

# **Error Detection and Correction Mechanism of TMS320C64x+ and TMS320C674x**

Frank Noha/Chad Courtney

## **ABSTRACT**

This application report describes the error detection and correction mechanism of the C64x+/C674x megamodule L1P and L2 memories implemented on some devices. Depending on the type of application, these mechanisms are used to either provide diagnostic measures to detect faults in the memory that could lead to unacceptable risk for the user or to increase the availability of the system.

## **Contents**

1	Introduction .....	2
2	L1P Error Detection .....	2
3	L2 Error Detection and Correction .....	6

## **List of Figures**

1	Error Detection Logic .....	2
2	L1P Error Detection Status Register (L1PEDSTAT) .....	3
3	L1P Error Detection Command Register (L1PEDCMD) .....	4
4	Error Detection Address Register (L1PEDADDR) .....	5
5	L2 Error Detection Status Register (L2EDSTAT) .....	8
6	L2 Error Detection Command Register (L2EDCMD) .....	9
7	L2 Error Detection Address Register (L2EDADDR) .....	10
8	L2 Error Detection Page Enable Registers (L2EDPEN0) .....	10
9	L2 Error Detection Page Enable Registers (L2EDPEN1) .....	10
10	L2 Error Detection Event Counter Registers (L2EDPEC) .....	11
11	L2 Error Detection Event Counter Registers (L2EDNPEC) .....	11

## **List of Tables**

1	L1P Register Summary Table .....	3
2	L1P Error Detection Status Register (L1PEDSTAT) Field Descriptions .....	3
3	L1P Error Detection Command Register (L1PEDCMD) Field Descriptions .....	4
4	Error Detection Address Register (L1PEDADDR) Field Descriptions .....	5
5	L2 Register Summary Table .....	8
6	L2 Error Detection Status Register (L2EDSTAT) Field Descriptions .....	8
7	L2 Error Detection Command Register (L2EDCMD) Field Descriptions .....	9
8	L2 Error Detection Status Register (L2EDSTAT) Field Descriptions .....	10
9	L2 Error Detection Page Enable Registers (L2EDPEN0) Field Descriptions .....	11
10	L2 Error Detection Page Enable Registers (L2EDPEN1) Field Descriptions .....	11

## 1 Introduction

Many applications have very stringent requirements to detect faults in the memory system of a processor to avoid failures of the end-system that could lead to dangerous situations for the end-user, or high requirements regarding the availability of the end-system. There are many mechanisms that could lead to faults in the memory of a processor. Some of these could lead to permanent faults and others to transient faults. Detection of transient faults while the system is running is very important for critical applications. While permanent faults can be equally important, the likelihood of them occurring is usually significantly lower than transient faults. Permanent faults can, in many cases, be detected by running appropriate test algorithms at application startup or shutdown. Transient faults are faults predominantly introduced by soft errors. Major contributors of these are alpha radiation of the materials used in the package of a chip or neutron particles from cosmic rays. These can lead to bit flips in memories or changes to the state of a flip-flop.

This application report describes the mechanisms implemented on the C64x+/C674x memory system to detect faults in the memory.

---

**NOTE:** The reader of this application report should check that these mechanisms are implemented on the specific device they are using, as these are optional features that are not implemented on all C64x+/C674x megamodule versions.

---

Errors can be detected by checking the single bit parity, which is simply the XOR-reduce of 256-bit wide read accesses to L1P/L2 RAM or L1P/L2 cache. This computed parity is checked against the separately stored parity. The parity bit and valid bit (parity bit qualifier) is stored in the parity RAM for each memory write. Generated parity errors can be corrected by error detection and correction logic, and an interrupt and exception is provided for extra processing by the system code.

## 2 L1P Error Detection

The L1P supports simple error detection on program fetches and DMA and IDMA accesses. When error detection is enabled, all 256-bit wide accesses on a 256-bit boundary are parity checked against a separately stored parity. The following sections define how the parity check scheme operates.

