ARM Cortex-M3处理器故障的分析与处理

阿城继电器股份有限公司　孙洪蛟

【摘要】本文介绍了ARM Cortex-M3处理器的4种故障：总线故障、用法故障、内存管理故障和硬故障。分析了这些故障产生的原因，叙述了如何通过故障状态寄存器找出故障原因，如何在程序开发阶段尽可能的避免故障的产生，以及故障的处理方法。

【关键词】ARM；故障；寄存器；中断

1.引言

在嵌入式领域，ARM Cortex -M3处理器凭借其高性能、低功耗、低成本等优势，得到了广泛的应用。该处理器具有很强的灵活性，在给开发人员较大开发空间的同时，也在一定程度上增加了开发的难度。尤其是当处理器出现故障时，故障原因常常难以找出。本文针对该处理器的以上特点，详细分析了处理器的4种故障，使读者可以对故障进行准确的分析和处理。

2.故障分析

ARM Cortex-M3处理器[1]共有4种故障：总线故障、用法故障、内存管理故障和硬故障。ARM将故障当作特殊的中断来处理，其中，硬故障在所有故障中拥有最高优先级，优先级为-1，其它故障的优先级是可整定的，但必须为非负数，默认优先级是0（ARM中优先级的数值越小，优先级的级别越高，负数拥有最高优先级[2]）。总线故障是指指令或数据在AHB总线上传输时出现错误而产生的一种故障，可在指令读取、数据读写以及堆栈的压栈和弹栈时产生。用法故障是指在对CPU的使用上出现错误而导致的故障。内存管理故障是指对内存的读写违反了MPU（内存保护单元）的规定或在不允许执行指令的地址执行指令而产生的一种故障。硬故障一般由其它故障升级导致。下面将对每种故障进行详细分析。

2.1 总线故障

总线故障产生的原因有以下几种：读写错误的内存地址（如对一处内存地址进行读写，但该地址并没有连接内存）；对一个设备进行读写，但该设备还没有准备好接受读写操作(如读写外部RAM，但外部RAM还没有初始化)；对设备读写的数据类型不符合设备的要求（如某设备只支持32位读写，当对该设备进行8位读写时，便会出现故障）；设备由于某种原因无法接受对它的读写操作[3]（如在非特权模式下对一个只接受特权模式操作的设备进行读写）。如果想使能总线故障处理，将SCB（系统控制模块）中SHCSR寄存器中的BUSFAULTENA位置1。如果将中断向量表放入RAM中，在置位之前应先确定总线故障处理程序的的初始地址已经在表中设置完毕。总线故障发生后，SCB中的BFSR寄存器提供了关于故障的相关信息，如表1所示：

表1 总线故障状态寄存器（BFSR）

位	名称	缺省值	描述
7	BFARVALID	0	BFAR中的值有效
6:5	-	-	-
4	STKERR	0	压栈错误
3	UNSTKERR	0	弹栈错误
2	IMPREISERR	0	不精确的数据操作错误
1	PRECISERR	0	精确的数据操作错误
0	IBUSERR	0	指令操作错误

表2 用法故障状态寄存器（UFSR）

位	名称	缺省值	描述
9	DIVBYZERO	0	除数为0
8	UNALIGNED	0	不对齐地址操作
7:4	-	-	-
3	NOCP	0	试图执行从处理器指令
2	INVPC	0	EXC_RETURN（中断返回值）错误
1	INVSTATE	0	试图转换到ARM状态
0	UNDEFINSTR	0	执行未定义指令

如果BFARVALID位置1，则SCB中BFAR寄存器中的值便代表了产生总线故障所进行读写操作的内存地址。当总线故障产生时，如果故障处理使能，且此时没有相同或更高优先级的中断在运行，总线故障的处理程序将被执行。如果故障处理使能，但此时有相同或更高优先级的中断正在运行，总线故障将处于等待状态。如果故障处理未使能，或故障出现在相同或更高优先级的中断程序中，则总线故障的处理程序无法被执行，于是总线故障升级为硬故障。

