# 3.1.1.8. DDRSS ECC

## 3.1.1.8.1. Overview

The DDR subsystem (DDRSS) comprises DDR controller, DDR PHY and wrapper logic to integrate these blocks in the device. For SDRAM data integrity, the DDRSS bridge supports inline ECC on the data written to or read from the SDRAM. ECC is stored together with the data so that a dedicated SDRAM device for ECC is not required. The 8-bit single error correction double error detection (SECDED) ECC data is calculated over 64-bit data quanta. For every 256-byte data block 32 bytes of ECC is stored inline. Thus 1/9th of the total SDRAM space is used for ECC storage and the rest 8/9th is available for system use. From system point of view that 8/9th of the SDRAM data space are seen as consecutive byte addresses. Even if there are non-ECC protected regions the previously described 1/9th-8/9th rule still applies and consecutive byte addresses are seen from system point of view. AM62x supports up to 3 ECC protected non-overlapping memory ranges.

The current U-Boot release supports a single region covering the entire SDRAM space.

The ECC is calculated for all accesses that are within the address ranges protected by ECC. 1-bit error is correctable by ECC and 2-bit error is not correctable and will be treated as unrecoverable error by software and trigger an exception.

### 3.1.1.8.1.1. DDRSS ECC Handling

The ECC feature of DDRSS is not enabled by default in U-Boot. But this can be enabled by adding the `ti,ecc-enable` boolean property to the `memorycontroller` node in arch/arm /dts/k3-am6*-ddr.dtsi and rebuilding U-Boot.

```
&memorycontroller {
        power-domains = <&k3_pds 170 TI_SCI_PD_SHARED>,
                        <&k3_pds 55 TI_SCI_PD_SHARED>;
        clocks = <&k3_clks 170 0>, <&k3_clks 16 4>;
        ti,ecc-enable;
};
```

The bootloader has the responsibility to pre-load the ECC protected region with known data before functional reads and writes are performed. During the ECC initialization, the R5 SPL fills the entire memory (2GB) with zeros using a CPU assisted memset or the BIST engine in the DDR controller. The BIST engine method allows priming with zeroes the entire 2GB memory space for around 1 second.

U-Boot command 'ddrss' can be used to simulate the ECC error by generating bit errors in SDRAM data at certain address. The ECC check can be tested by enabling ECC, shrinking the protected range, modifying a value, and then restoring range to original, and then reading the modified (and corrected) value, which will cause the ecc error count to increase. The 'ddrss' command simulates these steps.

The command syntax is:

```
ddrss ecc_err <addr in hex> <bit_err in hex>
```

The command will read a 32-bit data from `addr`, and write `(data ^ bit_err)` back to `addr`.

The following example generates a 1-bit error on bit 0 of the data at address 0x82400000.

```
=> ddrss ecc_err 82400000 1
        Testing DDR ECC:
        ECC test: Disabling DDR ECC ...
        ECC test: addr 0x82400000, read data 0x0, written data 0x1, err pattern: 0x1,
read after write data 0x1
        ECC test: Enabled DDR ECC ...
        ECC test: addr 0x82400000, read data 0x0
        ECC test: DDR ECC 1-bit error
        ECC test: 1-bit ECC err count: 1
        ECC test: 1-bit error in [0x2400000:0x2400008]
```

Reading from 0x82400000 returns a corrected value, the 1-bit error counter is incremented, and the offset of the error in the physical memory is reported in 8-byte

range.

The following example flips the bit back, which clears the error state.

```
=> ddrss ecc_err 82400000 1
Testing DDR ECC:
        ECC test: Disabling DDR ECC ...
        ECC test: addr 0x82400000, read data 0x1, written data 0x0, err pattern: 0x1,
read after write data 0x0
        ECC test: Enabled DDR ECC ...
        ECC test: addr 0x82400000, read data 0x0
```

The following example shows the message if the ECC is not enabled in DDRSS.

```
=> ddrss ecc_err 82400000 1
ECC not enabled. Please enable ECC any try again
```