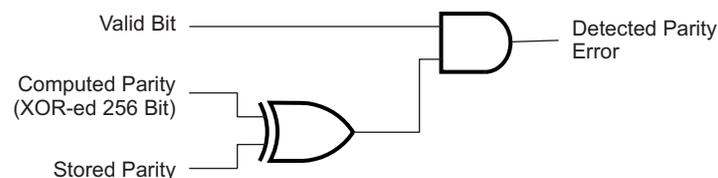
### 2.1 Parity Generation

A single parity is generated for each 256-bit word. For each write to the L1P memory, a parity bit and a valid bit is stored. For writes narrower than 256-bit (DMA and IDMA accesses <256-bit), the parity will be marked invalid.

### 2.2 Error Detection Mechanism

The L1P checks parity on each 256-bit wide, 256-bit aligned access to L1P RAM and L1P cache. This includes program fetches and DMA/IDMA reads. (It is worth noting that all program fetches will be checked, since they are always 256 bits wide and 256-bit aligned, but not necessarily all DMA/IDMA accesses.)

On each such read, L1P calculates a single bit parity which is simply the XOR-reduce of the 256 bits of data. It also reads the stored parity and valid bits. The following diagram illustrates how the computed parity, stored parity and valid bits affect the outcome of the parity check.



**Figure 1. Error Detection Logic**

When the L1P detects a parity error, it asserts one of two parity error event signals. In the case of a parity error on a program fetch, the L1P asserts the exception directly to the CPU. In the case of a parity error on a remote access (DMA and IDMA), the L1P asserts the error to the interrupt and exception controller.

### 2.3 L1P Error Detection Control and Status Registers

The L1P implements three memory-mapped registers (listed in [Table 1](#)) to allow software to control the error detection logic, as well as to allow software to handle errors when they are detected.

**Table 1. L1P Register Summary Table**

Name	Address	Description
L1PEDSTAT	0x01846404	L1P Error Detection Status Register
L1PEDCMD	0x01846408	L1P Error Detection Command Register
L1PEDADDR	0x0184640C	L1P Error Detection Address Register

Programs write to the L1PEDCMD command register to change the state of the error detection logic and to clear any reported errors. Programs read the L1PEDSTAT and L1PEDADDR registers to determine the current status of the L1P error detection logic.

### 2.4 L1P Error Detection Status Register (L1PEDSTAT)

The L1PEDSTAT register has five read-only single-bit fields in it. Three of these fields indicate the current operating mode of the L1P error detect logic. The remaining two fields indicate whether a program fetch or DMA access resulted in a parity-check error. The L1P error detection status register is shown in [Figure 2](#) and described in [Table 2](#).

**Figure 2. L1P Error Detection Status Register (L1PEDSTAT)**

31	Reserved								16
R, +0									
15	7	6	5	4	3	2	1	0	
Reserved		DMAERR	PERR	RSVD	SUSP	DIS	RSVD	EN	
R, +0		R, +0	R, +0	R, +0	R, +0	R, +1	R, +0	R, +0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 2. L1P Error Detection Status Register (L1PEDSTAT) Field Descriptions**

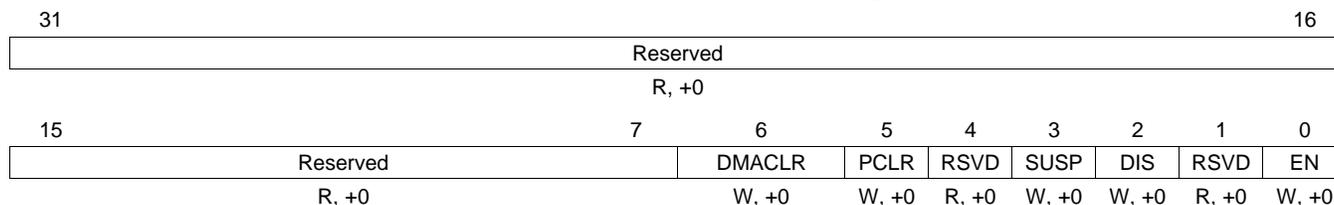
Bit	Field	Meaning When Set
31-7	Reserved	Reserved
6	DMAERR	A DMA access to L1P RAM resulted in a parity check error. L1PEDADDR contains the address of the affected 256-bit word.
5	PERR	A program fetch resulted in a parity check error. L1PEDADDR contains the address of the affected fetch packet.
4	Reserved	Reserved
3	SUSP	Error Detection logic is suspended. Writes do not modify parity or valid bits. Reads do not check parity.
2	DIS	Error detection logic is disabled. Writes clear 'valid'. Reads do not check parity.
1	Reserved	Reserved
0	EN	Error detection logic is enabled. Writes update parity/valid. Reads check parity.