2.2 用法故障

用法故障产生的原因有以下几种：使用未定义的指令；使用协处理器指令（Cortex-M3不支持协处理器）；试图从Thumb状态转换到ARM状态；中断返回值错误（LR寄存器中含有错误的返回值）；在使用STM或LDM指令时，内存地址未对齐；除以0或读写未对齐的内存地址（可通过对NVIC——可嵌套向量中断控制器相应寄存器的控制位选择是否产生故障）。如果想使能用法故障处理，将SCB中SHCSR寄存器中的USGFAULTENA位置1。如果将中断向量表放入RAM中，在置位之前先确定用法故障处理程序的的初始地址已经在表中设置完毕。用法故障发生后，SCB中的UFSR寄存器提供了关于故障的相关信息，如表2所示：

与总线故障相同，当用法故障产生时，如果故障处理使能，则在没有相同或更高优先级中断运行时，用法故障处理程序被执行，否则处于等待状态。如果故障处理未使能，或故障出现在相同或更高优先级的中断程序中，则升级为硬故障。

2.3 内存管理故障

内存管理故障产生的原因有以下几种：所读写的内存地址超出了MPU的设置范围；对只读区域进行写操作；在非特权状态下对只支持特权读写的区域（由MPU规定）进行读写操作。如果想使能内存管理故障处理，将SCB中SHCSR寄存器中的MEMFAULTENA位置1。如果将中断向量表放入RAM中，在置位之前应先确定内存管理故障处理程序的的初始地址已经在表中设置完毕。内存管理故障发生后，SCB中的MMSR寄存器提供了关于故障的相关信息，

如表3所示：

如果MMARVALID位置1，则SCB中MMAR寄存器中的值便代表了产生内存管理故障所进行读写操作的内存地址。与总线故障相同，当内存管理故障产生时，如果故障处理使能，则在没有相同或更高优先级中断运行时，内存管理故障处理程序被执行，否则处于等待状态。如果故障处理未使能，或故障出现在相同或更高优先级的中断程序中，则升级为硬故障。

2.4 硬故障和系统锁定

硬故障是一种比较特殊的故障，它一般在出现了上述故障，而由于某种原因上述故障的处理程序无法执行，或在故障处理程序中又产生了故障，或出现了某种用其它方式无法解决的故障时产生。当产生中断，对中断向量表中的向量进行读取时，如果出现了总线故障，或中断向量表中的向量不正确，硬故障也会产生。硬故障还可由调试引起。硬故障产生后，SCB中的HFSR寄存器提供了关于故障的相关信息，如表4所示：

当在硬故障或NMI（不可屏蔽中断，优先级为-2）的处理程序中产生了故障，或在重启后对SP、PC的读取中出现总线故障，系统会进入锁定状态。如果程序在一开始就不运行，可能是中断向量表被放在了错误的内存地址中，或向量的0位没有置1。如果程序运行一些指令后突然崩溃，则有可能是由于大端小端的设置出错或堆栈指针设置得不正确。

3.故障处理

在程序开发过程中和现场运行中对故障处理有着不同的要求。在程序开发过程中主要是找出产生故障的原因，使开发人员及时修正。而在现场运行中主要是使系统恢复到正常的工作状态[4]。

在程序开发过程中，为了尽量避免故障的产生，应将堆栈空间设置得足够大，如果使用操作系统，则应保证每个任务的堆栈都有足够的空间，以避免程序中出现错误改变堆栈中的数据。如果软件系统比较简单，可以在故障处理程序中报告相关的故障信息，如果软件系统比较复杂，则应将故障信息存储在内存中的某个位置，然后使用PendSV来报告故障信息。

故障处理程序应尽可能的简单，以避免在故障处理程序中产生新的故障。在故障的处理程序中应尽可能的避免不必要的压栈操作，在必要的压栈操作之前可以先检查堆栈指针是否正确，如果堆栈指针正确，再进行压栈操作。在硬故障和NMI处理程序中最好只使用R0-R3寄存器，以及R12寄存器，以避免在硬故障和NMI处理程序中进行压栈操作而产生总线故障使程序进入锁定状态。

