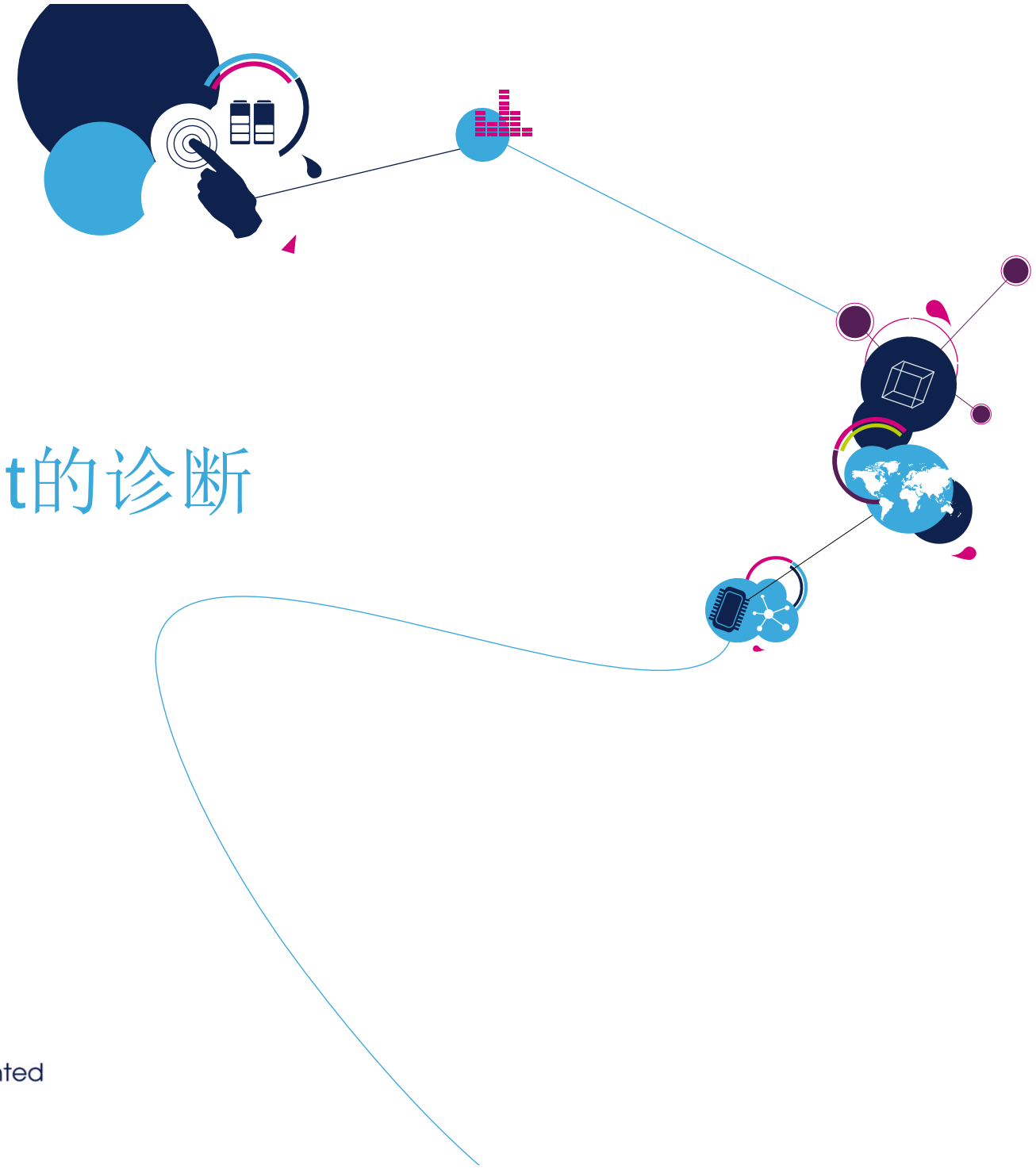

Hard Fault的诊断

life.augmented

Cortex-M3异常模型

异常编号	IRQ编号	异常类型	优先级	备注
1	-	Reset	-3，最高	
2	-14	NMI	-2	永远被使能
3	-13	Hard Fault	-1	Hard Fault
4	-12	MM Fault	可配置	Local Fault
5	-11	Bus Fault		
6	-10	Usage Fault		
7-10	-	-	-	
11	-5	SVCall	可配置	由SVC指令触发
12	-4	DebugMonitor		
13	-	-	-	
14	-2	PendSV	可配置	
15	-1	SysTick		

