_CORRECT**

This is the build option to choose the ECC mode single bit error correction multibit error detection.

## 4. How to Use ECC

### 4.1 Auto Configure Linux FDT Setting with ECC Configuration in U-boot

If ECC is enabled, U-boot will force update Linux memory setting in FDT (Flattened Device Tree) as below, to avoid mismatch config with ECC. The configuration is explained as below:

#### For system with 1GB DDR RAM:

- Memory node `/memory@48000000` is **re-configured**:
  - `reg = <0x0 0x48000000 0x0 0x30000000>`

#### For system with 2GB DDR RAM:

- Memory node `/memory@48000000` is **re-configured**:
  - `reg = <0x0 0x48000000 0x0 0x68000000>`

### 4.2 ECC Functions Supported by Linux Kernel

#### 4.2.1 Support From EDAC Subsystem

The ECC support in Linux kernel is from the EDAC subsystem. This system is enabled by default in the Linux kernel. When using the Linux kernel with ECC support from TF-A (RZ/G2L Series) or U-boot (RZ/Five), directory `/sys/devices/system/edac/mc/mc0/` will be created with below files in **Table 4-1**, **Table 4-2** and **Table 4-3**.

**Table 4-1** Files under `/sys/devices/system/edac/mc/mc0`

Filename	R/W/D (*)	Description
<code>ce_count</code>	R	The total count of correctable errors that have occurred on this memory controller.
<code>ce_noinfo_count</code>	R	The number of CEs that have occurred on this memory controller wherewith no information as to which DIMM slot is having errors.
<code>csrow0</code>	D	Directory for <code>csrow0</code> .
<code>max_location</code>	R	This attribute file displays the information about the last available memory slot in this memory controller. It is used by user space tools to display the memory filling layout.
<code>mc_name</code>	R	The type of memory controller that is being utilized.
<code>rank0</code>	D	Directory for <code>rank0</code> .
<code>reset_counters</code>	W	This write-only control file will zero all the statistical counters for UE and CE errors on the given memory controller.
<code>seconds_since_reset</code>	R	How many seconds have elapsed since the last counter reset. This can be used with the error counters to measure error rates.
<code>size_mb</code>	R	In count of megabytes, of memory that this memory controller manages.

ue_count	R	The total count of uncorrectable errors that have occurred on this memory controller.
ue_noinfo_count	R	The number of UEs that have occurred on this memory controller with no information as to which DIMM slot is having errors.

**Table 4-2 Files under /sys/devices/system/edac/mc/mc0/csrow0**

Filename	R/W/D (*)	Description
ce_count	R	The total count of correctable errors that have occurred on this csrow.
ch0_ce_count	R	The count of CEs on this DIMM located in channel 0.
ch0_dimm_label	R/W	This control file allows this DIMM to have a label assigned to it. With this label in the module, when errors occur the output can provide the DIMM label in the system log.
dev_type	R	What type of DRAM device is being utilized on this DIMM.
edac_mode	R	What type of Error detection and correction is being utilized.
mem_type	R	What type of memory is currently on this csrow.
size_mb	R	In count of megabytes, of memory that this csrow contains.
ue_count	R	Total count of uncorrectable errors that have occurred on this csrow.

**Table 4-3 Files under /sys/devices/system/edac/mc/mc0/rank0**

Filename	R/W/D (*)	Description
dimm_ce_count	R	Total count of correctable errors that have occurred on this DIMM.
dimm_dev_type	R	What type of DRAM device is being utilized on this DIMM
dimm_edac_mode	R	What type of Error detection and correction is being utilized.
dimm_label	R/W	With this label in the module, when errors occur the output can provide the DIMM label in the system log.
dimm_location	R	The location (csrow/channel, branch/channel/slot or channel/slot) of the dimm or rank.
dimm_mem_type	R	What type of memory is currently on this csrow.
dimm_ue_count	R	The total count of uncorrectable errors that have occurred on this DIMM.
size	R	The size of rank in MB.

(\*) R/W/D: R denotes for Read. W denotes for Write. D denotes for directory.



When the ECC error is happened, EDAC subsystem manages the error information and print it on console with below format:

```
<cnt> [CE/UE] <ctl> on <dimm> (csrow:<csrowid> page:<page> offset:<offset> grain:<grain> syndrome:<synd>)
```

Explanation of each parameter is in **Table 4-7**.