需要报告的故障信息包括以下几种：

(1)故障状态寄存器：BFSR、UFSR、MMSR、HFSR、DFSR和AFSR。根据芯片制造商的不同实施方案，AFSR寄存器可能并不存在，此时可不考虑该寄存器。

(2)堆栈中的数据：被压栈的PC、LR、PSR寄存器以及R0-R3、R12寄存器的值。在读取堆栈中的数据之前，应先通过LR寄存器中的bit[2]确定产生故障的程序所用的堆栈是MSP还是PSP。

(3)故障地址寄存器：BFAR和MMAR。对故障地址寄存器操作时，应按以下步骤进行：首先，读取BFAR/MMAR寄存器，而后读取BFSR/MMSR中的BFARVALID/MMARVALID位，如果上述位为0，则说明故障地址寄存器中的值不准确，将其丢弃，最后，清除BFARVALID/MMARVALID位。若不按上述步骤进行，而先读取FARVALID/MMARVALID位后读取故障地址寄存器，则如果在这两次操作之间，执行了优先级更高的故障处理程序，故障地址寄存器中的值可能被改动，当程序返回到原始的故障处理程序后，会误以为故障地址寄存器中的值没有变化而产生误判。

现场运行中对故障处理的常用的方法有以下几种：

1）关闭任务：如果使用了操作系统，可以关闭产生故障的任务，并在需要的时候重新启动该任务。

2）修复：在某些情况下，产生的故障可以被修复（如执行从处理器指令后，可以在用法故障中用软件模拟从处理器）。

3）重启：在非操作系统环境下，这是最常用的方法。将SCB中AIRCR寄存器中的SYSRESETREQ位置1，从而将整个系统重启。如果系统进入了锁定状态，PC寄存器中的值将固定为0xFFFFFFFX，并将从这一地址反复读取指令，此时Cortex-M3核心将会输出一个LOCKUP信号，芯片制造商可以根据这个信号触发中断。如果芯片制造商对LOCKUP信号未作处理，也可通过设置独立看门狗将系统重启。

4.结语

本文详细的分析了ARM Cortex-M3处理器的4种故障——总线故障、用法故障、内存管理故障和硬故障的产生原因、特点及查找方法，并叙述了在程序开发过程中和现场运行中故障处理的方法。

表3 内存管理故障状态寄存器（MMSR）

位	名称	缺省值	描述
7	MMARVALID	0	MMAR中的值有效
6:5	-	-	-
4	MSTKERR	0	压栈错误
3	MUNSTKERR	0	弹栈错误
2	-	-	-
1	DACCVIOL	0	数据操作错误
0	IACCVIOL	0	指令操作错误

表4 硬故障状态寄存器（HFSR）

位	名称	缺省值	描述
31	DEBUGEVT	0	硬故障由调试引起
30	FORCED	0	硬故障由其它故障升级产生
29:2	-	-	-
1	VECTBL	0	读中断向量表时产生故障
0	-	-	-

参考文献

[1]Cortex-M3 Technical Reference Manual. http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337i/DDI0337I_cortexm3_r2p1_trm.pdf

[2]Cortex-M3 Embedded Software Development. http://infocenter.arm.com/help/topic/com.arm.doc.dai0179b/AppsNote179.pdf

[3]Cortex-M3 Devices Generic User Guide. http://infocenter.arm.com/help/topic/com.arm.doc.dui0552a/DUI0552A_cortex_m3_dgug.pdf

[4]Joseph Yiu. The Definite Guide to the ARM Cortex-M3. Elsevier Inc.

