Application Note

DDR Inline ECC Error Address Decoding on Jacinto™ 7 and Sitara™ Processors

Linjun Meng, Zekun Bai, Tommy Song *Central FAE*

ABSTRACT

DDR Inline ECC is supported on Texas Instruments Jacinto™ 7 and Sitara™ processors to improve system reliability by detecting and reporting memory errors. When an ECC event occurs, the DDR controller records information in ECC error log registers. However, the logged values do not directly correspond to CPU-visible memory addresses because ECC parity data is stored within DDR memory and is managed transparently by the DDR controller. This application note explains how to interpret ECC error log registers, understand DDR Inline ECC memory organization, and translate ERR_ADR and ERR_MSK values into system and physical DDR addresses. Practical decoding examples and a companion decoding tool are also provided to simplify ECC debugging and fault analysis.

Table of Contents

List of Figures

List of Tables

Trademarks

Jacinto™Sitara™ are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction

DDR Inline ECC is commonly used in automotive, industrial, and functional safety applications to improve system robustness by detecting and correcting memory errors. When an ECC event occurs, the DDR controller logs information about the affected memory region in ECC error log registers.

Although these registers provide useful debug information, the logged values cannot be directly interpreted as memory addresses. DDR Inline ECC introduces additional ECC storage regions into DDR memory, and the memory organization differs across device families. As a result, additional decoding is required before the actual failing memory location can be identified.

This document explains:

- How ECC error log registers are used during debug
- Why ERR_ADR does not directly represent a memory address
- How DDR Inline ECC affects memory layout
- How to decode ERR_ADR and ERR_MSK values
- How to determine the corresponding system and physical DDR addresses
- How to use the companion decoding tool

2 Detailed Description

2.1 ECC Error Logging Overview

When the DDR controller detects an ECC event, the controller records information in a set of ECC error log registers. During ECC analysis, the registers shown in Table 2-1 are typically used.

Table 2-1. ECC Error Registers Used During Debug

Register	Usage
ECC_1B_ERR_CNT	Counts detected single-bit ECC errors
ECC_1B_ERR_THRSH	Defines the interrupt threshold for single-bit ECC events
ECC_1B_ERR_ADR_LOG	Identifies the ECC-protected block associated with a single-bit error
ECC_1B_ERR_MSK_LOG	Identifies the affected 8-byte segment
ECC_2B_ERR_ADR_LOG	Identifies the ECC-protected block associated with a double-bit error
ECC_2B_ERR_MSK_LOG	Identifies the affected 8-byte segment

The detailed register definitions are available in the corresponding technical reference manuals and are not repeated in this document. This application note focuses on how the reported values are decoded.

2.1.1 Understanding ERR_ADR and ERR_MSK

ERR_ADR does not directly represent the failing memory address. Instead:

- ERR_ADR identifies an ECC-protected block.
- ERR_MSK identifies the affected 8-byte segment within that block.
- Both values must be interpreted together.
- Additional decoding is required to determine the final memory address.

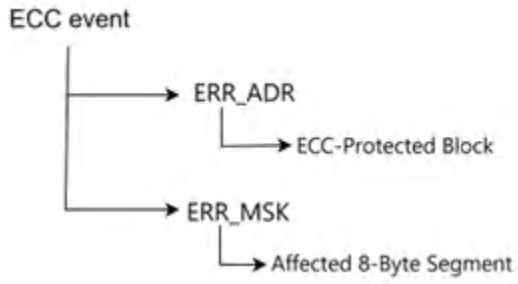

Figure 2-1. Relationship Between ERR_ADR and ERR_MSK

2.2 Why Address Decoding is Required

When DDR Inline ECC is enabled, ECC parity information is stored together with user data in DDR memory. However, the ECC storage region is invisible to software and is managed internally by the DDR controller. Consequently, CPU-visible system addresses differ from physical DDR addresses.

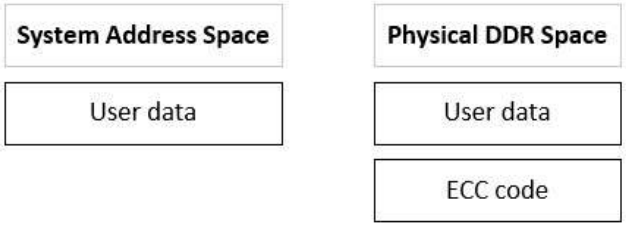

Figure 2-2. The View Between System Address Space and Physical DDR Space

As a result:

- ERR_ADR is not a direct memory address.
- Physical DDR addresses include ECC parity storage.
- Address translation must consider the DDR Inline ECC memory layout.

2.3 DDR Inline ECC Memory Layout

The DDR Inline ECC memory organization is the foundation of the address decoding process. Although ECC reporting granularity varies across processor families, all supported devices use the same principle:

- User Data
- ECC Protection Data

The primary difference between device families is the size of the user data protected by each ECC segment.