**Table 4-4 Description of each parameter in EDAC log**

Parameter Name	Data Type	Description
cnt	Decimal	Count for the CE or UE errors.
CE/UE	String	Error type. CE: Correctable Error. UE: Uncorrectable Error.
ctl	String	Name of the Memory controller.
dimm	String	The dimm where the ECC error happened.
csrowid	Decimal	ID of the csrow where the ECC error happened.
page	Hexadecimal	Indicate the page number where the ECC error happened.
offset	Hexadecimal	Indicate the offset in the page where the ECC error happened. The ECC error happened at address ((page << 12)   offset) + 0x4000_0000.
grain	Decimal	Minimum granularity for an error report, in bytes.
synd	Hexadecimal	Indicate which bit is incorrect at address ((page << 12)   offset) + 0x4000_0000. Please check <b>Table 4-5</b> to know the details.

**Table 4-5 64-Bit ECC Syndromes**

Syndrome	Incorrect Bit	Syndrome	Incorrect Bit	Syndrome	Incorrect Bit
0x00	No Error	0x4F	Data [46]	0xa4	Data [22]
0x01	Check [0]	0x52	Data [45]	0xa7	Data [21]
0x02	Check [1]	0x54	Data [44]	0xa8	Data [20]
0x04	Check [2]	0x57	Data [43]	0xab	Data [19]
0x08	Check [3]	0x58	Data [42]	0xad	Data [18]
0x0b	Data [63]	0x5b	Data [41]	0xb0	Data [17]
0x0e	Data [62]	0x5e	Data [40]	0xb5	Data [16]
0x10	Check [4]	0x62	Data [39]	0xcb	Data [15]

0x13	Data [61]	0x64	Data [38]	0xce	Data [14]
0x15	Data [60]	0x67	Data [37]	0xd3	Data [13]
0x16	Data [59]	0x68	Data [36]	0xd5	Data [12]
0x19	Data [58]	0x6b	Data [35]	0xd6	Data [11]
0x1a	Data [57]	0x6d	Data [34]	0xd9	Data [10]
0x1c	Data [56]	0x70	Data [33]	0xda	Data [9]
0x20	Check [5]	0x75	Data [32]	0xdc	Data [8]
0x23	Data [55]	0x80	Check [7]	0xe3	Data [7]
0x25	Data [54]	0x8a	Data [31]	0xe5	Data [6]
0x26	Data [53]	0x8f	Data [30]	0xe6	Data [5]
0x29	Data [52]	0x92	Data [29]	0xe9	Data [4]
0x2a	Data [51]	0x94	Data [28]	0xea	Data [3]
0x2c	Data [50]	0x97	Data [27]	0xec	Data [2]
0x31	Data [49]	0x98	Data [26]	0xf1	Data [1]
0x34	Data [48]	0x9b	Data [25]	0xf4	Data [0]
0x40	Check [6]	0x9d	Data [24]		
0x4a	Data [47]	0xa2	Data [23]		
Any Other Syndrome			Multi-Bit Error		

Example of a report for a happened ECC error.

```
EDAC MC0: 1 CE renesas,r9a07g044-edac on mc#0csrow#0 (csrow:0 page:0xc52b offset:0x680 grain:8 syndrome:0xf4)
```

This message shows there is a CE error happened, the memory controller name is “renesas,r9a07g044-edac”, the error happened in mc0csrow0. For the details, it happened in csrow 0, at address 0x4c52\_b680, bit 0 in data.

#### 4.2.2 Panic Control on Un-correctable Error

When the uncorrectable (multibit) ECC error happens, kernel will stop the process which caused the uncorrectable ECC error then show the kernel trace information.

However, some users may want the uncorrectable ECC error cause the kernel panic and freeze the system when UE detected. Please use the command below for such case.

```
$ echo 1 > /sys/module/edac_core/parameters/edac_mc_panic_on_ue
```

### 4.2.3 Application Notification

We have provided a function to notify the application when ECC error happened in DDR memory.

When the Linux kernel built with ECC enabled, it will create directory “/sys/kernel/debug/mfis\_ecc”. Under this directory, several files will be created. They are listed on a table.

