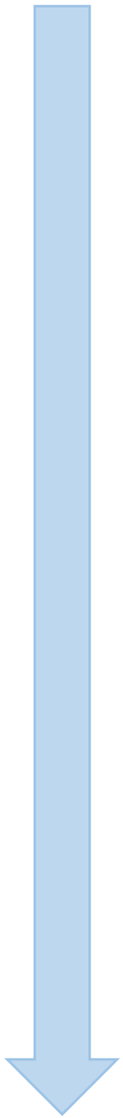
Cortex-M4 异常分析

Youwei Wang

V1.0

WWW.NOT

分析流程及代码结构



main()使能除0异常后，
调用test_funcA()

test_funcA()调用test_funcB()

test_funcB()中除0操作(模拟产生错误)

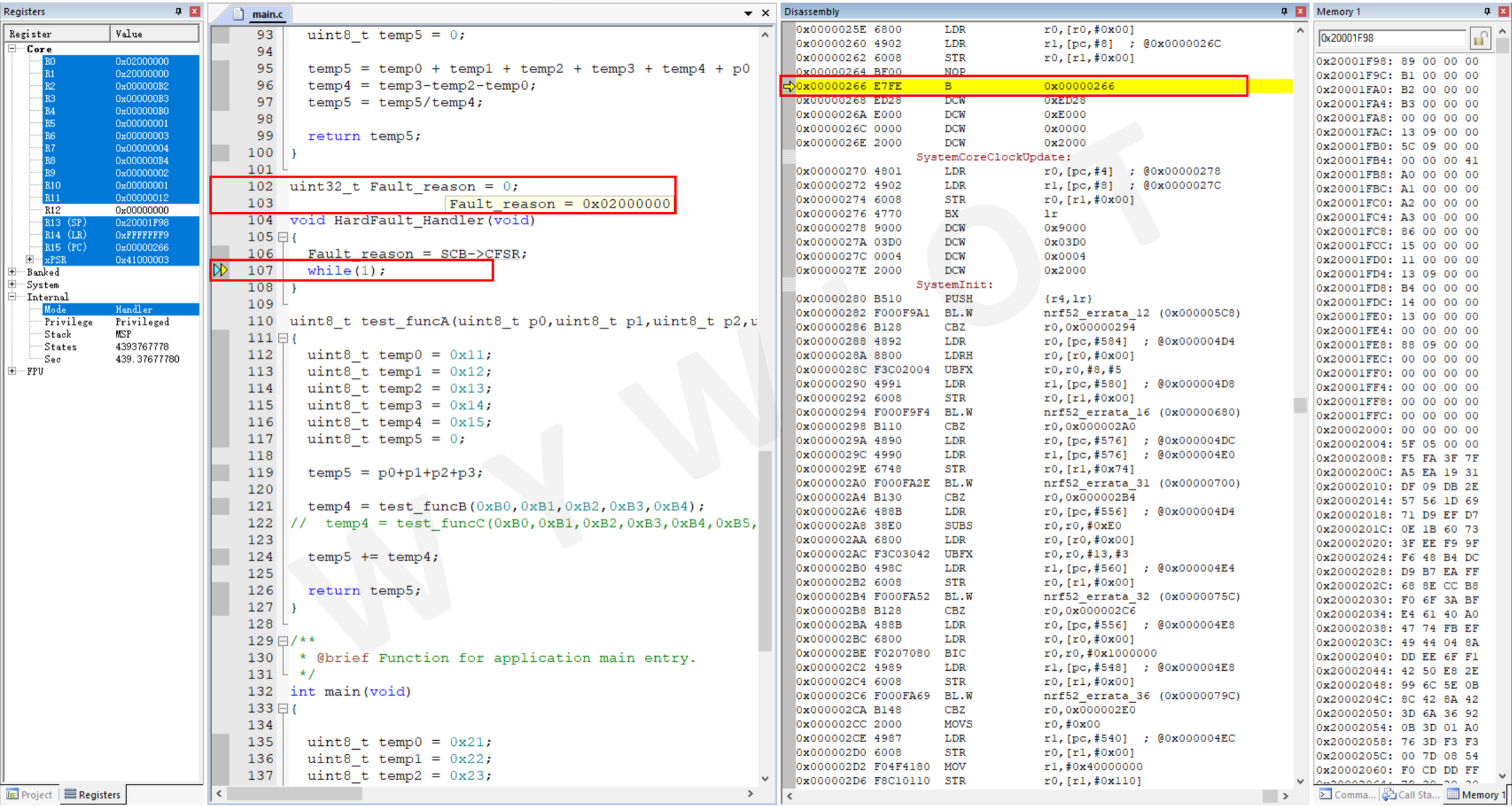
进入Hardfault()异常回调

分析栈数据和SCB中的寄存器找到错误代码和原因

```
void HardFault_Handler(void)
{
  …
}

uint8_t test_funcB(…)
{
  …
  //除0操作
}

uint8_t test_funcA(…)
{
  …
  test_funcB(…);
  …
}

mian()
{
  …
  //使能除0异常
  test_funcA(…);
  …
}
```

WWW.IOT

运行测试代码后进入HardFault回调



Registers

Register	Value
Core	
R0	0x02000000
R1	0x20000000
R2	0x000000B2
R3	0x000000B3
R4	0x000000B0
R5	0x00000001
R6	0x00000003
R7	0x00000004
R8	0x000000B4
R9	0x00000002
R10	0x00000001
R11	0x00000012
R12	0x00000000
R13 (SP)	0x20001F98
R14 (LR)	0xFFFFFFF9
R15 (PC)	0x00000266
xPSR	0x41000003
Banked	
System	
Internal	
Mode	Handler
Privilege	Privileged
Stack	MSP
States	4393767778
Sec	439.37677780
FPU	

Project | Registers

main.c

```
 93    uint8_t temp5 = 0;
 94
