



(12) 发明专利申请

(10) 申请公布号 CN 116430835 A

(43) 申请公布日 2023. 07. 14

(21) 申请号 202310695106.7

(22) 申请日 2023.06.13

(71) 申请人 力高(山东)新能源技术股份有限公司

地址 264000 山东省烟台市经济技术开发区长江路300号内8号501室

(72) 发明人 张明龙 王云 姜明军 孙艳  
江梓贤 刘欢

(74) 专利代理机构 合肥中谷知识产权代理事务所(普通合伙) 34146

专利代理师 杜瑶

(51) Int. Cl.

G05B 23/02 (2006.01)

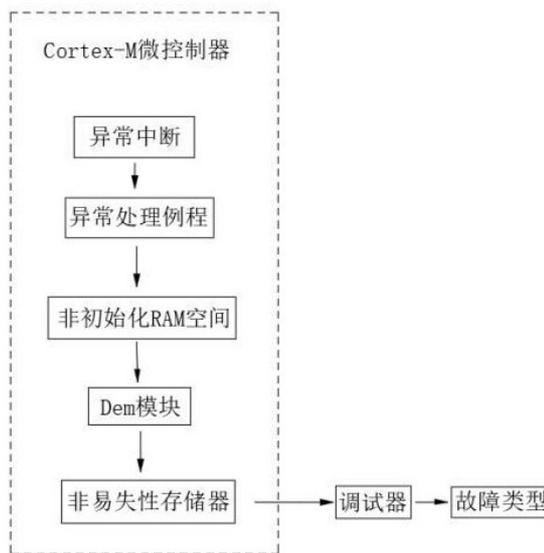
权利要求书2页 说明书5页 附图1页

(54) 发明名称

一种Cortex-M微控制器的故障存储与分析方法

(57) 摘要

本发明属于汽车电子嵌入式实时系统技术领域内的一种Cortex-M微控制器的故障存储与分析方法,包括在微控制器发生故障触发异常中断时调用异常处理例程收集故障数据,并将故障数据存储到非初始化RAM空间中,微控制器执行系统软件复位和初始化后对故障数据执行信息校验,调用Dem模块将完成校验的故障数据转存到Dem模块的Ram空间中,并在清零非初始化RAM空间后将完成校验的故障数据存储到非易失性存储器;本发明通过定向收集微控制器故障相关的数据,结合调试器和包含调试信息的.elf文件,可以快速地定位故障时的汇编语句,及引发微控制器故障的相关模块,能快速有效地分析出引起微控制器故障的原因。



1. 一种Cortex-M微控制器的故障存储与分析方法,其特征在于,包括如下步骤:

S1:在Cortex-M微控制器发生故障触发异常中断时,Cortex-M微控制器主动调用预设的异常处理例程定向收集故障数据,并将所述故障数据存储到Cortex-M微控制器内非初始化RAM空间中;

其中,故障数据包括异常编号、CPU进入异常前的堆栈空间数据以及故障状态下寄存器数据;

S2:Cortex-M微控制器执行系统软件复位并初始化,在初始化完成后对非初始化RAM空间中所述故障数据执行信息校验,Cortex-M微控制器调用Dem模块中的Dem\_SetEventStatus函数将完成校验的故障数据转存到Dem模块的Ram空间中,并在清零非初始化RAM空间后,Dem模块调用NvM\_WriteBlock函数将完成校验的故障数据存储到非易失性存储器中;

S3:通过调试器加载所述故障数据中elf文件以开启调试界面,在调试界面中分析CPU进入异常前的堆栈空间数据中的PC寄存器值,定位到Cortex-M微控制器在进入异常中断前的汇编语句并反向推导末条汇编,根据末条汇编手动输入PC寄存器、LR寄存器、R12寄存器、R3寄存器、R2寄存器、R1寄存器以及R0寄存器的值以还原故障时Cortex-M微控制器上下文数据,结合异常编号和故障状态下寄存器数据,反向推导Cortex-M微控制器的故障原因。