In this register, exactly one of EN, DIS, or SUSP can be set at any given time. Similarly, only one of PERR or DMAERR can be set at a given time. The CPU cannot modify the state of these bits by accessing L1PEDSTAT directly. Rather, the state of these bits is controlled by writing to L1PEDCMD, described in [Section 2.5](#).

## 2.5 L1P Error Detection Command Register (L1PEDCMD)

Writing to the L1PEDCMD register controls the action of the L1P error detection logic. Writing a 0 to a given bit within this register has no effect. Writing a 1 to a given bit initiates the action associated bit. The L1P error detection command register is shown in [Figure 3](#) and described in [Table 3](#).

**Figure 3. L1P Error Detection Command Register (L1PEDCMD)**



LEGEND: R/W = Read/Write; W = Write; R = Read only; -n = value after reset

**Table 3. L1P Error Detection Command Register (L1PEDCMD) Field Descriptions**

Bit	Field	Meaning When Set
31-7	Reserved	Reserved
6	DMACLR	Clears the DMA-read parity error status
5	PCLR	Clears the program-fetch parity error status
4	Reserved	Reserved
3	SUSP	Suspends the error detection logic
2	DIS	Disables the error detection logic
1	Reserved	Reserved
0	EN	Enables the error detection logic

Note that programs must not write a 1 to more than one of EN, DIS, and SUSP simultaneously. The error detect logic transitions to the disabled state if it sees more than one of these bits written with 1 simultaneously. Programs can write a 1 to both PCLR and DMACLR in parallel to blindly clear PERR and DMAERR. Furthermore, they can do so in parallel to writing a 1 to one of EN, DIS, and SUSP.

## 2.6 L1P Error Detection Address Register (L1PEDADDR)

The error detection address register contains the address of the detected error. The register only holds the upper 27 bits of the error address, as only errors on a 256-bit boundary are detected. The L1P error detection address register is shown in Figure 4 and described in Table 4.

**Figure 4. Error Detection Address Register (L1PEDADDR)**

31	ADDR	5	4	1	0
R, +0		Reserved		RAM	
		R, +0		R, +0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 4. Error Detection Address Register (L1PEDADDR) Field Descriptions**

Bit	Field	Value	Meaning When Set
31-5	ADDR		Address of parity error
4-1	Reserved	0	Reserved
0	RAM	0	Error detected in L1P cache
		1	Error detected in L1P RAM

## 2.7 L1P Error Detection Logic Operation

The error detect logic defaults to the disabled state after reset. While in the disabled state, the error detect logic clears the valid bits associated with each parity bit whenever it sees a write within L1P. The error detect logic does not check parity on reads while in this state.

Programs enable the error detect logic by writing a 1 to L1PEDCMD.EN bit. The error detect logic in L1P does not initialize the parity RAM when transitioning from the disabled to the enabled state. Rather, it relies on the valid-clearing behavior that occurs during the disabled state in order to ensure that the parity RAM is cleared. Therefore, upon entering the enabled state, there may be invalid parity values in the parity RAM whose corresponding valid bits are also set. To avoid false parity errors for program code executing from L1P RAM, programs should write to all L1P RAM addresses prior to executing code from L1P RAM. Programs executing from L1P cache do not require additional consideration. While the error detect logic is enabled, all 256-bit writes will update the stored parity and valid bits. Writes narrower than 256 bits will update the parity RAM to indicate 'invalid parity'. All 256-bit reads will be parity checked.