Fault handler

常见的四种错误异常

- 用法错误（Usage Fault）
  - 执行未定义指令、非对齐操作、除零
  - 复位时默认未使能
- 总线错误（Bus Fault）
  - 取指令、数据读写、堆栈操作
  - 复位时默认未使能
- 存储器管理错误（Memory Management Fault）
  - 违反MPU设定的存储器访问规则
  - 复位时默认未使能
- 硬错误（Hard Fault）
  - 以上异常处理被关闭，而又发生了异常，则触发
  - 执行异常处理时，发生了异常，则触发
  - 复位时默认使能

用法错误异常

- 用法错误异常是与指令译码相关联的处理机制，通常包括以下类错误型：
  - 企图执行未定义指令
  - 企图进入ARM状态
  - 无效的中断返回码
  - 企图执行协处理器指令
  - 使用多重加载/存储指令时地址没对齐
  - ……
- 使能控制
  - SCB->SHCSR. USGFAULTENA @0xE000ED24

何时会意外地试图切入ARM状态:
>> 执行BX Rn 指令时，Rn 的最低位(LSB)为0
>> 异常向量表中的向量为偶数，LSB = 0
>> POP {…，PC}时，弹给PC的值为偶数，LSB = 0

用法错误异常寄存器

- 发生用法错误时，可通过查看寄存器了解出错的原因
  - SCB->CFSR. Usage Fault（UFSR） @0xE000ED2A

位段	名称	类型	含义	备注
9	DIVBYZERO	可读、写1清零	企图执行除0操作（指令：SDIV、UDIV）	使能控制：SCB->CCR.DIV_0_TRP
8	UNALIGNED		企图执行非对齐访问	使能控制：SCB->CCR.UNALIGN_TRP
3	NOCP		企图执行协处理器指令	
2	INVPC		无效的异常返回码	
1	INVSTATE		试图切换到ARM状态	
0	UNDEFINSTR		企图执行未定义指令	

总线错误异常

- 当内核通过AHB接口传送数据时收到了一个错误应答信号，则认总线传输错误，进而发起总线错误异常请求；通常异常来自：
  - 取指令时发生错误，通常称为“预取指流产”
  - 数据读、写时发生错误，也叫“数据流产”
  - 响应中断时，出、入栈发生错误
- 使能控制
  - SCB->SHCSR. BUSFAULTENA @0xE000ED24

AHB总线收到”错误”应答，通常有以下诱因：

>> 企图访问无效的存储器区域，常见于企图访问的地址没有存储器
>> 设备未准备就绪，不能进行总线访问
>> 企图发起的传输宽度不被目标设备所支持
>> 在用户权限下程序企图访问特权模式下的设备

总线错误异常寄存器

- 发生Bus Fault时，可以查看总线错误状态寄存器了解异常的大致类型，供异常处理程序分析
  - SCB->CFSR.Bus Fault（BFSR） @0xE000ED29

位段	名称	类型	含义	备注
7	BFARVALID	可读、写1清零	=1 表示BFAR有效	BFAR包含引起总线异常的指令地址
6:5	--		--	
4	STKERR		入栈时发生错误	Stack
3	UNSTKERR		出栈时发生错误	
2	IMPRECISERR		不精确的数据总线错误	D-Bus
1	PRECISERR		精确的数据总线错误	
0	IBUSERR		指令总线错误	I-Bus

存储器管理错误异常

- 存储器管理错误异常是由于违规访问存储器空间或由某些非法访问引发的；通常包括以下类型：
  - 访问存储区域时违反了MPU的设置规则
  - 越权访问，访问了没有权限访问的地址；
  - 访问了没有存储器的地址
  - 试图从不可执行区域（XN）执行代码
  - 对只读区域进行写操作
- 使能控制
  - SCB->SHCSR. MEMFAULTENA @0xE000ED24

存储器管理错误异常，通常是与MPU相关联的。其诱因往往是违反了MPU设置的访问规则。
某些非法访问，如在不可执行的存储器区域取指，也会会引发该异常，既使系统中没有MPU 或MPU被关掉了

存储器管理错误异常寄存器

- 发生Memory Management Fault时，可以查看存储器管理错误状态寄存器了解异常的类型
  - SCB->CFSR. Memory Manage Fault（MFSR） @0xE000ED28

