应用手册

使用 RAM 奇偶校验增强系统可靠性

适用范围：

- SYM32 全系列芯片

文档说明

本应用手册将详细介绍 SYM32 系列微控制器使用 RAM 奇偶校验功能增强系统可靠性的方法。

1 为何要使用 RAM 奇偶校验

由于外部电磁干扰、半导体制造工艺上的缺陷以及其他原因可能导致二进制信息在传递或存错时发生错误，即可能由 1 变为 0 或者 0 变为 1。当发生错误时可能导致程序运行的结果不符合预期甚至产生较为严重的损失，因此需要能够检测到这类错误并及时处理以避免损失。

使用 RAM 奇偶校验功能能够有效的检测到传递或存储在 RAM 中的信息是否发生错误。通常奇偶校验码由两部分组成，一是信息位，即要传递的信息；二是奇偶校验位，仅一位。奇偶校验位的编码方式有两种，一是使信息位和校验位的“1”的个数为奇数，称为奇校验；二是使信息位和校验位的“1”的个数为偶数，称为偶校验。

2 SYM32 系列芯片 RAM 奇偶校验的特点

SYM32 系列芯片支持 RAM 的奇偶校验功能，具有以下特点：

- 以字节为单位的校验：每字节数据存放在 9bit 的物理空间中，8bit 数据位 + 1bit 校验位。
- 地址加数据的奇校验：采用当前 RAM 地址 + 当前数据的方式进行奇校验。
- 硬件计算并同步写入：对 RAM 写入数据时，控制器将同步写入数据位和校验位。
- 同步读出并硬件计算：从 RAM 读取数据时，控制器将同步读出数据位和校验位，并硬件进行奇偶校验。
- 硬件自动置位标志：若读出时校验出错则 RAM_ISR.PARITY 标志会被硬件置位。
- 硬件保存出错地址：若读出时校验出错则出错地址将存储到 RAM_ADDR 寄存器。
- 可配置的出错中断：若使能了奇偶校验出错中断（RAM_IER.PARITY = 1），当发生奇偶校验错误时 CPU 会执行校验出错中断服务程序。

SYM32 系列芯片的 RAM 支持三种读写方式：字节（8bit）、半字（16bit）、字（32bit）。对于字节操作模式，当奇偶校验出错时 RAM_ADDR 寄存器中的地址是错误字节所在的地址；对于半字操作模式，当奇偶校验出错时 RAM_ADDR 寄存器中的地址是错误半字所在的地址；对于全字操作模式，当奇偶校验出错时 RAM_ADDR 寄存器中的地址是错误字所在的地址。

3 SYM32 系列芯片 RAM 奇偶校验的使用

3.1 RAM 奇偶校验使用的步骤

Step1. 清除奇偶校验错误标志

Step2. 使能奇偶校验错误中断

Step3. 清除 RAM 对应的 NVIC 标志并使能 NVIC

Step4. 在中断服务程序中进行可靠性、安全性处理，例如上报错误信息、进行紧急事件处理、保存当前数据等。

注意事项：由于 RAM 的固有特性，在上电后未写入过数据的 RAM 内容为随机值，故每个字节数据奇偶校验出错的机率为 50% 。因此不可对未写入过数据的 RAM 进行读出奇偶校验，如若进行则每个字节会有 50%的奇偶校验出错的概率。

3.2 RAM 奇偶校验功能的代码实现（寄存器方式）

```
void RAM_Parity_Init(void)
{
    // Step1: 清除奇偶校验错误标志
    SYM_RAM->ICR = 0;

    // Step2: 使能奇偶校验错误中断
    SYM_RAM->IER_f.PARITY = 1;

    // Step3: 清除 RAM 对应的 NVIC 标志并使能 NVIC
    NVIC_ClearPendingIRQ(FLASHRAM_IRQn);
    NVIC_EnableIRQ(FLASHRAM_IRQn);
}

// Step4: 实现中断服务程序
void FLASHRAM_IRQHandler(void)
{
    // 确认是 RAM 奇偶校验错误中断
    if (SYM_RAM->IER_f.PARITY && SYM_RAM->ISR)
    {
        // 清除 RAM 奇偶校验错误标志
        SYM_RAM->ICR = 0;

        // 可读出错误地址进行处理（可选）
        uint32_t ErrorAddr = SYM_RAM->ADDR;

        // 用户进行可靠性、安全性的处理 开始
        // 例如上报错误信息、进行紧急事件处理、保存当前数据等。
        // .....
        // 用户进行可靠性、安全性的处理 结束

        while (1);
    }
}
```

3.3 RAM 奇偶校验功能的代码实现（库函数方式）

```
void RAM_Parity_Init(void)
{
    // Step1: 清除奇偶校验错误标志
    HAL_RAM_CLR_FLAG(RAM_IT_FLAG_PARITY);

    // Step2: 使能奇偶校验错误中断
    HAL_RAM_ENABLE_IT(RAM_IT_SOURCE_PARITY);

    // Step3: 清除 RAM 对应的 NVIC 标志并使能 NVIC
    NVIC_ClearPendingIRQ(FLASHRAM_IRQn);
    NVIC_EnableIRQ(FLASHRAM_IRQn);
}

// Step4: 实现中断服务程序
void FLASHRAM_IRQHandler(void)
{
    // 确认是 RAM 奇偶校验错误中断
    if (HAL_RAM_GET_IT_SOURCE(RAM_IT_SOURCE_PARITY)
        && HAL_RAM_GET_FLAG(RAM_IT_FLAG_PARITY))
    {
        // 清除 RAM 奇偶校验错误标志
        HAL_RAM_CLR_FLAG(RAM_IT_FLAG_PARITY);

        // 可读出错误地址进行处理（可选）
        uint32_t ErrorAddr = HAL_RAM_GetParityErrorAddr();

        // 用户进行可靠性、安全性的处理 开始
        // 例如上报错误信息、进行紧急事件处理、保存当前数据等。
        // .....
        // 用户进行可靠性、安全性的处理 结束

        while (1);
    }
}
```

4 版本记录

版本	修订日期	修订说明
Rev1.0	2023-01-06	初始版本