Programs can suspend the error detect logic by writing a 1 to L1PEDCMD.SUSP. While suspended, the L1P neither checks parity nor updates the valid bits. The purpose of this mode is to allow testing of the error detect logic. Programs can enable error detection, initialize the parity RAM with a set of valid parity entries (writing a program image into the RAM via DMA) and then suspend the logic. While suspended, the program can then manually introduce parity errors in the memory, without invalidating the corresponding parity entries. Finally, the program can re-enable error detection and verify that the hardware correctly detects the error.

Programs can disable the error detect logic at any time by writing a 1 to L1PEDCMD.DIS. While in this mode, the L1P clears the 'valid' bits on each parity entry whenever it sees a write within L1P. It does not do an explicit block-clear of L1P's parity memory.

## 2.8 Error Exceptions

The L1P provides two error detection exception outputs. These outputs signal that L1P detected a parity error by triggering an interrupt or exception. L1P asserts one of these outputs directly to the CPU, and the other to the interrupt and exception controller.

### 2.8.1 Parity Errors Due to Program Fetch

The first of these signals indicates that a program fetch resulted in a parity error. L1P asserts this signal directly to the CPU. This result is known 1 to 2 cycles after the CPU has fetched the data. When asserted, the CPU takes an exception immediately.

The L1P asserts the parity error to the CPU and the CPU responds sufficiently quickly that no instructions from the corrupted fetch packet are executed. Programs must consult the L1PEDADDR register to determine where the actual parity error occurred.

### 2.8.2 Parity Errors Due to DMA/IDMA Activity

The second of these signals indicates that a DMA and IDMA access resulted in a parity error. L1P asserts this signal to the interrupt and exception controller, where it can be selected by software as an interrupt (or optionally, exception) input (default event number 113: L1P\_ED) to the CPU.

### 2.9 Parity Error Event Recording

The L1P can precisely record one error-detection event at a time until software clears the L1P's ED status registers. It records the first error detection event that it sees. That is, once a parity error is reported, it sets DMAERR or PERR as appropriate, records the address, and signals no further errors. There is one important exception to this rule: if a program fetch results in a parity error when there is already a DMA error recorded, the program fetch results in an exception, with the details of the program fetch error replacing the details of the DMA error.

### 2.10 Emulation Behavior

Emulation reads in L1P are never checked for parity, because they are narrower than 256 bits. Emulation writes clear the 'valid' bit for the parity (again, because they are narrower than 256 bits), except when the error detection logic is suspended. When debugging an application, you can set and remove breakpoints. The debugger sets and clears software breakpoints with emulation writes to memory. As noted above, these emulation writes invalidate the parity associated with the fetch packet that they reside in. For fetch packets held in L1P, valid parity can be re-established by re-DMAing into L1P RAM, or by invalidating the affected lines in L1P cache to force a re-fetch.

## 3 L2 Error Detection and Correction

The L2 memory provides support for error detection and correction (EDC). It performs full EDC on all program fetches from the L1P cache and on 256-bit wide DMA accesses. The EDC block only provides parity-check based single-bit error detection on data fetches from the L1D cache. Furthermore, the error detection is performed after the data is returned to L1D; thereby removing the error detection delay from the cost of L1D cache misses.

The EDC implements a distance-3 "detect 2, correct 1" Hamming code. 10 parity bits for each 256 data bits are implemented. The L2 always performs a full Hamming Code check on 256-bit reads originating from L1P, IDMA or DMA. It only performs a 256-bit parity check for 256-bit fetches from L1D.

Programs can suspend the error detection logic by writing a one to the L2 Error Detection Command Registers Suspend bit (L2EDCMD.SUSP = '1'). While suspended, the L2 neither checks parity nor updates the valid bits.

### 3.1 Parity Generation

The L2 updates the stored parity information in response to every 256-bit write on a 256-bit boundary. These writes can arrive from IDMA, DMA, or L1D (as victim write-backs). If the write is for 256 bits on a 256-bit boundary, the L2 calculates the new parity and writes it to the parity RAM, along with a valid bit set to 1. If the write is for less than 256 bits or is not 256-bit aligned, the L2 sets the valid bit to 0.

### 3.2 Memory Scrubbing Techniques

The previous section illustrates that the L2 calculates parity bits and marks them "Valid" only under certain conditions. It follows L2 does not generate valid parity information in some cases. Specifically, the L2 clears the valid bit whenever DMA, IDMA or L1D write less than 256 bits to L2 memory.