2.3.1 TDA4VM / TDA4VE / TDA4VH

Characteristics:

- DDR width: 32-bit
- VBUSM width: 512-bit
- ERR_ADR granularity: 64 bytes

Memory Layout:

Table 2-2. Memory Layout for TDA4VM / TDA4VE / TDA4VH

System Offset	DDR Content	DDR Physical Offset
0x000	512B User Data 0	0x000 - 0x1FF
Invisible	ECC for Data Block 0	0x200 - 0x23F
Invisible	ECC for Data Block 1	0x240 - 0x27F
0x200	512B User Data 1	0x280 - 0x47F

Therefore:

```
512B User Data0+ 64B ECC code0 + 64B ECC Code1 + 512B User Data1 = 1152B (0x480)
```

Each 0x400 bytes of system-visible memory occupies 0x480 bytes in physical DDR space.

2.3.2 DRA821 / DRA829 / TDA4VEN / AM62A / AM62P

Characteristics:

- DDR width: 32-bit
- VBUSM width: 256-bit
- ERR_ADR granularity: 32 bytes

Memory layout:

Table 2-3. Memory Layout for DRA821 / DRA829 / TDA4VEN / AM62A / AM62P

System Offset	DDR Content	DDR Physical Offset
0x00	256B User Data 0	0x000 - 0xFF
Invisible	ECC for Data Block 0	0x100 - 0x11F
Invisible	ECC for Data Block 1	0x120 - 0x13F
0x100	256B User Data 1	0x140 - 0x23F

Therefore:

```
256B User Data0+ 32B ECC Code0 + 32B ECC Code1 + 256B User Data1 = 576B (0x240)
```

Each 0x200 bytes of system-visible memory occupies 0x240 bytes in physical DDR space.

2.3.3 AM62 / AM62L / AM64

AM62, AM62L, and AM64 use the same physical DDR Inline ECC memory organization as the devices described in Section 2.3.2. However, the ECC error reporting granularity differs because these devices use a 16-bit DDR interface. As a result:

- ERR_ADR granularity = 16 bytes
- ERR_MSK width = 4 bits

while the physical DDR layout remains unchanged.

2.4 Address Decoding Methodology

The ERR_ADR register does not contain a direct system address. Instead, the register records the index of an ECC-protected memory block. The address granularity depends on the processor family:

- AM62 / AM62L / AM64: 16 bytes
- AM62A / AM62P / DRA821 / DRA829 / TDA4VEN: 32 bytes
- TDA4VM / TDA4VE / TDA4VH: 64 bytes

The ECC block start address is derived from ERR_ADR, while ERR_MSK identifies the affected 8-byte segment within the reporting range.

To locate the affected memory location, the following steps are required:

1. Determine the ECC block represented by ERR_ADR.
2. Use ERR_MSK to identify the affected 8-byte segment.
3. Convert the resulting offset into a system address.
4. Apply the ECC memory layout to determine the physical DDR address.

2.5 AM62A Decoding Example

Assume:

ERR_ADR = 0x101

ERR_MSK = 0x10

AM62A uses:

- 32-byte ERR_ADR granularity
- 8-bit ERR_MSK
- 256-bit VBUSM width

ECC Block Start Offset:

0x101 × 0x20 = 0x2020

Because ERR_ADR is odd, the corresponding ECC block shares the same ERR_MSK field with the adjacent even block. Therefore, the offset must be adjusted according to the parity of ERR_ADR.

Bit 4 of ERR_MSK is asserted:

System Address:

= 0x2020 - 0x20 + (4 × 8) + 0x80000000

= 0x80002020

Every 16 ERR_ADR entries correspond to one 0x240-byte physical DDR segment (including ECC storage). Therefore, the block count is obtained from ERR_ADR / 16.

Physical Address:

Block count = 0x101 / 0x10 = 0x10

Offset = 0x101 % 0x10 = 1 for block in which it is located

Locations 0x10 × 0x240 + 0x20 + 0x80000000 = 0x80002420 within DDR

2.6 TDA4VE Dual-DDR Interleave Example

TDA4VE Configuration:

TDA4VE uses two DDR controllers operating in interleaved mode. Each DDR controller maintains an independent ECC error log register set. Therefore, the DDR controller reporting the ECC event determines which channel contains the failing location.