 95    temp5 = temp0 + temp1 + temp2 + temp3 + temp4 + p0
 96    temp4 = temp3-temp2-temp0;
 97    temp5 = temp5/temp4;
 98
 99    return temp5;
100  }
101
102  uint32_t Fault_reason = 0;
103                          Fault_reason = 0x02000000
104  void HardFault_Handler(void)
105  {
106    Fault_reason = SCB->CFSR;
107    while(1);
108  }
109
110  uint8_t test_funcA(uint8_t p0,uint8_t p1,uint8_t p2,u
111  {
112    uint8_t temp0 = 0x11;
113    uint8_t temp1 = 0x12;
114    uint8_t temp2 = 0x13;
115    uint8_t temp3 = 0x14;
116    uint8_t temp4 = 0x15;
117    uint8_t temp5 = 0;
118
119    temp5 = p0+p1+p2+p3;
120
121    temp4 = test_funcB(0xB0,0xB1,0xB2,0xB3,0xB4);
122  //  temp4 = test_funcC(0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,
123
124    temp5 += temp4;
125
126    return temp5;
127  }
128
129  /**
130   * @brief Function for application main entry.
131   */
132  int main(void)
133  {
134
135    uint8_t temp0 = 0x21;
136    uint8_t temp1 = 0x22;
137    uint8_t temp2 = 0x23;
```

