MSP使用的HardFault_Handler

chatgpt.com/c/ba0f5a3c-ba60-4f35-b57b-e2d7afc1ff4c

> STM32单片机，如果进入HardFault_Handler的时候使用的是PSP，如果在HardFault_Handler中使用push指令，此时操作的是MSP还是PSP

在STM32单片机中，异常处理程序（包括 HardFault_Handler）使用的堆栈指针可能会根据具体情况有所不同。通常情况下，异常处理程序会切换到主堆栈指针（MSP）来处理异常。

当一个异常（如硬故障）发生时，处理器会自动切换到主堆栈指针（MSP），即使在进入异常处理程序之前使用的是进程堆栈指针（PSP）。因此，在 HardFault_Handler 中执行的 `push` 指令会使用 MSP 而不是 PSP。

具体来说，当异常发生时：

1. 处理器将当前寄存器的内容压入当前使用的堆栈（可能是 PSP）。
2. 处理器切换到使用主堆栈指针（MSP）。
3. 跳转到异常处理程序（如 HardFault_Handler）。

因此，在 HardFault_Handler 中执行的 `push` 指令将操作 MSP。

以下是一个简单的示例，展示异常发生时如何使用 MSP：

c

```c
void HardFault_Handler(void) {
    // 此时处理器使用的是 MSP
    __asm volatile (
        "TST lr, #4 \n"          // 检查异常发生时使用的是哪个堆栈指针
        "ITE EQ \n"
        "MRSEQ r0, MSP \n"       // 如果是0，使用MSP
        "MRSNE r0, PSP \n"       // 如果是1，使用PSP
    );

    // 此时操作的堆栈指针是MSP
    // 您可以在这里添加自定义的异常处理代码
    while (1); // 循环停止
}
```

简而言之，当进入 `HardFault_Handler` 时，处理器会切换到使用主堆栈指针（MSP），并且在 `HardFault_Handler` 中使用 `push` 指令将操作 MSP。

如果在HardFault_Handler中想把R4-R11的值压入进入HardFault_Handler前使用的栈，应该怎样处理

如果在 HardFault_Handler 中想将 R4-R11 的值压入进入 HardFault_Handler 前使用的栈（即可能是 PSP 或 MSP），可以通过以下步骤实现：

1. 检查并确定进入 HardFault_Handler 之前使用的是 PSP 还是 MSP。
2. 根据确定的堆栈指针，将 R4-R11 的值压入对应的堆栈。

以下是一个示例代码，展示如何在 HardFault_Handler 中实现这一点：

c

```
void HardFault_Handler(void) {
    __asm volatile (
        "TST lr, #4 \n"          // 检查异常发生时使用的是哪个堆栈指针
        "ITE EQ \n"
        "MRSEQ r0, MSP \n"       // 如果是0，使用MSP
        "MRSNE r0, PSP \n"       // 如果是1，使用PSP

        // 将 R4-R11 压入进入 HardFault_Handler 前使用的堆栈
        "STMDB r0!, {r4-r11} \n"

        // 保存堆栈指针到 r1（可以根据需要进行进一步处理）
        "MOV r1, r0 \n"

        // 其余的异常处理代码（例如调用 C 函数处理异常）
        "B HardFault_Handler_C \n"
    );
}

// 定义 C 函数处理异常（可选）
void HardFault_Handler_C(uint32_t *fault_stack_address) {
    // 现在可以访问保存的寄存器值
    // fault_stack_address 指向压入堆栈的 R4-R11
    // 您可以在这里添加自定义的异常处理代码
    while (1); // 循环停止
}
```

上述代码实现的步骤解释如下：

1. 使用 TST lr, #4 和条件执行指令（ITE EQ, MRSEQ, MRSNE）来检查和选择正确的堆栈指针（PSP 或 MSP）。
2. 将 R4-R11 压入选择的堆栈指针（即进入 HardFault_Handler 前使用的堆栈）。

3. 将堆栈指针地址保存到 r1 寄存器，并跳转到一个 C 函数（`HardFault_Handler_C`）进一步处理异常。

在 C 函数 `HardFault_Handler_C` 中，传递的 `fault_stack_address` 指向保存的寄存器值（R4-R11），您可以根据需要进一步处理这些值。

这种方法确保在 `HardFault_Handler` 中将 R4-R11 寄存器值保存到进入异常处理程序前使用的正确堆栈中。