- Dual DDR controllers
- 32-bit DDR interface width
- 512-bit VBUSM width
- ERR_ADR granularity = 64 bytes
- ERR_MSK width = 16 bits
- Default DDR interleave granularity = 128 bytes

In dual-DDR interleaved mode, consecutive 128-byte memory regions are alternately mapped to DDR0 and DDR1. For TDA4VE dual-DDR interleaved mode, every 16 ERR_ADR entries form one repeating address pattern. Table 2-4 shows the logical and physical offsets corresponding to ERR_ADR mod 16 as shown below:

Table 2-4. ADR Mapping Table for Logical Offset and Physical Offset

MOD(ADR,16)	DDR0 Logical Offset	DDR1 Logical Offset	DDR0 Physical Offset	DDR1 Physical Offset
0	0x0	0x80	0x0	0x80
1	0x40	0xC0	0x40	0xC0
2	0x100	0x180	0x100	0x180
3	0x140	0x1C0	0x140	0x1C0
4	0x200	0x280	0x200	0x280
5	0x240	0x2C0	0x240	0x2C0
6	0x300	0x380	0x300	0x380
7	0x340	0x3C0	0x340	0x3C0
Hamming Code	Invisible	Invisible	0x400	0x480
Hamming Code	Invisible	Invisible	0x440	0x4C0
8	0x400	0x480	0x500	0x580
9	0x440	0x4C0	0x540	0x5C0
10	0x500	0x580	0x600	0x680
11	0x540	0x5C0	0x640	0x6C0
12	0x600	0x680	0x700	0x780
13	0x640	0x6C0	0x740	0x7C0
14	0x700	0x780	0x800	0x880
15	0x740	0x7C0	0x840	0x8C0

Assume:

DDR0_1B_ERR_ADR = 0x49

DDR0_1B_ERR_MSK = 0x0400

Decode the System Address

Address granularity = 64 bytes

The system is configured in dual-DDR interleaved mode with an interleave granularity of 128-byte.

For every 16 ERR_ADR entries:

Logical address space occupied:

16 × 128B = 0x800B

Physical DDR space occupied:

16 × 128B + ECC storage

= 0x900B

The ECC block start offset is:

floor (0x49 / 16) × 0x800

0x4 × 0x800

= 0x2000

The ECC block segment offset is:

0x49 mod 16 = 9

From the address mapping table, the DDR0 logical offset corresponding to ERR_ADR mod 16 = 9 is 0x440.

Bit 10 in ERR_MSK is set:

Error Offset = 10 × 8 = 0x50

Since ERR_ADR is odd, the corresponding ECC block shares the same ERR_MSK field with the previous address block. Therefore:

System Offset

= 0x440 - 0x40 + 0x50 + 0x2000

= 0x2450

System Address

= System Offset + 0x80000000

= 0x80002450

Physical Address Calculation:

The ECC block start off set is:

floor (0x49 / 16) × 0x900

= 0x4 × 0x900

= 0x2400

The ECC block segment offset is:

0x49 mod 16 = 9

From the address mapping table, the DDR0 physical offset corresponding to ERR_ADR mod 16 = 9 is 0x540.

Bit 10 in ERR_MSK is set:

Error Offset = 10 × 8 = 0x50

Since ERR_ADR is odd, the corresponding ECC block shares the same ERR_MSK field with the previous address block. Therefore:

Physical Offset

= 0x540 - 0x40 + 0x50 + 0x2400

= 0x2950

Physical address

= Physical Offset + 0x80000000

= 0x80002950

2.7 Decode Tool Usage

The spreadsheet tool TI TDA4x Inline ECC Error Register Translator automates the address decoding process by converting ERR_ADR and ERR_MSK values into corresponding system and physical DDR addresses. The tool also highlights the affected 8-byte region associated with the ECC Error register.

3 Summary

DDR Inline ECC provides an effective mechanism for detecting memory corruption events on Jacinto 7 and Sitara processors. ECC error log registers record information about the affected ECC-protected memory block, but the logged values do not directly correspond to system memory addresses. By understanding ECC memory organization and applying the decoding methodology described in this document, users can accurately identify the memory location associated with an ECC event. The accompanying decode tool further simplifies debug and fault analysis.

4 References

1. Texas Instruments, J721E DRA829/TDA4VM Processors Technical Reference Manual, technical reference manual.
2. Texas Instruments, J721S2/TDA4VE/AM68A Technical Reference Manual, technical reference manual.
3. Texas Instruments, J784S4/TDA4VH Technical Reference Manual, technical reference manual.
4. Texas Instruments, J7200 DRA821 Technical Reference Manual, technical reference manual.
5. Texas Instruments, AM62Ax Sitara™ Processors Technical Reference Manual, technical reference manual.
6. Texas Instruments, AM62Px Sitara™ Processors Technical Reference Manual, technical reference manual.
7. Texas Instruments, AM62x Sitara™ Processors Technical Reference Manual, technical reference manual.
8. Texas Instruments, AM64x/AM243x Technical Reference Manual, technical reference manual.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you fully indemnify TI and its representatives against any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale, TI’s General Quality Guidelines, or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025