作者简介：孙洪蛟（1984—），男，大学本科，助理工程师，现供职于阿城继电器股份有限公司，主要从事基于ARM Cortex-M3处理器的嵌入式软件平台的研发工作。

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证
7. VPX 总线技术及其实现
8. 基于 Xilinx FPGA 的 PCIE 接口实现
9. 基于 PCI 总线的 GPS 授时卡设计
10. 基于 CPCI 标准的 6U 信号处理平台的设计
11. USB30 电路保护
12. USB30 协议分析与框架设计
13. USB 30 中的 CRC 校验原理及实现
14. 基于 CPLD 的 UART 设计
15. IPMI 在 VPX 系统中的应用与设计
16. 基于 CPCI 总线的 PMC 载板设计
17. 基于 VPX 总线的工件台运动控制系统研究与开发
18. PCI Express 流控机制的研究与实现
19. UART16C554 的设计
20. 基于 VPX 的高性能计算机设计
21. 基于 CAN 总线技术的嵌入式网关设计
22. Visual C 串行通讯控件使用方法与技巧的研究
23. IEEE1588 精密时钟同步关键技术研究
24. GPS 信号发生器射频模块的一种实现方案
25. 基于 CPCI 接口的视频采集卡的设计
26. 基于 VPX 的 3U 信号处理平台的设计
27. 基于 PCI Express 总线 1394b 网络传输系统 WDM 驱动设计
28. AT89C52 单片机与 ARINC429 航空总线接口设计
29. 基于 CPCI 总线多 DSP 系统的高速主机接口设计
30. 总线协议中的 CRC 及其在 SATA 通信技术中的应用
31. 基于 FPGA 的 SATA 硬盘加解密控制器设计
32. Modbus 协议在串口通讯中的研究及应用
33. 高可用的磁盘阵列 Cache 的设计和实现
34. RAID 阵列中高速 Cache 管理的优化

35. 一种新的基于 RAID 的 CACHE 技术研究与实现
36. 基于 PCIE-104 总线的高速数据接口设计
37. 基于 VPX 标准的 RapidIO 交换和 Flash 存储模块设计
38. 北斗卫星系统在海洋工程中的应用
39. 北斗卫星系统在远洋船舶上应用的研究
40. 基于 CPCI 总线的红外实时信号处理系统
41. 硬件实现 RAID 与软件实现 RAID 的比较
42. 基于 PCI Express 总线系统的热插拔设计
43. 基于 RAID5 的磁盘阵列 Cache 的研究与实现

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计
12. VxWorks 的内存配置和管理
13. 基于 VxWorks 系统的 PCI 配置与应用
14. 基于 MPC8270 的 VxWorks BSP 的移植
15. Bootrom 功能改进经验谈
16. 基于 VxWorks 嵌入式系统的中文平台研究与实现
17. VxBus 的 A429 接口驱动
18. 基于 VxBus 和 MPC8569E 千兆网驱动开发和实现
19. 一种基于 vxBus 的 PPC 与 FPGA 高速互联的驱动设计方法
20. 基于 VxBus 的设备驱动开发
21. 基于 VxBus 的驱动程序架构分析
22. 基于 VxBus 的高速数据采集卡驱动程序开发
23. Vxworks 下的冗余 CAN 通讯模块设计
24. WindML 工业平台下开发 S1d13506 驱动及显示功能的实现

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道
7. Linux 串口编程实例
8. 基于 Yocto Project 的嵌入式应用设计
9. Android 应用的反编译
10. 基于 Android 行为的加密应用系统研究
11. 嵌入式 Linux 系统移植步步通
12. 嵌入式 CC++语言精华文章集锦
13. 基于 Linux 的高性能服务器端的设计与研究
14. S3C6410 移植 Android 内核
15. Android 开发指南中文版
16. 图解 Linux 操作系统架构设计与实现原理（第二版）
17. 如何在 Ubuntu 和 Linux Mint 下轻松升级 Linux 内核
18. Android 简单 mp3 播放器源码
19. 嵌入式 Linux 系统实时性的研究
20. Android 嵌入式系统架构及内核浅析
21. 基于嵌入式 Linux 操作系统内核实时性的改进方法研究
22. Linux TCP IP 协议详解
23. Linux 桌面环境下内存去重技术的研究与实现
24. 掌握 Android 7.0 新增特性 Quick Settings
25. Android 应用逆向分析方法研究
26. Android 操作系统的课程教学
27. Android 智能手机操作系统的研究
28. Android 英文朗读功能的实现
29. 基于 Yocto 订制嵌入式 Linux 发行版
30. 基于嵌入式 Linux 的网络设备驱动设计与实现
31. 如何高效学习嵌入式
32. 基于 Android 平台的 GPS 定位系统的设计与实现