2. 根据权利要求1所述的一种Cortex-M微控制器的故障存储与分析方法,其特征在于:步骤S1中异常中断包括Cortex-M微控制器内部总线故障异常中断、存储器管理故障异常中断、用法故障异常中断和硬故障异常中断。

3. 根据权利要求1所述的一种Cortex-M微控制器的故障存储与分析方法,其特征在于:步骤S1中故障数据存储到非初始化RAM空间中的同时,Cortex-M微控制器通过CAN总线CAN\_ID(0x00)向外部平台输出Cortex-M微控制器异常的故障数据。

4. 根据权利要求1所述的一种Cortex-M微控制器的故障存储与分析方法,其特征在于:步骤S1中异常处理例程包括:Cortex-M微控制器主动收集所述故障数据,并置Cortex-M微控制器内“微控制器故障标志位”为有效,并在所述故障数据中添加CRC32校验信息后将所述故障数据存储到非初始化RAM空间。

5. 根据权利要求4所述的一种Cortex-M微控制器的故障存储与分析方法,其特征在于:步骤S2中,Cortex-M微控制器初始化完成后,信息校验过程为:

S2.1:对非初始化RAM空间中的故障数据进行CRC32校验信息检查以判断故障数据是否完整,若判断结果为完整则执行步骤S2.2;

S2.2:对非初始化RAM空间中“微控制器故障标志位”是否有效进行判断,若判断结果为有效则调用Dem模块对故障数据进行转存。

6. 根据权利要求1所述的一种Cortex-M微控制器的故障存储与分析方法,其特征在于:Cortex-M微控制器中,定向收集的故障数据中异常编号及其对应的故障类型包括:

编号3对应硬故障;编号4对应存储器管理故障;编号5对应内部总线故障;编号6对应用法故障。

7. 根据权利要求6所述的一种Cortex-M微控制器的故障存储与分析方法,其特征在于:Cortex-M微控制器中,故障数据中故障状态下寄存器数据与异常编号的对应关系为:

若异常编号为3,需提取“硬故障状态寄存器”的数据,寄存器地址0xE00\_ED2C,有效长

度32bit;

若异常编号为4,需提取“存储器管理故障状态寄存器”的数据,寄存器地址0xE000\_ED28,有效长度8bit;

若异常编号为5,需提取“内部总线故障”的数据,寄存器地址0xE000\_ED29,有效长度8bit;

异常编号为6,需提取“用法故障”的数据,寄存器地址0xE000\_ED2A,有效长度9bit。

## 一种Cortex-M微控制器的故障存储与分析方法

### 技术领域

[0001] 本发明属于汽车电子嵌入式实时系统技术领域,具体涉及一种Cortex-M微控制器的故障存储与分析方法。

### 背景技术

[0002] AUTOSAR,全称为Automotive Open System Architecture,即汽车开放系统架构。它是由全球各家汽车制造商、零部件供应商以及各种研究、服务机构共同参与的一种汽车电子系统的合作开发框架,并建立了一个开放的汽车控制器(ECU)标准软件架构;AUTOSAR架构下的汽车ECU中Cortex-M微控制器包括确认故障发生的Dem模块,用于诊断事件管理模块收集故障状态字节和故障诊断快照数据,Dem模块调用NvM模块(非易失性存储模块),请求存储故障诊断快照数据。

[0003] 现有技术中汽车ECU是基于嵌入式系统方案,在遇到Cortex-M微控制器内核的故障时,如程序异常跑飞、程序死机崩溃等问题时,即无法正常地记录故障,也无法有效地定位故障;具体体现在如下两个方面:

(1)Cortex-M微控制器内核故障时,无法正常运行Dem\NvM等模块,导致无法正常地记录故障;

(2)Cortex-M微控制器内核故障后,系统死机,之后看门狗复位、系统重启,故障现场的上下文信息丢失,如若内核故障无法复现,该故障无法有效地定位。

### 发明内容

[0004] 本发明的目的就在于提供一种Cortex-M微控制器的故障存储与分析方法,以解决背景技术中提出的问题。

[0005] 本发明通过以下技术方案来实现上述目的:

一种Cortex-M微控制器的故障存储与分析方法,包括如下步骤:

S1:在Cortex-M微控制器发生故障触发异常中断时,Cortex-M微控制器主动调用预设的异常处理例程定向收集故障数据,并将所述故障数据存储到Cortex-M微控制器内非初始化RAM空间中;

其中,故障数据包括异常编号、CPU进入异常前的堆栈空间数据以及故障状态下寄存器数据;

S2:Cortex-M微控制器执行系统软件复位并初始化,在初始化完成后对非初始化RAM空间中所述故障数据执行信息校验,Cortex-M微控制器调用Dem模块中的Dem\_SetEventStatus函数将完成校验的故障数据转存到Dem模块的Ram空间中,并在清零非初始化RAM空间后,Dem模块调用NvM\_WriteBlock函数将完成校验的故障数据存储到非易失性存储器中;

S3:通过调试器加载所述故障数据中elf文件以开启调试界面,在调试界面中分析CPU进入异常前的堆栈空间数据中的PC寄存器值,定位到Cortex-M微控制器在进入异常中

断前的汇编语句并反向推导末条汇编,根据末条汇编语句手动输入PC寄存器、LR寄存器、R12寄存器、R3寄存器、R2寄存器、R1寄存器以及R0寄存器的值以还原故障时Cortex-M微控制器上下文数据,结合异常编号和故障状态下寄存器数据,反向推导Cortex-M微控制器的故障原因。

[0006] 作为本发明的进一步优化方案,步骤S1中异常中断包括Cortex-M微控制器内部总线故障异常中断、存储器管理故障异常中断、用法故障异常中断和硬故障异常中断。

[0007] 作为本发明的进一步优化方案,步骤S1中故障数据存储到非初始化RAM空间中的同时,Cortex-M微控制器通过CAN总线CAN\_ID(0x00)向外部平台输出Cortex-M微控制器异常的故障数据。

[0008] 作为本发明的进一步优化方案,步骤S1中异常处理例程包括:Cortex-M微控制器主动收集所述故障数据,并置Cortex-M微控制器内“微控制器故障标志位”为有效,并在所述故障数据中添加CRC32校验信息后将所述故障数据存储到非初始化RAM空间。

[0009] 作为本发明的进一步优化方案,步骤S2中,Cortex-M微控制器初始化完成后,信息校验过程为:

S2.1:对非初始化RAM空间中的故障数据进行CRC32校验信息检查以判断故障数据是否完整,若判断结果为完整则执行步骤S2.2;

S2.2:对非初始化RAM空间中“微控制器故障标志位”是否有效进行判断,若判断结果为有效则调用Dem模块对故障数据进行转存。

[0010] 作为本发明的进一步优化方案,Cortex-M微控制器中,定向收集的故障数据中异常编号及其对应的故障类型包括:

编号3对应硬故障;编号4对应存储器管理故障;编号5对应内部总线故障;编号6对应用法故障。

[0011] 作为本发明的进一步优化方案,Cortex-M微控制器中,故障数据中故障状态下寄存器数据与异常编号的对应关系为:

若异常编号为3,需提取“硬故障状态寄存器”的数据,寄存器地址0xE000\_ED2C,有效长度32bit;

若异常编号为4,需提取“存储器管理故障状态寄存器”的数据,寄存器地址0xE000\_ED28,有效长度8bit;

若异常编号为5,需提取“内部总线故障”的数据,寄存器地址0xE000\_ED29,有效长度8bit;

异常编号为6,需提取“用法故障”的数据,寄存器地址0xE000\_ED2A,有效长度9bit。

[0012] 本发明的有益效果在于:

本发明实现了汽车ECU出现微控制器故障而崩溃死机时,仍旧可以记录微控制器故障相关的数据;

本发明通过定向收集微控制器故障相关的数据,结合调试器和包含调试信息的.elf文件,可以快速地定位故障时的汇编语句,及引发微控制器故障的相关模块,进而快速有效地分析出引起微控制器故障的原因;