Because L2 does not compute incremental parity for smaller-than-256-bit writes, programs may find it necessary to 'scrub' memory periodically in order to ensure that valid parity is set for all address ranges of interest. This can be achieved within L2 RAM by using the IDMA to read and write a range of L2 RAM addresses. This operation can also be used to correct single-bit parity errors that occur during normal program operation on 256-bit words that already have valid parity.

Specifically, to scrub a range of memory, the program initiates IDMA transfers with the source and destination addresses equal to each other, and the byte count set to cover the desired block. The address range must be 256-bit aligned and a multiple of 256 bits for the entire range to be scrubbed. As the IDMA reads the block of memory from L2, the EDC hardware corrects any single-bit errors that might be present on 256-bit words that do have valid parity. When the IDMA writes the data back to L2, the EDC generates parity for the write and marks it valid. In the case where a 2-bit error is detected, the resulting data for the 256-bit block of data will return all zero's. Note that the resulting exception will indicate a 2-bit error and appropriate action by SW needs to be taken.

Note that L2 addresses held within L1D potentially result in snoop-read and snoop-write requests to L1D. The IDMA reads will read updated data from L1D only if L1D is marked as 'dirty'. The IDMA writes will update both L1D and L2 if L1D holds a copy of the data, regardless of whether it is dirty. This snoop behavior is not a correctness issue with respect to the 'scrubbing' technique as long as the CPU is not writing to the address range being scrubbed as the scrubbing-IDMA is in progress. Programs that wish to avoid stalls due to these IDMA related snoops can do so by manually writing back and invalidating the range being scrubbed from L1D prior to scrubbing.

The IDMA technique described above is assured to work for L2 RAM addresses that are not configured as cache. IDMA scrubbing under cache works under the following conditions:

- No CPU-initiated write accesses to L2 cache. This includes:
  - L1D write misses
  - L1D victim write-backs
  - L2 line allocations due to L2 read misses
- No L1D or L2 cache control operations (such as write-backs, invalidates, size-change) in progress.

It is recommended that the programmer places L1D and L2 caches in freeze mode while the RAM under L2 cache is being scrubbed. Furthermore, the programmer should avoid accessing L2 cache while the scrubbing IDMA is active, although read requests that hit the L2 cache do not negatively impact the process. (By allowing read requests to L2 cache during the process, the scrubbing code itself may be held in L2 cache, provided L2 is frozen as recommended above, in order to prevent unwanted line allocations.)

To fully scrub L2 cache without knowing its contents, one should initiate a global write-back-invalidate of the entire L2. As updated lines are written back from L2, any lines for which valid parity exists will be corrected prior to write-out. Later, when the data is brought back in by normal program activity, the EDC hardware generates valid parity as the lines are allocated in L2.

Programs can regenerate parity for ranges of addresses that might be cached in with software running on the CPU. To do so, programs should perform the following steps on that range:

1. Read a single word. This causes 64 consecutive bytes to be cached into L1D.
2. Write the word back to the same address. This causes the L1D cache line to be marked dirty without actually changing its contents.
3. Repeat steps 1 and 2 at 64-byte intervals. This causes the entire address range to move through L1D cache, while operating on the minimum amount of data.
4. Perform an L1D block write-back or L1D block write-back invalidate for the address range. This forces the data in this range that is held in L1D to be written to L2, thereby generating new parity.

This technique is suitable for small to medium ranges of addresses, such as a CPU stack, collections of data structures in a heap, and so on. While this technique is suitable when no bit errors currently exist or the range in question lacks valid parity, it is insufficient for scrubbing memory that already has valid parity and may have bit errors that need correcting.

To manually scrub a range of addresses in L2 cache and correct bit errors, one must employ the technique above along with an appropriate interrupt or exception handler to catch and correct single-bit errors as they are detected. Bit errors can only be corrected by forcing L2 to write-back-invalidate the affected line in L2. The line in L1D must be invalidated without write-back.

