

不少工程师在项目开发过程中会遇到代码运行进 `HardFault_Handler` 中断的情况。因进 `HardFault_Handler` 中断的原因（RAM 溢出/空指针异常/堆栈溢出等等）比较多，情况比较复杂，搞得工程师没有头绪。现提供排查思路如下：

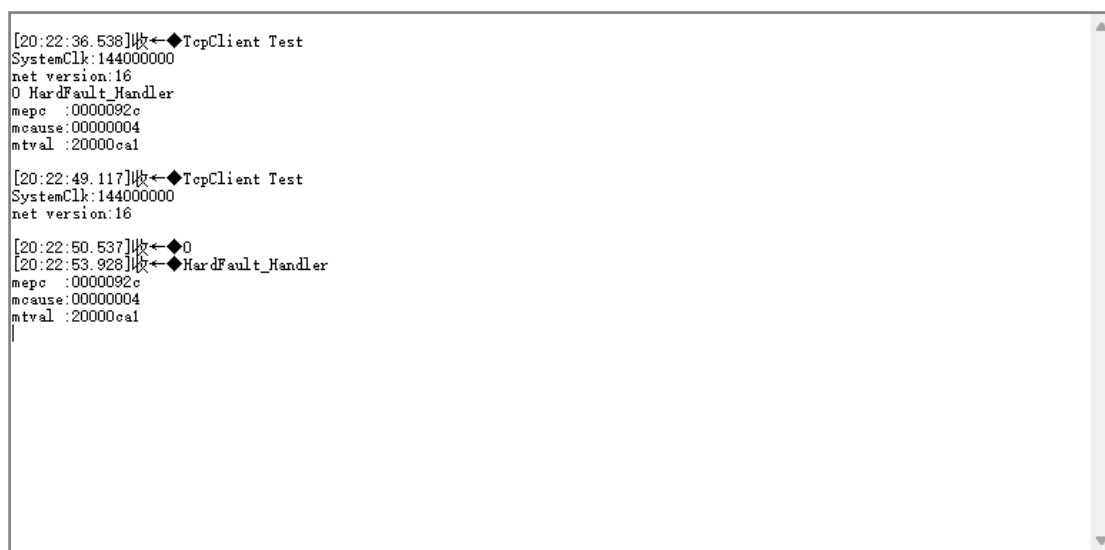
### HardFault\_Handler 定位：

可在 `void HardFault_Handler(void)` 中断服务函数中添加如下代码：

```
printf("mepc :%08x\r\n", __get_MEPC());
printf("mcause:%08x\r\n", __get_MCAUSE());
printf("mtval :%08x\r\n", __get_MTVAL());
while(1);
```

该操作仅适用于代码串口可用的情况下，如串口不可用但可以仿真，也可通过仿真查看 `__get_MEPC()`；`__get_MCAUSE()`；`__get_MTVAL()`；三个函数的返回值。

串口呈现方式：（图一）

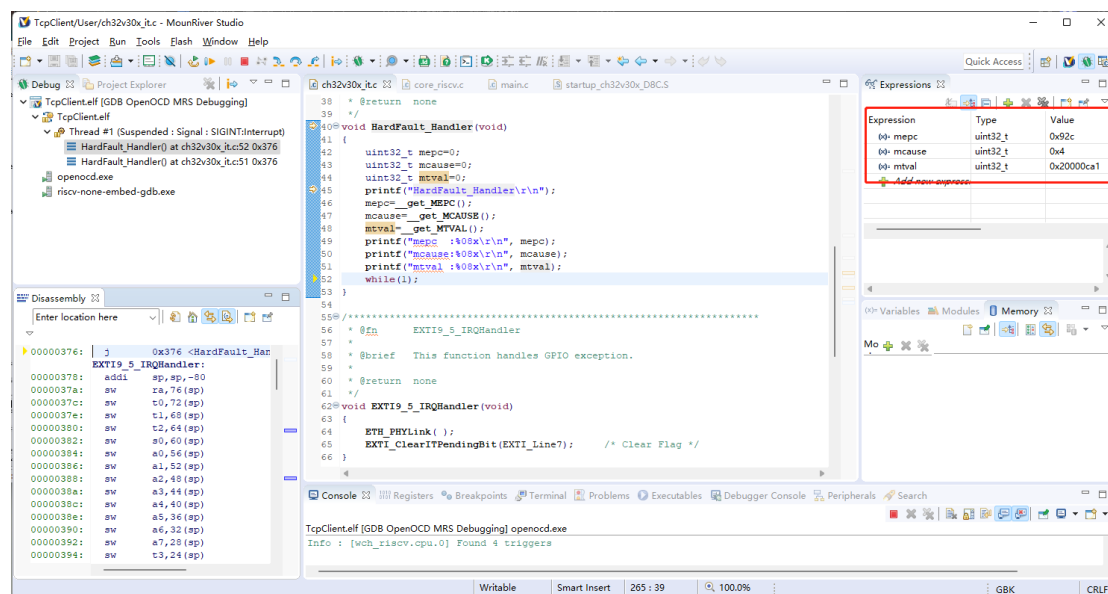


```
[20:22:36.538]收←◆TcpClient Test
SystemClk:14400000
net version:16
0 HardFault_Handler
mepc :0000092c
mcause:00000004
mtval :20000ca1

[20:22:49.117]收←◆TcpClient Test
SystemClk:14400000
net version:16

[20:22:50.537]收←◆0
[20:22:53.926]收←◆HardFault_Handler
mepc :0000092c
mcause:00000004
mtval :20000ca1
```