本发明中,在进入微控制器故障异常中断时,将微控制器故障相关的数据存储到

非初始化RAM中,复位后再存储相应的数据,解决了微控制器崩溃死机时无法正常存储故障数据的问题;

本发明中,Cortex-M微控制器故障时的相关数据,包括异常编号、CPU进入异常前的堆栈空间数据、相应异常对应的故障状态寄存器的数据,以上数据对于分析微控制器故障的原因非常有效;

本发明中,Cortex-M微控制器故障时的相关数据结合调试器和包括调试信息的.elf文件,快速还原故障时的现场,快速有效地定位微控制器故障的真正原因。

## 附图说明

[0013] 图1是本发明的整体流程框图。

## 具体实施方式

[0014] 下面结合附图对本发明作进一步详细描述,有必要在此指出的是,以下具体实施方式只用于对本发明进行进一步的说明,不能理解为对本发明保护范围的限制,该领域的技术人员可以根据上述发明内容对本发明作出一些非本质的改进和调整。

## 实施例1

[0015] 如图1所示,本发明提供了一种Cortex-M微控制器的故障存储与分析方法,包括如下步骤:

S1:在Cortex-M微控制器发生故障触发异常中断时,Cortex-M微控制器主动调用预设的异常处理例程定向收集故障数据,并将所述故障数据存储到Cortex-M微控制器内非初始化RAM空间中;

其中,故障数据包括异常编号、CPU进入异常前的堆栈空间数据以及故障状态下寄存器数据。

[0016] S2:Cortex-M微控制器执行系统软件复位并初始化,在初始化完成后对非初始化RAM空间中所述故障数据执行信息校验,Cortex-M微控制器调用Dem模块中的Dem\_SetEventStatus函数将完成校验的故障数据转存到Dem模块的Ram空间中,并在清零非初始化RAM空间后,Dem模块调用NvM\_WriteBlock函数将完成校验的故障数据存储到非易失性存储器中;

其中,Dem模块,即诊断事件管理模块收集故障状态字节、故障诊断快照数据,Dem模块的Ram空间是用于存储诊断事件(错误)和相关的信息,Cortex-M微控制器的非初始化RAM空间是用于存储Cortex-M微控制器内的flash数据。

[0017] S3:通过调试器加载所述故障数据中elf文件以开启调试界面,在调试界面中分析CPU进入异常前的堆栈空间数据中的PC寄存器值,定位到Cortex-M微控制器在进入异常中断前的汇编语句并反向推导末条汇编语句,根据末条汇编语句手动输入PC寄存器、LR寄存器、R12寄存器、R3寄存器、R2寄存器、R1寄存器以及R0寄存器的值以还原故障时Cortex-M微控制器上下文数据,结合异常编号和故障状态下寄存器数据,反向推导Cortex-M微控制器的故障原因。

[0018] 进一步的,Cortex-M微控制器的内核故障可分成内部总线故障,存储器管理故

障,用法故障和硬故障;其中,内部总线故障主要是指数据访问、取指访问、入栈和出栈时发生的错误;存储器管理故障主要是指违反了存储器保护的要求而进行的非法访问;用法故障主要是指内核尝试使用非法的汇编指令;硬故障主要是指内部总线故障、存储器管理故障及用法故障。

[0019] 当汽车ECU中Cortex-M微控制器出现内核故障时,对于故障信息的存储过程包括:当汽车ECU出现内核故障时,会触发系统内核异常中断(内部总线故障异常中断,存储器管理故障异常中断,用法故障异常中断和硬故障异常中断),并在异常中断中调用异常处理例程。

[0020] 进一步的,步骤S1中异常处理例程包括:Cortex-M微控制器主动收集故障数据,并置Cortex-M微控制器内“微控制器故障标志位”为有效,并在故障数据中添加CRC32校验信息后将故障数据存储到非初始化RAM空间。

[0021] 在Cortex-M微控制器异常处理例程中,程序会主动定向收集异常编号、CPU进入异常前的堆栈空间数据以及故障状态下的寄存器数据等,置“微控制器故障标志位”为有效,并添加CRC32的校验信息,将以上所有的数据存储到“非初始化RAM空间”中。