**Table 4-6 Files under path /sys/kernel/debug/mfis\_ecc**

Filename	R/W	Description
pid	R/W	Process ID of the application which receive the notification from ECC driver for ECC error.
sig_id	R/W	Signal ID, when ECC error happened, ECC driver will send signal to this sig_id.
sbit_count	R/W	Counter for single bit ECC error.
mbit_count	R/W	Counter for multi bit ECC error.

For the application which wants to receive the ECC error notification, it needs to register the pid and signal id to ECC driver. Here is the example for such application:

```
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>

void ecc_error_interrupt(int n, siginfo_t *info, void *unused) {
    if (info->si_int == 1){
        printf("multi-bits error interrupt\n");
    } else {
        printf("single-bits error interrupt\n");
    }
}

int main ( int argc, char **argv )
{
    int configfd;
    int sigfd;
    int sbit_fd;
    int mbit_fd;
    int sig_id = 0;
    int inter_fd = 0;
    char buf[10];

    /* setup the signal handler for SIG_TEST
    * SA_SIGINFO -> we want the signal handler function with 3 arguments
    */
    struct sigaction sig;
    sig.sa_sigaction = ecc_error_interrupt;
    sig.sa_flags = SA_SIGINFO;
```

```

/* kernel needs to know our pid to be able to send us a signal ->
 * we use debugfs for this -> do not forget to mount the debugfs!
 */
configfd = open("/sys/kernel/debug/mfis_ecc/pid", O_WRONLY);
if(configfd < 0) {
    perror("open");
    return -1;
}

sprintf(buf, "%i", getpid());
if (write(configfd, buf, strlen(buf) + 1) < 0) {
    perror("write");
    return -1;
}

/* Get the SIGNAL_ID from rzg2l edac driver */
sigfd = open("/sys/kernel/debug/mfis_ecc/sig_id", O_RDONLY);
if(sigfd < 0) {
    perror("open");
    return -1;
}

if (read(sigfd, buf, strlen(buf)) < 0) {
    perror("read");
    return -1;
}
sscanf(buf, "%i", &sig_id);

/* Register handler for signal from rzg2l edac driver */
sigaction(sig_id, &sig, NULL);

/* Enable single-bit error interrupt */
sbit_fd = open("/sys/kernel/debug/mfis_ecc/sbit_count", O_WRONLY);
if(sbit_fd < 0) {
    perror("open");
    return -1;
}
sprintf(buf, "%i", 7);
if (write(sbit_fd, buf, strlen(buf) + 1) < 0) {
    perror("write");
    return -1;
}

/* Enable multi-bits error interrupt */
mbit_fd = open("/sys/kernel/debug/mfis_ecc/mbit_count", O_WRONLY);
if(mbit_fd < 0) {
    perror("open");
    return -1;
}

sprintf(buf, "%i", 7);
if (write(mbit_fd, buf, strlen(buf) + 1) < 0) {

```

```
        perror("write");
        return -1;
    }
    while(1);

    return 0;
}
```

## 5. Appendix

### 5.1 List of ECC Registers

Table 5-1. ECC register list

Parameter name	Address [bit filed]	Description
DRAM_CLASS	0x1141_0000 [11:8]	Defining the class of DRAM memory which is connected to the controller.
LP_AUTO_ENTRY_EN	0x1141_00f0 [3:0]	Enable auto entry into each of the low power states when the associated idle timer expires. Bit (0) controls power-down, bit (1) controls self-refresh long, bit (2) controls self-refresh long with memory and controller clock gating, and bit (3) controls self-refresh short. Set each bit to 1 to enable.
BIST_GO	0x1141_0150 [24]	Initiate a BIST operation. Set to 1 to trigger. WRITEONLY.
ADDR_SPACE	0x1141_0154 [13:8]	Sets the number of address bits to check during BIST operation.
BIST_DATA_CHECK	0x1141_0154 [16]	Enable data checking with BIST operation. Set to 1 to enable.
BIST_ADDR_CHECK	0x1141_0154 [24]	Enable address checking with BIST operation. Set to 1 to enable.
BIST_START_ADDRESS [31:30]	0x1141_0158 [31:0]	Start BIST checking at this address.
BIST_START_ADDRESS[33:32]	0x1141_015c [1:0]	Start BIST checking at this address.
BIST_TEST_MODE	0x1141_0164 [2:0]	Sets the BIST test mode. Value of 0 specifies standard BIST operation, value of 1 specifies a reduced BIST operation, value of 2 specifies a self-refresh retention test, value of 3 specifies an idle retention test, and value of 4 specifies memory initialization function. All other values are reserved.
BIST_DATA_PATTERN[31:0]	0x1141_0168 [31:0]	Data pattern to be used when the BIST_TEST_MODE parameter is programmed to 1, 2, 3 or 4. Only data corresponding to active portion of core word will be used while inactive portion will be ignored.
BIST_DATA_PATTERN [63:32]	0x1141_016c [31:0]	Data pattern to be used when the BIST_TEST_MODE parameter is programmed to 1, 2, 3 or 4. Only data corresponding to active portion of core word will be used while inactive portion will be ignored.