中断呈现方式：（图二）



仿真操作可参考该帖：

[https://blog.csdn.net/qq\\_36353650/article/details/120441264?spm=1001.2014.3001.5502](https://blog.csdn.net/qq_36353650/article/details/120441264?spm=1001.2014.3001.5502)

[https://blog.csdn.net/qq\\_36353650/article/details/120665601?spm=1001.2014.3001.5502](https://blog.csdn.net/qq_36353650/article/details/120665601?spm=1001.2014.3001.5502)

### 关于三个函数返回值的介绍:

MCAUSE: 反映当前的异常种类或中断的编号, mcause[31]=1 表示为中断, mcause[31]=0 表示为异常。本文仅分析mcause[31]=0的情况。

表一-异常编码

interrupt	异常编码	同步/异步	异常原因
1	0-1	-	保留
1	2	精确异步	NMI 中断
1	3-11	-	保留
1	12	精确异步	SysTick 中断
1	13	-	保留
1	14	同步	软件中断
1	15	-	保留
1	16-255	精确异步	外部中断 16-255
0	0	同步	指令地址不对齐
0	1	同步	取指令访问错误
0	2	同步	非法指令
0	3	同步	断点
0	4	同步	Load 指令访存地址不对齐
0	5	非精确异步	Load 指令访存错误
0	6	同步	Store/AMO 指令访存地址不对齐
0	7	非精确异步	Store/AMO 指令访存错误
0	8	同步	用户模式下环境调用
0	11	同步	机器模式下环境调用

MEPC: 机器模式异常指针寄存器, 标准定义退出异常或中断后微处理器的返回地址保存在MEPC中。所以当发生异常或中断后, 硬件自动更新MEPC 值为当前遇到异常时的指令PC 值, 或中断前下一条预执行的指令PC值。

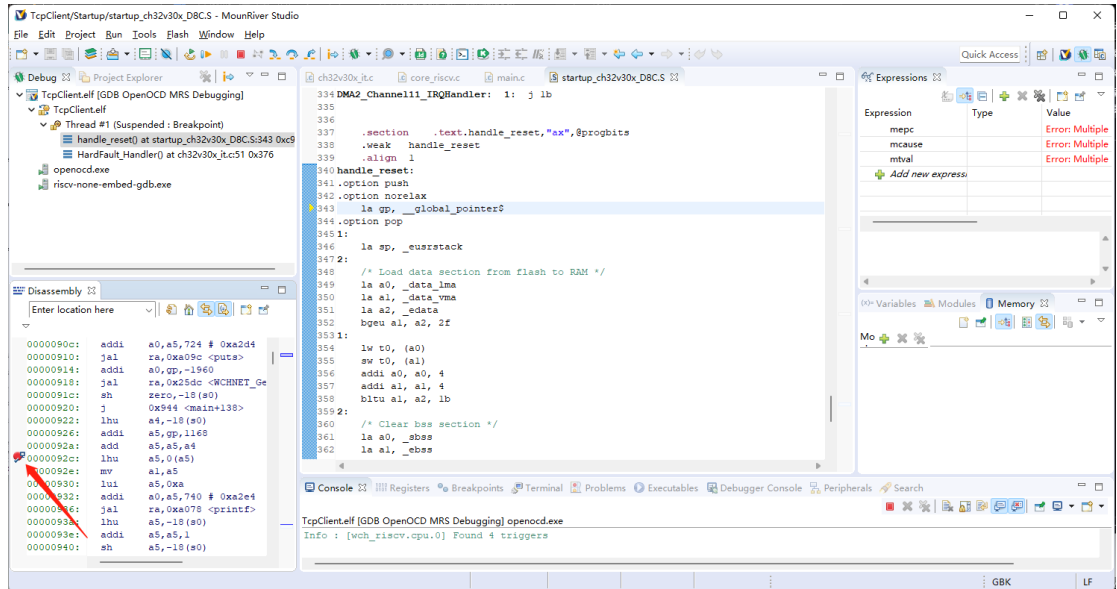
MTVAL: 该值即为引起异常的值, 有以下几种情况:

1. 如果存储器访问引起的异常, 硬件会将异常时存储器访问的地址存入mtval。
2. 如果是非法指令引起的异常, 硬件会将该指令的指令编码存入mtval。
3. 如果是硬件断点引起的异常, 硬件会将断点处PC值存入mtval。
4. 对于其他的异常, 硬件将mtval的值设为0, 例如ebreak, ecall 指令引起的异常。
5. 进入中断时, 硬件将mtval的值设为0。

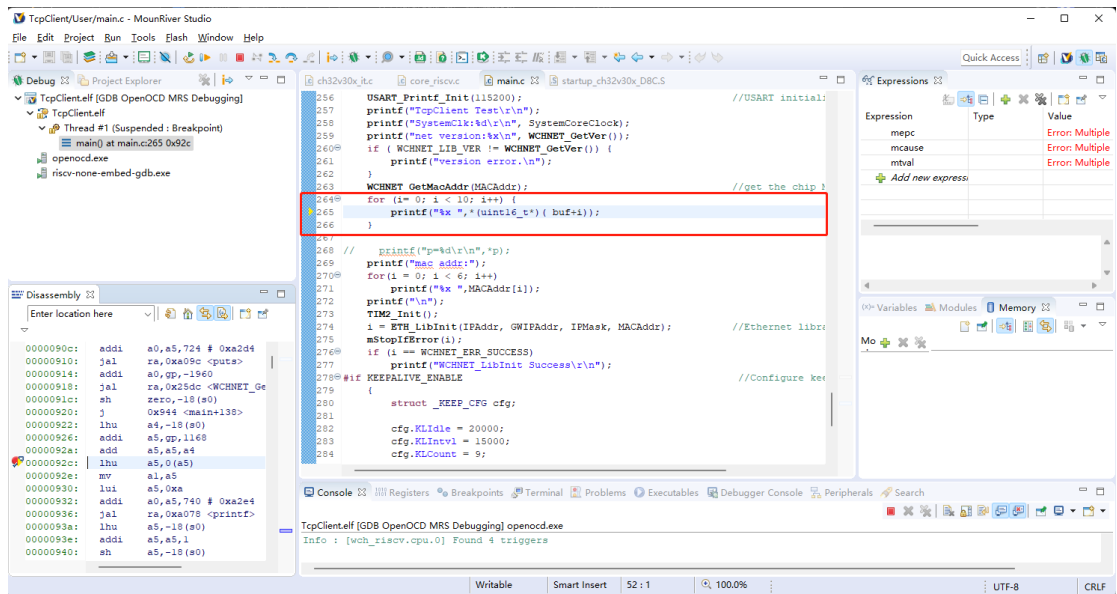
综上所述, 我们可以通过MEPC的值确定代码进HardFault\_Handler中断的位置, 在某些情况下可以方便我们确定报错原因。本文提供两种方法:

#### 方法一 仿真法:

以上文图一呈现的HardFault报错信息为例, 由打印信息或仿真信息可知MEPC=0x92c, 且mcause[31]=0, 因此触发HardFault为异常而不是中断, 所以出现异常的PC指针地址为0x92c。我们可以在仿真时在0x92c处打断点, 如图所示: (图三)