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究

4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码
7. Windows 下的 USB 设备驱动程序开发
8. WinCE 的大容量程控数据传输解决方案设计
9. WinCE6.0 安装开发详解
10. DOS 下仿 Windows 的自带计算器程序 C 源码
11. G726 局域网语音通话程序和源代码
12. WinCE 主板加载第三方驱动程序的方法
13. WinCE 下的注册表编辑程序和源代码
14. WinCE 串口通信源代码
15. WINCE 的 SD 卡程序[可实现读写的源码]
16. 基于 WinCE 的 BootLoader 研究
17. Windows CE 环境下无线网卡的自动安装
18. 基于 Windows CE 的可视电话的研究与实现

PowerPC：

1. Freescale MPC8536 开发板原理图
2. 基于 MPC8548E 的固件设计
3. 基于 MPC8548E 的嵌入式数据处理系统设计
4. 基于 PowerPC 嵌入式网络通信平台的实现
5. PowerPC 在车辆显控系统中的应用
6. 基于 PowerPC 的单板计算机的设计
7. 用 PowerPC860 实现 FPGA 配置
8. 基于 MPC8247 嵌入式电力交换系统的设计与实现
9. 基于设备树的 MPC8247 嵌入式 Linux 系统开发
10. 基于 MPC8313E 嵌入式系统 UBoot 的移植
11. 基于 PowerPC 处理器 SMP 系统的 UBoot 移植
12. 基于 PowerPC 双核处理器嵌入式系统 UBoot 移植
13. 基于 PowerPC 的雷达通用处理机设计
14. PowerPC 平台引导加载程序的移植

ARM：

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码
7. STM32 SD 卡移植 FATFS 文件系统源码
8. STM32 ADC 多通道源码
9. ARM Linux 在 EP7312 上的移植
10. ARM 经典 300 问
11. 基于 S5PV210 的频谱监测设备嵌入式系统设计与实现
12. Uboot 中 start.S 源码的指令级的详尽解析
13. 基于 ARM9 的嵌入式 Zigbee 网关设计与实现
14. 基于 S3C6410 处理器的嵌入式 Linux 系统移植
15. CortexA8 平台的μC-OSⅡ及 LwIP 协议栈的移植与实现
16. 基于 ARM 的嵌入式 Linux 无线网卡设备驱动设计
17. ARM S3C2440 Linux ADC 驱动
18. ARM S3C2440 Linux 触摸屏驱动
19. Linux 和 Cortex-A8 的视频处理及数字微波传输系统设计
20. Nand Flash 启动模式下的 Uboot 移植
21. 基于 ARM 处理器的 UART 设计

Hardware:

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计
5. 高速 PCB 信号完整性分析及应用
6. DM642 高速图像采集系统的电磁干扰设计
7. 使用 COMExpress Nano 工控板实现 IP 调度设备
8. 基于 COM Express 架构的数据记录仪的设计与实现
9. 基于 COM Express 的信号系统逻辑运算单元设计
10. 基于 COM Express 的回波预处理模块设计
11. 基于 X86 平台的简单多任务内核的分析与实现
12. 基于 UEFI Shell 的 PreOS Application 的开发与研究
13. 基于 UEFI 固件的恶意代码防范技术研究
14. MIPS 架构计算机平台的支持固件研究
15. 基于 UEFI 固件的攻击验证技术研究

16. 基于 UEFI 的 Application 和 Driver 的分析与开发

Programming:

1. 计算机软件基础数据结构 - 算法