The loop that processes the range can be modeled upon the sequence given above. However, it must be written to account for the fact that a bit error might occur between the read in step #1 and the write in step #2. This can be achieved by doing the following: Reading the data twice (that is, repeat step 1), and ensuring that there is sufficient distance between the reads such that an interrupt triggered by the first read can be taken prior to the second read being issued.

It is necessary to note that this software-oriented scrubbing process must not pipeline L1D read misses to L2. The EDC hardware only captures the error context for the first miss. Code that pipelines missed would need to “rewind” and re-read if multiple errors were detected during a sequence of pipelined read misses.

### 3.3 L2 Error Detection Control and Status Registers

The L2 contains seven registers dedicated to the error detection logic, shown in Table 5. These registers are memory mapped, and located along with the other L2 control registers.

**Table 5. L2 Register Summary Table**

Name	Address	Description
L2EDSTAT	0x01846004	L2 Error Detection Status Register
L2EDCMD	0x01846008	L2 Error Detection Command Register
L2EDADDR	0x0184600C	L2 Error Detection Address Register
L2EDPEN0	0x01846010	L2 Error Detection Page Enable 0 Register
L2EDPEN1	0x01846014	L2 Error Detection Page Enable 1 Register
L2EDCPEC	0x01846018	L2 Correctable Error Count Register
L2EDNPEC	0x0184601C	L2 Uncorrectable Error Count Register

### 3.4 L2 Error Detection Status Register (L2EDSTAT)

The L2 error detection status register (L2EDSTAT) provides the status of the error detection logic. It contains fields to indicate an L1P access error (PERR), an L1D access (DERR), the error detection and parity generation logic state (SUSP, DIS, EN). The L2 error detection status register is shown in Figure 5 and described in Table 6.

**Figure 5. L2 Error Detection Status Register (L2EDSTAT)**

31							24	23					16	
Reserved							BITPOS							
R, +0							R, +0							
15				10	9	8	7	6	5	4	3	2	1	0
RSV			NEER		RSV	DMAERR	PEER	DEER	SUSP	DIS	RSV	EN		
R, +0			R, +0		R, +0	R, +0	R, +0	R, +0	R, +0	R, +0	R, +0	R, +1	R, +0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 6. L2 Error Detection Status Register (L2EDSTAT) Field Descriptions**

Bit	Field	Value	Description
31-24	Reserved	0	Reserved
23-16	BITPOS	00000000	Single Bit error in position 0
		00000001	Single Bit error in position 1
		...	...
		11111110	Single Bit error in position 254
		11111111	Single Bit error in position 255
15-10	Reserved	0	Reserved

**Table 6. L2 Error Detection Status Register (L2EDSTAT) Field Descriptions (continued)**

Bit	Field	Value	Description
9-8	NEER	00	Single Bit error
		01	Double Bit error
		10	Reserved
		11	Error in the parity value, 256-bit data is correct.
7	Reserved	0	Reserved
6	DMAERR	0	No parity error occurred during DMA access
		1	Parity error occurred during DMA access
5	PEER	0	No parity error occurred during L1P access
		1	Parity error occurred during L1P access
4	DEER	0	No parity error occurred during L1D data access
		1	Parity error occurred during L1D data access
3	SUSP	0	Error detection/parity generation logic is not suspended
		1	Error detection/parity generation logic is suspended
2	DIS	0	Error detection/parity generation logic is not disabled
		1	Error detection/parity generation logic is disabled
1	Reserved	0	Reserved
0	EN	0	Error detection/parity generation logic is not enabled
		1	Error detection/parity generation logic is enabled

### 3.5 L2 Error Detection Command Register (L2EDCMD)

The L2 error detection command register (L2EDCMD) lets the software clear the error reported in the L2EDSTAT register as well as enable, disable or suspend the error detection and parity generation logic. The L2 error detection command register is shown in [Figure 6](#) and described in [Table 7](#).