Disassembly

```
0x0000025E 6800      LDR          r0,[r0,#0x00]
0x00000260 4902      LDR          r1,[pc,#8]  ; @0x0000026C
0x00000262 6008      STR          r0,[r1,#0x00]
0x00000264 BF00      NOP
0x00000266 E7FE      B            0x00000266
0x00000268 ED28      DCW          0xED28
0x0000026A E000      DCW          0xE000
0x0000026C 0000      DCW          0x0000
0x0000026E 2000      DCW          0x2000
                SystemCoreClockUpdate:
0x00000270 4801      LDR          r0,[pc,#4]  ; @0x00000278
0x00000272 4902      LDR          r1,[pc,#8]  ; @0x0000027C
0x00000274 6008      STR          r0,[r1,#0x00]
0x00000276 4770      BX           lr
0x00000278 9000      DCW          0x9000
0x0000027A 03D0      DCW          0x03D0
0x0000027C 0004      DCW          0x0004
0x0000027E 2000      DCW          0x2000
                SystemInit:
0x00000280 B510      PUSH         {r4,lr}
0x00000282 F000F9A1  BL.W         nrf52_errata_12 (0x000005C8)
0x00000286 B128      CBZ          r0,0x00000294
0x00000288 4892      LDR          r0,[pc,#584]  ; @0x000004D4
0x0000028A 8800      LDRH         r0,[r0,#0x00]
0x0000028C F3C02004  UBFX         r0,r0,#8,#5
0x00000290 4991      LDR          r1,[pc,#580]  ; @0x000004D8
0x00000292 6008      STR          r0,[r1,#0x00]
0x00000294 F000F9F4  BL.W         nrf52_errata_16 (0x00000680)
0x00000298 B110      CBZ          r0,0x000002A0
0x0000029A 4890      LDR          r0,[pc,#576]  ; @0x000004DC
0x0000029C 4990      LDR          r1,[pc,#576]  ; @0x000004E0
0x0000029E 6748      STR          r0,[r1,#0x74]
0x000002A0 F000FA2E  BL.W         nrf52_errata_31 (0x00000700)
0x000002A4 B130      CBZ          r0,0x000002B4
0x000002A6 488B      LDR          r0,[pc,#556]  ; @0x000004D4
0x000002A8 38E0      SUBS         r0,r0,#0xE0
0x000002AA 6800      LDR          r0,[r0,#0x00]
0x000002AC F3C03042  UBFX         r0,r0,#13,#3
0x000002B0 498C      LDR          r1,[pc,#560]  ; @0x000004E4
0x000002B2 6008      STR          r0,[r1,#0x00]
0x000002B4 F000FA52  BL.W         nrf52_errata_32 (0x0000075C)
0x000002B8 B128      CBZ          r0,0x000002C6
0x000002BA 488B      LDR          r0,[pc,#556]  ; @0x000004E8
0x000002BC 6800      LDR          r0,[r0,#0x00]
0x000002BE F0207080  BIC          r0,r0,#0x1000000
0x000002C2 4989      LDR          r1,[pc,#548]  ; @0x000004E8
0x000002C4 6008      STR          r0,[r1,#0x00]
0x000002C6 F000FA69  BL.W         nrf52_errata_36 (0x0000079C)
0x000002CA B148      CBZ          r0,0x000002E0
0x000002CC 2000      MOVS         r0,#0x00
0x000002CE 4987      LDR          r1,[pc,#540]  ; @0x000004EC
0x000002D0 6008      STR          r0,[r1,#0x00]
0x000002D2 F04F4180  MOV          r1,#0x40000000
0x000002D6 F8C10110  STR          r0,[r1,#0x110]
```