位段	名称	类型	含义
7	MMARVALID	可读、写1清零	=1 表示MMAR有效
6:5	--		--
4	MSTKERR		入栈时企图访问不被允许的区域
3	MUNSTKERR		出栈时企图访问不被允许的区域
2	--		--
1	DACCVIOL		企图从不允许访问的区域读、写数据
0	IACCVIOL		企图从不允许访问的区域取指令

硬错误异常

- 硬错误异常是应对系统严重错误而设计的，其优先级为-1，仅次于NMI异常。触发硬错误异常的原因有：
  - 调试出错
  - 前面三个常规错误异常不能及时响应
  - 为响应中断，取中断向量时发生错误
  - 仍可通过PRIMASK关闭

注意：

>> 当总线错误异常、存储器管理错误异常、用法错误异常无法及时得到响应时或未被使能时，系统将产生硬错异常。

>> 当硬错误异常服务例程又引发新的硬错误异常时，系统将进入死锁状态，只能由复位使其退出

硬错误异常寄存器

- 硬错误状态寄存器中记录有产生硬错误异常的诱因
  - SCB->HFSR @0xE000ED2C

位段	名称	类型	含义
31	DEBUGEVT	可读、写1清零	因调试事件而产生
30	FORCED		由总线错误、存储器管理错误、用法错误提升而来的硬错误
29:2	--		--
1	VECTBL		取中断向量时出错
0	--		--

调试异常寄存器

- 由于调试原因引发的硬错误，可以通过调试错误状态寄存器查找其诱因
  - SCB->DFSR @0xE000ED30

位段	名称	类型	含义
31:5	--	可读、写1清零	--
4	EXTERNAL		外部调试请求
3	VCATCH		发生向量捕获
2	DWTTRAP		数据监测点匹配
1	BKPT		执行BKPT指令
0	HALTED		NVIC停机请求

错误异常发生时

- 上下文（Stack Frame）被入栈保存
  - R0~R3, R12, LR,PC,xPSR
- 上下文保存在哪个堆栈中？由此时的LR决定
  - LR = EXC_RETURN
  - LR.2 = 1，保存在Main Stack，由MSP指向
  - LR.2 = 0，保存在Process Stack，由PSP指向

记录发生异常时的现场数据

- 采集现场有助于分析出错原因。可以通过仿真器查看，也可以通过串行通信口打印方式获得，如下面这段代码

```
HardFault_Handler:
  TST LR, #4
  ITE EQ
  MRSEQ R0, MSP
  MRSNE R0, PSP
  B hard_fault_handler_c
```

```
void hard_fault_handler_c (unsigned int * hardfault_args)
 {
   unsigned int stacked_r0;
   unsigned int stacked_r1;
   unsigned int stacked_r2;
   unsigned int stacked_r3;
   unsigned int stacked_r12;
   unsigned int stacked_lr;
   unsigned int stacked_pc;
   unsigned int stacked_psr;

   stacked_r0 = ((unsigned long) hardfault_args[0]);
   stacked_r1 = ((unsigned long) hardfault_args[1]);
   stacked_r2 = ((unsigned long) hardfault_args[2]);
   stacked_r3 = ((unsigned long) hardfault_args[3]);
   stacked_r12 = ((unsigned long) hardfault_args[4]);
   stacked_lr = ((unsigned long) hardfault_args[5]);
   stacked_pc = ((unsigned long) hardfault_args[6]);
   stacked_psr = ((unsigned long) hardfault_args[7]);
```

异常处理程序样例（续）


```
printf ("\n\n[Hard fault handler - all numbers in hex]\n");
printf ("R0 = %x\n", stacked_r0);
printf ("R1 = %x\n", stacked_r1);
printf ("R2 = %x\n", stacked_r2);
printf ("R3 = %x\n", stacked_r3);
printf ("R12 = %x\n", stacked_r12);
printf ("LR [R14] = %x subroutine call return address\n", stacked_lr);
printf ("PC [R15] = %x program counter\n", stacked_pc);
printf ("PSR = %x\n", stacked_psr);

printf ("BFAR = %x\n", (*((volatile unsigned long *)(0xE000ED38))));
printf ("CFSR = %x\n", (*((volatile unsigned long *)(0xE000ED28))));
printf ("HFSR = %x\n", (*((volatile unsigned long *)(0xE000ED2C))));
printf ("DFSR = %x\n", (*((volatile unsigned long *)(0xE000ED30))));
printf ("AFSR = %x\n", (*((volatile unsigned long *)(0xE000ED3C))));

printf ("SCB_SHCSR = %x\n", SCB->SHCSR); while (1);

}
```