然后全速运行，代码就会停在出错位置附近，如图所示：（图四）



方法二 查表法：可在所在工程的.lst文件中搜索MEPC值，即该案例中的92c，如图：（图五）

```

core_riscv.c | main.c | startup_ch32v30x_D8C.S | TcpClientIst
1880 }
1881 WCHNET_GetMacAddr(MACAddr); //get the chip MAC address
1882 914: 85818513      addi    a0,gp,-1960 # 20000068 <_edata>
1883 918: 4c5010ef      jal    ra,25dc <WCHNET_GetMacAddr>
1884 E:\work\压缩包\V307_ETH_FULL\ETH\TcpClient\obj\../User/main.c:264
1885     for (i= 0; i < 10; i++) {
1886     91c: fe041723      sh    zero,-18(s0)
1887     920: a015        s    s44,main+0x5a)
1888 E:\work\压缩包\V307_ETH_FULL\ETH\TcpClient\obj\../User/main.c:265 (discriminator 3)
1889     printf("%x ",*(uint16_t*)( buf+i));
1890     922: fee45703      lhu   a4,-18(s0)
1891     926: 49018793      addi   a5,gp,1168 # 20000ca0 <buf>
1892     92a: 97ba        add   a5,a5,a4
1893     92c: 239e        lhu   a5,0(a5)
1894     92e: 83be        mv    a1,a5
1895     930: 67a5        lui   a5,0xa
1896     932: 2e478513      addi   a0,a5,740 # a2e4 <memcpy+0x1e6>
1897     936: 742090ef      jal    ra,a078 <printf>
1898 E:\work\压缩包\V307_ETH_FULL\ETH\TcpClient\obj\../User/main.c:264 (discriminator 3)
1899     for (i= 0; i < 10; i++) {
1900     93a: fee45783      lhu   a5,-18(s0)
1901     93e: 0795        addi   a5,a5,1
1902     940: fe041723      sh    a5,-18(s0)
1903 E:\work\压缩包\V307_ETH_FULL\ETH\TcpClient\obj\../User/main.c:264 (discriminator 1)
1904     944: fee45703      lhu   a4,-18(s0)
1905     948: 47a5        li    a5,9
1906     94a: fce7fce3      bgeu  a5,a4,922 <main+0xe8>
1907 E:\work\压缩包\V307_ETH_FULL\ETH\TcpClient\obj\../User/main.c:269
1908     }
1909
1910 //     printf("p=%d\r\n",*p);
1911     printf("mac addr:");
1912     94e: 67a9        lui   a5,0xa
1913     950: 2e478513      addi   a0,a5,744 # a2e8 <memcpy+0x1ea>

```

由lst文件可知报错函数为printf("%x ",\*(uint16\_t\*)( buf+i));如果你是汇编大佬也可以分析一下这段代码，相信你也可以分析出报错原因。如果不是也不要灰心，可以继续往下看。

HardFault\_Handler报错原因分析:

由上文可知报错原因可以通过MCAUSE与MTVAL确定，该案例中MCAUSE=4，MTVAL=20000ca1。

由表一可知该案例的报错原因为Load指令访存地址不对齐，结合MTVAL的情况1可知异常时存储器访问的地址为0x20000ca1。显然该地址为RAM地址结合代码可知，该问题是由于定义的数组为8位，但通过16位访问造成的。

```

core_riscv.c | main.c | startup_ch32v30x_D8C.S | TcpClientIst
238     WCHNET_HandleSockInt(i, socketint);
239 }
240 }
241 }
242 u8 buf[10];
243
244 @*****
245 * @fn      main
246 *
247 * @brief   Main program
248 *
249 * @return  none
250 */
251 int main(void)
252 {
253     ul6 i;
254
255     Delay_Init();
256     USART_Printf_Init(115200); //USART initialize
257     printf("TcpClient Test\r\n");
258     printf("SystemClk:%d\r\n", SystemCoreClock);
259     printf("net version:%x\r\n", WCHNET_GetVer());
260     if ( WCHNET_LIB_VER != WCHNET_GetVer() ) {
261         printf("version error.\r\n");
262     }
263     WCHNET_GetMacAddr(MACAddr); //get the chip MAC address
264     for (i= 0; i < 10; i++) {
265         printf("%x ",*(uint16_t*)( buf+i));
266     }
267
268     //     printf("p=%d\r\n",*p);
269     printf("mac addr:");
270     for(i = 0; i < 6; i++)
271         printf("%x ",MACAddr[i]);

```

注：如果通过仿真的方式查找问题且应用中有读写FLASH/低功耗/看门狗等操作，需将相关代码注释然后再仿真，否则将无法仿真。本文只是提供了一个查找HardFault\_Handler问题的思路，实际情况可能比案例要复杂的多。可根据实际情况灵活处理。