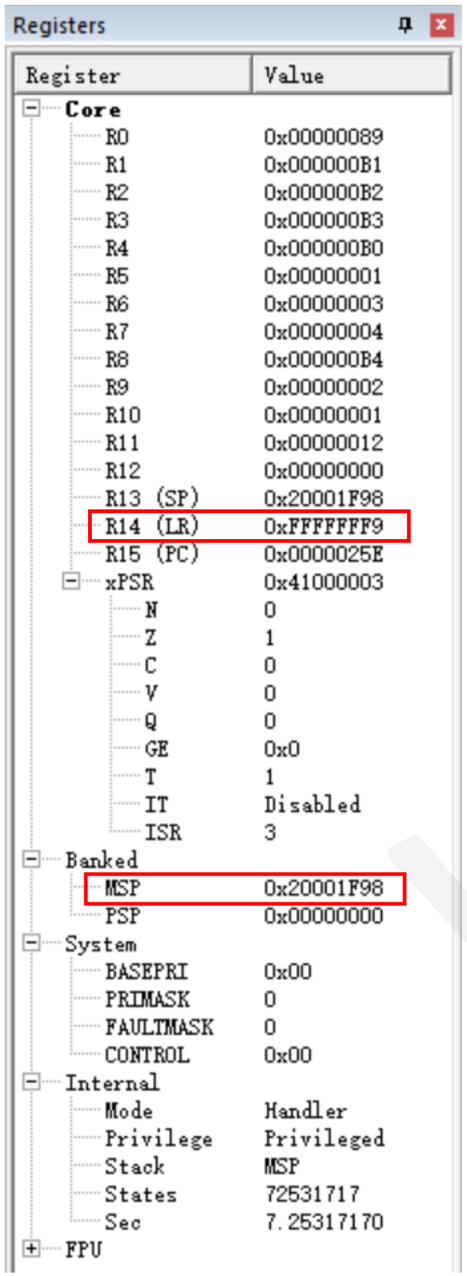
Memory 1

0x20001F98

```
0x20001F98: 89 00 00 00
0x20001F9C: B1 00 00 00
0x20001FA0: B2 00 00 00
0x20001FA4: B3 00 00 00
0x20001FA8: 00 00 00 00
0x20001FAC: 13 09 00 00
0x20001FB0: 5C 09 00 00
0x20001FB4: 00 00 00 41
0x20001FB8: A0 00 00 00
0x20001FBC: A1 00 00 00
0x20001FC0: A2 00 00 00
0x20001FC4: A3 00 00 00
0x20001FC8: 86 00 00 00
0x20001FCC: 15 00 00 00
0x20001FD0: 11 00 00 00
0x20001FD4: 13 09 00 00
0x20001FD8: B4 00 00 00
0x20001FDC: 14 00 00 00
0x20001FE0: 13 00 00 00
0x20001FE4: 00 00 00 00
0x20001FE8: 88 09 00 00
0x20001FEC: 00 00 00 00
0x20001FF0: 00 00 00 00
0x20001FF4: 00 00 00 00
0x20001FF8: 00 00 00 00
0x20001FFC: 00 00 00 00
0x20002000: 00 00 00 00
0x20002004: 5F 05 00 00
0x20002008: F5 FA 3F 7F
0x2000200C: A5 EA 19 31
0x20002010: DF 09 DB 2E
0x20002014: 57 56 1D 69
0x20002018: 71 D9 EF D7
0x2000201C: 0E 1B 60 73
0x20002020: 3F EE F9 9F
0x20002024: F6 48 B4 DC
0x20002028: D9 B7 EA FF
0x2000202C: 68 8E CC B8
0x20002030: F0 6F 3A BF
0x20002034: E4 61 40 A0
0x20002038: 47 74 FB EF
0x2000203C: 49 44 04 8A
0x20002040: DD EE 6F F1
0x20002044: 42 50 E8 2E
0x20002048: 99 6C 5E 0B
0x2000204C: 8C 42 8A 42
0x20002050: 3D 6A 36 92
0x20002054: 0B 3D 01 A0
0x20002058: 76 3D F3 F3
0x2000205C: 00 7D 08 54
0x20002060: F0 CD DD FF
```