[0022] 微控制器异常处理例程在结束时调用Cortex-M微控制器软件复位,Cortex-M微控制器复位后会重新初始化,待初始化完成后,会检查“非初始化RAM空间”,先做CRC32的校验检查,确认数据的完整性和一致性,再检查“微控制器故障标志位”是否为有效

若“微控制器故障标志位”为有效,系统会调用Dem模块中的Dem\_SetEventStatus函数,会将“非初始化RAM空间”的故障数据转存到Dem模块的Ram空间中,并清零“非初始化RAM空间”。

[0023] Dem模块进一步地调用NvM\_WriteBlock函数,将故障数据存储到非易失存储介质中。

[0024] 进一步的,步骤S1中故障数据存储到非初始化RAM空间中的同时,Cortex-M微控制器通过CAN总线CAN\_ID(0x00)向外部平台输出Cortex-M微控制器异常的故障数据。

[0025] 进一步的,步骤S2中,Cortex-M微控制器初始化完成后,信息校验过程为:

S2.1:对非初始化RAM空间中的故障数据进行CRC32校验信息检查以判断故障数据是否完整,若判断结果为完整则执行步骤S2.2;

S2.2:对非初始化RAM空间中“微控制器故障标志位”是否有效进行判断,若判断结果为有效则调用Dem模块对故障数据进行转存。

[0026] 进一步的,Cortex-M微控制器中,定向收集的故障数据中异常编号及其对应的故障类型包括:

编号3对应硬故障;编号4对应存储器管理故障;编号5对应内部总线故障;编号6对应用法故障。

[0027] 需要说明的是,如何定位Cortex-M微控制器内核异常,目前没有普遍有效的方法,针对上述缺陷,本发明中技术方案结合Cortex-M的特殊架构,说明了应当提取微控制器异常时具体的故障数据,发生Cortex-M内核异常时,会有大量的上下文信息,本发明挑选了最关键有效的Cortex-M内核数据,具有一定代表性且故障定位准确性较高。

[0028] 具体的,在Cortex-M微控制器发生异常中断后,需要通过进入异常例程时堆栈指针SP反推出进入异常前的上下文数据,并做相应的提取。包括进入异常中断前的xPSR寄存

器值、PC寄存器值、LR寄存器值、R12寄存器值、R3寄存器值、R2寄存器值、R1寄存器值、R0寄存器值。

[0029] 进一步的,Cortex-M微控制器中,故障数据中故障状态下寄存器数据与异常编号的对应关系为:

若异常编号为3,需提取“硬故障状态寄存器”的数据,寄存器地址0xE000\_ED2C,有效长度32bit;

若异常编号为4,需提取“存储器管理故障状态寄存器”的数据,寄存器地址0xE000\_ED28,有效长度8bit;

若异常编号为5,需提取“内部总线故障”的数据,寄存器地址0xE000\_ED29,有效长度8bit;

异常编号为6,需提取“用法故障”的数据,寄存器地址0xE000\_ED2A,有效长度9bit。

[0030] 对于微控制器的故障分析方法,具体包括如下:

结合微控制器故障的相关数据,调试器、包含调试信息的可执行文件.elf文件,分析导致微控制器故障的原因。

[0031] 通过诊断仪读取微控制器故障时的相关数据。

[0032] 调试器加载.elf文件,开启调试界面。

[0033] 分析“CPU进入异常前的堆栈空间的数据”中的PC寄存器值,定位到Cortex-M微控制器在进入微控制器异常中断前的最后一条汇编语句。

[0034] 通过调试器定位到“进入微控制器异常中断前的最后一条汇编语句”,反向推导最后一条C语句。

[0035] 通过调试器手工输入PC寄存器、LR寄存器、R12寄存器、R3寄存器、R2寄存器、R1寄存器、R0寄存器,还原故障时的微控制器上下文数据。

[0036] 结合“异常编号”、“相应异常对应的故障状态寄存器的数据”,反向推导可能的引起微控制器异常的原因。

[0037] 在实际的故障定位处理中,首先通过收集“CPU进入异常前的堆栈空间的数据”中的PC寄存器值,定位到微控制器死机前的最后一条语句,若最后一条语句对应异常编号为5,可得到为“内部总线故障”,同时其故障状态寄存器为0x04,表示为“不精确的数据访问违例”,最终分析出微控制器故障的原因是对一个不可写的地址范围进行了写操作。