**Figure 6. L2 Error Detection Command Register (L2EDCMD)**

31	Reserved								16		
R, +0											
15	Reserved			7	6	5	4	3	2	1	0
			DMACLR	PCLR	DCLR	SUSP	DIS	RSVD	EN		
R, +0			W, +0	W, +0	W, +0	W, +0	W, +0	W, +0	R, +0	W, +0	

LEGEND: R/W = Read/Write; W = Write; R = Read only; -n = value after reset

**Table 7. L2 Error Detection Command Register (L2EDCMD) Field Descriptions**

Bit	Field	Meaning When Set
31-7	Reserved	Reserved
6	DMACLR	DMA read parity error bit cleared and address reset in L2EDSTAT
5	PCLR	Program fetch parity error bit cleared in L2EDSTAT and L2EDADDR cleared
4	DCLR	Data fetch parity error bit cleared in L2EDSTAT and L2EDADDR cleared
3	SUSP	Suspends error detection and parity generation logic
2	DIS	Disables error detection and parity generation logic
1	Reserved	Reserved
0	EN	Enables error detection and parity generation logic

Software can clear multiple error status bits by writing 1 to more than one of L2EDCMD.DCLR, L2EDCMD.PCLR and L2EDCMD.DMACLR at the same time. Programs must not write 1 to more than one of L2EDCMD.SUSP, DIS, and EN in a single write. In the event that an incorrect program does write 1 to more than one of these bits in a single write, L2 treats the write identically to writing 1 to L2EDCMD.DIS, and transitions the EDC logic to the disabled state.

### 3.6 L2 Error Detection Address Register (L2EDADDR)

The L2 error detection address register (L2EDADDR) provides the address information for the detected error. It contains fields to indicate address (ADDR), the way of cache within the L2 memory (L2WAY), and the error was encountered in L2 cache or RAM section (RAM). The L2 error detection address register is shown in Figure 7 and described in Table 8.

**Figure 7. L2 Error Detection Address Register (L2EDADDR)**

31	ADDR	5	4	3	2	1	0
				L2WAY	RSV	RAM	
R, +000 0000 0000 0000 0000 0000				R, +00	R, +00	R, +0	

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 8. L2 Error Detection Status Register (L2EDSTAT) Field Descriptions**

Bit	Field	Value	Description
31-5	ADDR		Address of parity error (5 LSBs assumed to be 000000b).
4-3	L2WAY	00	Error detected in way 0 of L2 cache
		01	Error detected in way 1 of L2 cache
		10	Error detected in way 2 of L2 cache
		11	Error detected in way 3 of L2 cache
			Note: L2WAY = 00 when RAM = 1
2-1	Reserved	0	Reserved
0	RAM	0	Error detected in L2 cache
		1	Error detected in L2 RAM

### 3.7 L2 Error Detection Page Enable Registers (L2EDPEN0, L2EDPEN1)

Apart from the global enable, disable and suspended logic of L2 data, it provides the ability to individually select or deselect L2 pages by programming the error detection page enable registers. L2 memory may be divided into multiple regions. The L2 error detection page enable registers are shown in Figure 8 and Figure 9 and described in Table 9 and Table 10.

**Figure 8. L2 Error Detection Page Enable Registers (L2EDPEN0)**

31	Page Enables for region 0	0
RW, +0000 0000 0000 0000 0000 0000 0000 0000		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 9. L2 Error Detection Page Enable Registers (L2EDPEN1)**

31	Page Enables for region 1	0
RW, +0000 0000 0000 0000 0000 0000 0000 0000		

LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Table 9. L2 Error Detection Page Enable Registers (L2EDPEN0) Field Descriptions**

Bit	Field	Value	Description
31-0	ENx	0	Page is disabled. Contents within the page are never checked against their parity value for errors, regardless of L2EDSTAT.EN.
		1	Page is enabled. Contents within the page are checked against their parity value for errors when L2EDSTAT.EN == '1'.

**Table 10. L2 Error Detection Page Enable Registers (L2EDPEN1) Field Descriptions**

Bit	Field	Value	Description
31-0	ENx	0	Page is disabled. Contents within the page are never checked against their parity value for errors, regardless of L2EDSTAT.EN.
		1	Page is enabled. Contents within the page are checked against their parity value for errors when L2EDSTAT.EN == '1'.