ECC_ENABLE	0x1141_0174 [25:24]	ECC error checking and correcting control register. Clear to 0 to fully disable ECC, program to 1 to enable ECC with no error detection or error correction, program to 2 to enable ECC with error detection without error correction, or program to 3 to enable ECC with both error detection and error correction.
ECC_DISABLE_W_UC_ERR	0x1141_0178 [16]	Controls auto-corruption of ECC when uncorrectable errors occur in R/M/W operations. Set to 1 to disable corruption.
XOR_CHECK_BITS	0x1141_0178 [15:8]	Value to xor with generated ECC codes for forced write check.
ECC_U_ADDR[31:0]	0x1141_0184 [31:0]	Address of uncorrectable ECC event. READONLY
ECC_U_ADDR[33:32]	0x1141_0188 [1:0]	Address of uncorrectable ECC event. READONLY
ECC_U_SYND	0x1141_0188 [15:8]	Syndrome for uncorrectable ECC event. READ-ONLY
ECC_U_DATA[31:0]	0x1141_018c [31:0]	Data associated with uncorrectable ECC event. READONLY
ECC_U_DATA[63:32]	0x1141_0190 [31:0]	Data associated with uncorrectable ECC event. READONLY
ECC_C_ADDR[31:0]	0x1141_0194 [31:0]	Address of correctable ECC event. READ-ONLY
ECC_C_ADDR[33:32]	0x1141_0198 [1:0]	Address of correctable ECC event. READ-ONLY
ECC_C_SYND	0x1141_0198 [15:8]	Syndrome for correctable ECC event. READ-ONLY
ECC_C_DATA[31:0]	0x1141_019c [31:0]	Data associated with correctable ECC event. READONLY
ECC_C_DATA[63:32]	0x1141_01a0 [31:0]	Data associated with correctable ECC event. READONLY
ECC_U_ID	0x1141_01a4 [17:0]	Source ID associated with the uncorrectable ECC event. READ-ONLY
ECC_C_ID	0x1141_01a8 [17:0]	Source ID associated with correctable ECC event. READ-ONLY
NON_ECC_REGION_START_ADDR_0	0x1141_01ac [13:0]	Set the base address of the soft-designated non-ECC region 0. The address written is 1Mbyte aligned. RGN=0
NON_ECC_REGION_END_ADDR_0	0x1141_01ac [29:16]	Set the ending address of the soft-designated non-ECC region 0. The address written is 1Mbyte aligned. RGN=0
NON_ECC_REGION_START_ADDR_1	0x1141_01b0	Set the base address of the soft-designated non-ECC region 1. The address written is 1Mbyte