[0038] 本实施方式中,当出现不精确的数据访问冲突时,需要结合程序的执行流,往前追踪几条或十几条汇编语句,逐条分析,以便于定位真正出错的汇编语句。

[0039] 需要说明的是,上述方案是适用于汽车ECU在前期开发、样车过程中容易出现微控制器内核故障的场景下,以及售后小概率性出现微控制器内核故障时,难以发现,难以定位的问题。

[0040] 以上所述实施例仅表达了本发明的几种实施方式,其描述较为具体和详细,但不能因此而理解为对本发明范围的限制。应当指出的是,对于本领域的普通技术人员来说,在不脱离本发明构思的前提下,还可以做出若干变形和改进,这些都属于本发明的保护范围。

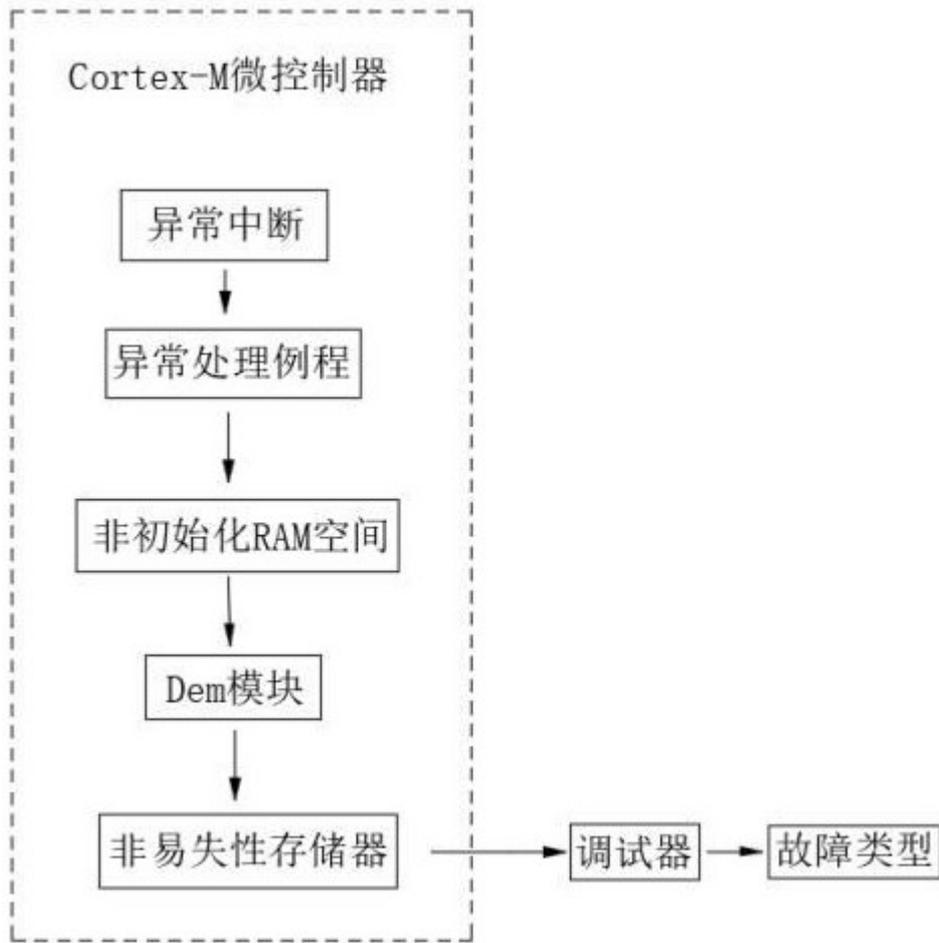


图 1