When the ENx bit for a given page is 1, the EDC logic functions as described. When the ENx bit is 0 the following logic holds good:

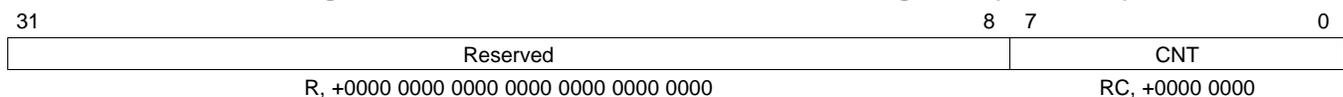
- L2 does not generate or modify stored parity or parity-valid
- L2 does not compute parity for reads or writes
- L2 does not check parity
- L2 generates no EDC-related exceptions
- L2 counts no EDC-related events in L2EDCPEC or L2EDNPEC

The power consumption can be reduced accordingly by disabling as much of the EDC logic as possible when it sees addresses within a page for which EN = 0. The amount of power savings varies from device to device.

### 3.8 L2 Error Detection Event Counter Registers (L2EDPEC, L2EDNPEC)

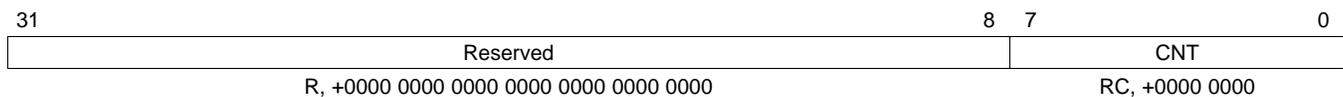
The EDC logic counts the number of correctable and non-correctable parity errors that occur. This allows software in a long-running system to assess the rate and pattern of parity error occurrences. L2 provides two 8-bit EDC event counters, L2EDCPEC and L2EDNPEC. L2EDCPEC provides the correctable parity error count, and L2EDNPEC provides the non-correctable parity error count. The L2 error detection event counter registers are shown in [Figure 10](#) and [Figure 11](#).

**Figure 10. L2 Error Detection Event Counter Registers (L2EDPEC)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

**Figure 11. L2 Error Detection Event Counter Registers (L2EDNPEC)**



LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

The L2 increments the appropriate counter even if a previously captured error has not been handled yet.

### 3.9 Error Exceptions

The EDC block in L2 has two exception signals that are routed to the CPU through the interrupt selection logic. These signals are UMC\_ED1 (default event number 116) and UMC\_ED2 (default event number 117). These two signals indicate when L2 has detected an error (UMC\_ED2 for un-correctable errors), and whether the error was corrected (UMC\_ED1 for correctable errors).

In either case, the CPU is then responsible to read the L2EDSTAT register, perform any necessary function (software error correction), then clear the associated PERR or DERR bit by writing to PCLR or DCLR, respectively. All single-bit errors except those coming from L1D record sufficient information in L2EDSTAT and L2EDADDR to enable software correction.

The EDC logic can only have one exception outstanding to the CPU at a time. If an exception is signaled to the CPU and a subsequent error is detected, a new exception cannot be generated until the current error status is cleared by the CPU. Likewise, the error status reflected in the status register cannot be updated with new error information until the previous status is cleared by the CPU.

There is one exception to this rule. If a non-correctable exception occurs after an unacknowledged correctable exception, the EDC logic records the details of the non-correctable exception and signals a non-correctable exception.

Because L2 permits simultaneous access to both UMAPs, two EDC failures can occur in the same cycle. L2 prioritizes non-correctable faults over correctable faults. It further prioritizes L1D over L1P in the case when two non-correctable faults occur in parallel.

### 3.10 Emulation Behavior

Parity data is calculated for emulation writes to the L2 memory in the same manner as for CPU writes. All emulation reads bypass the error detection hardware, however, and do not generate an exception if the parity does not correspond to the data.

When debugging an application, you can set and remove breakpoints. The debugger sets and clears software breakpoints with emulation writes to memory. Because these writes are narrower than 256 bits, they invalidate the parity associated with the fetch packet that they reside in, if the fetch packet is located in L2 RAM or L2 cache.

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

### Products

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>

### Applications

Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>

### TI E2E Community

[e2e.ti.com](http://e2e.ti.com)