	[13:0]	aligned. RGN=1
NON_ECC_REGION_END_ADDR_1	0x1141_01b0 [29:16]	Set the ending address of the soft-designated non-ECC region 1. The address written is 1Mbyte aligned. RGN=1
NON_ECC_REGION_ENABLE	0x1141_01b4 [1:0]	Enables each soft-designated non-ECC region. Bit (0) correlates to region 0, bit (1) correlates to region 1, etc. Set each bit to 1 to enable.
ROW_DIFF_0	0x1141_01e8 [26:24]	Difference between number of address pins available and number being used.
BANK_DIFF_0	0x1141_01e8 [9:8]	Encoded number of banks on the DRAM(s).
BANK_DIFF_1	0x1141_01e8 [17:16]	Encoded number of banks on the DRAM(s).
ROW_DIFF_1	0x1141_01ec [2:0]	Difference between number of address pins available and number being used.
COL_DIFF_0	0x1141_01ec [11:8]	Difference between number of column pins available and number being used.
COL_DIFF_1	0x1141_01ec [19:16]	Difference between number of column pins available and number being used.
CS_VAL_LOWER_0	0x1141_01f0 [15:0]	Lower bound address for chip select 0.
CS_VAL_UPPER_0	0x1141_01f0 [31:16]	Upper bound address for chip select 0.
ROW_START_VAL_0	0x1141_01f4 [2:0]	Row start value for chip select 0.
CS_VAL_LOWER_1	0x1141_01f8 [15:0]	Lower bound address for chip select 1.
CS_VAL_UPPER_1	0x1141_01f8 [31:16]	Upper bound address for chip select 1.
ROW_START_VAL_1	0x1141_01fc [2:0]	Row start value for chip select 1.
ADDR_COLLISION_MPM_DIS	0x1141_0204 [8:8]	Disable address collision detection extension using micro page mask for command queue placement and selection. Set to 1 to disable.
SWAP_EN	0x1141_020c [24:24]	Enable command swapping logic in execution unit. Set to 1 to enable. If inline ECC is enabled (if the ECC_ENABLE parameter is programmed to 1, 2 or 3), the SWAP_EN parameter must be cleared to 0 because these two features are not supported together.
DISABLE_RD_INTERLEAVE	0x1141_0210 [0:0]	Disable read data interleaving for commands from the same port, regardless of the requestor ID. If inline ECC is enabled (if the ECC_ENABLE parameter is programmed to 1, 2 or 3), the DISABLE_RD_INTERLEAVE parameter must be



		cleared to 0 because these two features are not supported together.
CS_MAP	0x1141_0210 [17:16]	Defines which chip selects are active.
IN_ORDER_ACCEPT	0x1141_0214 [16:16]	Forces the controller to accept commands in the order in which they are placed in the command queue. If inline ECC is enabled (if the ECC_ENABLE parameter is programmed to 1, 2 or 3), the IN_ORDER_ACCEPT parameter must be set to 1.
CONTROLLER_BUSY	0x1141_0218 [0:0]	Indicator that the controller is processing a command. Evaluates all ports for outstanding transactions. Value of 1 indicates controller busy. READ-ONLY
INT_STATUS_ECC	0x1141_0234 [15:0]	Status of interrupts in the controller related to ECC. READ-ONLY
INT_STATUS_BIST	0x1141_0244 [23:16]	Status of interrupts in the controller related to BIST. READ-ONLY
INT_ACK_ECC	0x1141_0254 [15:0]	Clear status of the INT_STATUS_ECC parameter. WRITE-ONLY
INT_ACK_BIST	0x1141_0264 [23:16]	Clear status of the INT_STATUS_BIST parameter. WRITE-ONLY
INT_MASK_ECC	0x1141_0274 [15:0]	Mask for the controller_int signal from the INT_MASK_ECC parameter
INT_MASK_BIST	0x1141_0284 [23:16]	Mask for the controller_int signal from the INT_MASK_BIST parameter

Revision History	Linux Interface Specification Device Driver ECC User's Manual: Software
------------------	--

Rev.	Date	Description	
		Page	Summary
1.00	Oct.31, 2023	—	First Edition issued
1.01	Dec.31, 2023	—	Add RZ/V2L information to ECC support.
1.02	May. 30, 2025	1	Add information about only supporting ECC in Linux kernel v5.10
1.03	Nov. 28, 2025	1	Add information about supporting ECC in both Linux kernel v5.10 and v6.1
		13	Correct the information in “4.2.2 Panic Control on Un-correctable Error”.

---

Linux Interface Specification Device Driver ECC  
User's Manual: Software

Publication Date: Rev.1.03 Nov. 28, 2025

Published by: Renesas Electronics Corporation

---

RZ/G2L Group, RZ/V2L and RZ/Five