Comma... | Call Sta... | Memory 1

根据LR(EXC_RETURN)查找返回地址

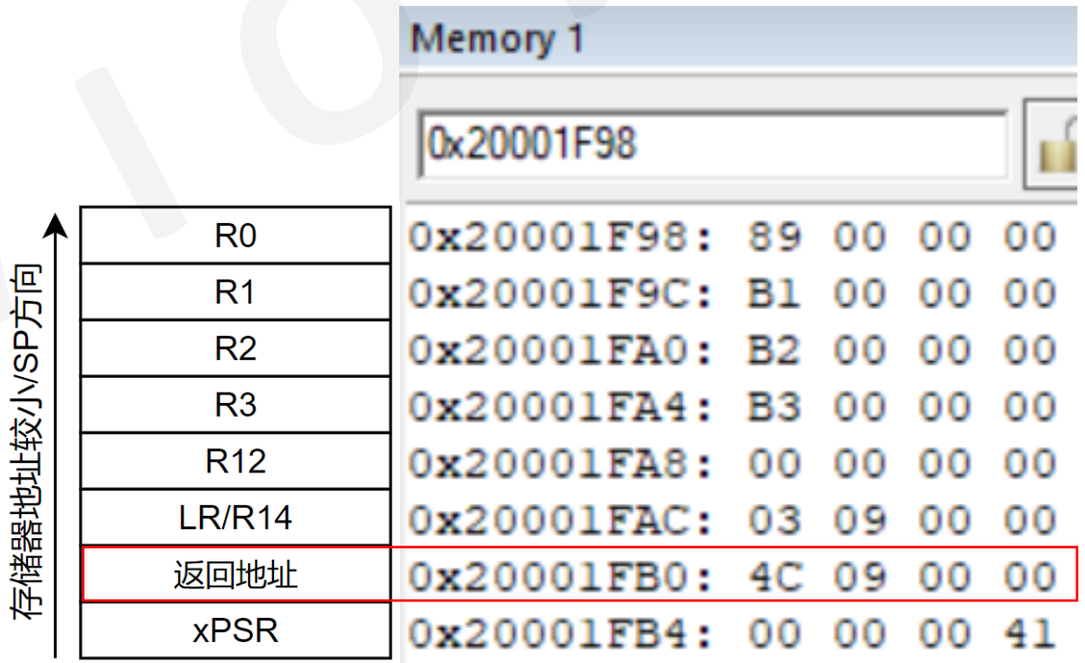


LR=EXC_RETURN=0xFFFFFFF9

0xF9=1111 1001B

返回线程模式,MSP,8字

MSP=0x20001F98



异常处理之后的返回地址为：**0x94C**

该地址是产生异常之前的程序位置。

WWW.IOT

根据返回地址查找错误地址

```
0x000008F8 21B1      MOVS          r1,#0xB1
0x000008FA 9000      STR           r0,[sp,#0x00]
0x000008FC 20B0      MOVS          r0,#0xB0
0x000008FE F000F808  BL.W          test_funcB (0x00000912)
0x00000902 4681      MOV           r9,r0
0x00000904 EB080009  ADD           r0,r8,r9
0x00000908 F00008FF  AND           r8,r0,#0xFF
0x0000090C 4640      MOV           r0,r8
0x0000090E E8BD8FFE  POP           {r1-r11,pc}
                test_funcB:
0x00000912 E92D47F0  PUSH          {r4-r10,lr}
0x00000916 4604      MOV           r4,r0
0x00000918 F8DD8020  LDR           r8,[sp,#0x20]
0x0000091C 2501      MOVS          r5,#0x01
0x0000091E F04F0902  MOV           r9,#0x02
0x00000922 2603      MOVS          r6,#0x03
0x00000924 2704      MOVS          r7,#0x04
0x00000926 F04F0C05  MOV           r12,#0x05
0x0000092A 2000      MOVS          r0,#0x00
0x0000092C EB050A09  ADD           r10,r5,r9
0x00000930 44B2      ADD           r10,r10,r6
0x00000932 44BA      ADD           r10,r10,r7
0x00000934 44E2      ADD           r10,r10,r12
0x00000936 44A2      ADD           r10,r10,r4
0x00000938 448A      ADD           r10,r10,r1
0x0000093A 4492      ADD           r10,r10,r2
0x0000093C 449A      ADD           r10,r10,r3
0x0000093E 44C2      ADD           r10,r10,r8
0x00000940 F00A00FF  AND           r0,r10,#0xFF
0x00000944 EBA70A06  SUB           r10,r7,r6
0x00000948 EBAA0C05  SUB           r12,r10,r5
0x0000094C FB90FAFC  SDIV          r10,r0,r12
0x00000950 F00A00FF  AND           r0,r10,#0xFF
0x00000954 E8BD87F0  POP           {r4-r10,pc}
0x00000958 0978      DCW           0x0978
0x0000095A 0000      DCW           0x0000
0x0000095C 0000      DCW           0x0000
```

根据地址0x94C找到汇编代码

结合上下文找到对应的C函数

WWW.IOT

分析C与汇编代码查找错误

结合函数中的C和汇编代码定位出错C代码：temp5=temp5/temp4；处产生的异常。

```
uint8_t test_funcB(uint8_t p0,uint8_t p1,uint8_t p2,u
{
  uint8_t temp0 = 0x1;
  uint8_t temp1 = 0x2;
  uint8_t temp2 = 0x3;
  uint8_t temp3 = 0x4;
  uint8_t temp4 = 0x5;
  uint8_t temp5 = 0;

  temp5 = temp0 + temp1 + temp2 + temp3 + temp4 + p0
  temp4 = temp3-temp2-temp0;
  temp5 = temp5/temp4;

  return temp5;
}

uint32_t Fault_reason = 0;

void HardFault_Handler(void)
{
  Fault_reason = SCB->CFSR;
  while(1);
}
```

```
                test_funcB:
0x00000922 E92D47F0  PUSH    {r4-r10,lr}
0x00000926 4604      MOV     r4,r0
0x00000928 F8DD8020  LDR     r8,[sp,#0x20]
0x0000092C 2501      MOVS    r5,#0x01
0x0000092E F04F0902  MOV     r9,#0x02
0x00000932 2603      MOVS    r6,#0x03
0x00000934 2704      MOVS    r7,#0x04
0x00000936 F04F0C05  MOV     r12,#0x05
0x0000093A 2000      MOVS    r0,#0x00
0x0000093C EB050A09  ADD     r10,r5,r9
0x00000940 44B2      ADD     r10,r10,r6
0x00000942 44BA      ADD     r10,r10,r7
0x00000944 44E2      ADD     r10,r10,r12
0x00000946 44A2      ADD     r10,r10,r4
0x00000948 448A      ADD     r10,r10,r1
0x0000094A 4492      ADD     r10,r10,r2
0x0000094C 449A      ADD     r10,r10,r3
0x0000094E 44C2      ADD     r10,r10,r8
0x00000950 F00A00FF  AND     r0,r10,#0xFF
0x00000954 EBA70A06  SUB     r10,r7,r6
0x00000958 EBAA0C05  SUB     r12,r10,r5
0x0000095C FB90FAFC  SDIV    r10,r0,r12
0x00000960 F00A00FF  AND     r0,r10,#0xFF
0x00000964 E8BD87F0  POP     {r4-r10,pc}
0x00000968 0988      DCW     0x0988
0x0000096A 0000      DCW     0x0000
0x0000096C 0000      DCW     0x0000
0x0000096E 2000      DCW     0x2000
0x00000970 0008      DCW     0x0008
0x00000972 0000      DCW     0x0000
```

寄存器压栈

局部变量设置

加法运算

减法运算

除法运算

函数返回

结合CFSR找到具体错误原因

bit31 to bit16	bit15 to bit8	bit7 to bit0
用法Fault状态寄存器（UFSR）	总线Fault状态寄存器（BFSR）	存储器管理Fault状态寄存器（MMFSR）

SCB->CFSR为 0x02000000

右移16bit后为0x0200

即bit9为1，DIVBYZERO错误，除0错误

即语句 temp5=temp5/temp4; 中，temp4为0。

位	名称	描述
[15:10]	-	保留
[9]	DIVBYZERO	0：无除以零Fault或除以零捕获未使能 1：处理器已执行SDIV或UDIV指令（除以零）。 当处理器将该位设为1时，异常返回的压入堆栈的PC值指向执行除以零的指令。 注：通过将CCR中的DIV_0_TRP位设为1使能除以零捕获，默认是不使能的。
[8]	UNALIGNED	0：无非对齐访问Fault，或非对齐访问捕获未使能 1：处理器已进行了一次非对齐的存储器访问。 注：通过将CCR中的UNALIGN_TRP位设为1来使能非对齐访问捕获，默认是不使能的。非对齐的LDM、STM、LDRD和STRD指令总是出错，与UNALIGN_TRP的设置无关。
[7:4]	-	保留
[3]	NOCP	无协处理器用法Fault。处理器不支持协处理器指令： 0：试图访问一个协处理器未引起用法Fault 1：处理器已试图访问一个协处理器。
[2]	INVPC	EXC_RETURN的无效PC加载引起的无效PC加载用法Fault： 0：没有发生无效PC加载用法Fault 1：处理器已试图将EXC_RETURN非法载入PC，作为一个无效的上下文或一个无效的EXC_RETURN值。 当该位被设为1时，异常返回的压入堆栈的PC值指向尝试执行非法PC加载的指令。
[1]	INVSTATE	无效状态用法Fault： 0：未发生无效状态用法Fault 1：处理器已试图执行一个非法使用EPSR的指令。 当该位设为1时，异常返回的压入堆栈的PC值指向一个尝试非法使用EPSR的指令。 如果一个未定义的指令使用了EPSR，则该位不被置位为1。
[0]	UNDEFINSTR	未定义的指令用法Fault: 0：无未定义的指令用法Fault 1：处理器已试图执行一个未定义的指令。当该位设为1时，异常返回的压入堆栈的PC值指向未定义的指令。 未定义的指令是一条不能被处理器译码的指令。

总结

1.先查看LR(EXC_RETURN),判断栈大小和指针。

2.根据栈内容查找返回地址。

3.分析返回地址处C和汇编代码，确认产生错误的C语句。

4.查看SCB->CFSR寄存器找到错误种类。

5.错误种类结合C语句，确定错误原因。

6.进入异常时，芯片会自动压栈，无需用户操作。

WWW.IOT