



(12) 发明专利申请

(10) 申请公布号 CN 103927187 A

(43) 申请公布日 2014. 07. 16

(21) 申请号 201410195480. 1

(22) 申请日 2014. 05. 09

(71) 申请人 成都凯智科技有限公司  
地址 610041 四川省成都市高新区科园二路  
一号

(72) 发明人 郭辉 罗彬 覃树建

(74) 专利代理机构 北京天奇智新知识产权代理  
有限公司 11340  
代理人 杨春

(51) Int. Cl.  
G06F 9/44 (2006. 01)

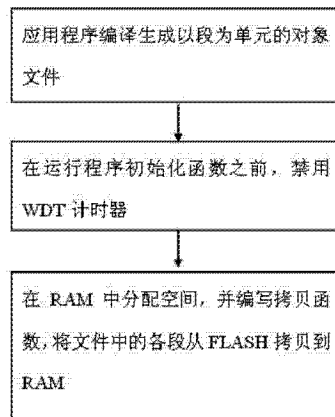
权利要求书1页 说明书5页 附图1页

(54) 发明名称

嵌入式系统程序执行方法

(57) 摘要

本发明提供了一种嵌入式系统程序执行方法,该方法包括:应用程序编译生成以段为单元的对象文件;在运行程序初始化函数之前,禁用WDT 计时器;在RAM 中分配空间,并编写拷贝函数,将文件中的各段从FLASH 拷贝到RAM。本发明将整个嵌入式系统的应用程序都拷贝到RAM 中执行,保证了程序运行的实时性。



1. 一种嵌入式系统程序执行方法,用于在嵌入式系统中执行应用程序,其特征在于,包括:

应用程序编译生成以段为单元的对象文件;

在运行程序初始化函数之前,禁用 WDT 计时器;

在 RAM 中分配空间,并编写拷贝函数,将文件中的各段从 FLASH 拷贝到 RAM。

2. 根据权利要求 1 所述的方法,其特征在于,所述文件以段的形式组织代码和数据,分为赋值段和未赋值段,其中在嵌入式系统上电时,赋值段含有真实有效的参数,未赋值段用于保留程序运行过程中所需开辟的变量的地址空间,在程序运行时,所有的赋值段链接到非易失性存储器中,未赋值段的代码链接到 RAM 中,

所述赋值段包括以下内容段: .text、.cinit、.pinit、.switch、

.const、.econst, 其中 .text、.cinit、.pinit、.switch 段为程序空间,.const、.econst 段为数据空间,

所述未赋值段包括以下内容段: .bss、.ebss、.stack、.system、.esystem,均为数据空间。

3. 根据权利要求 2 所述的方法,其特征在于,所述嵌入式系统基于 TMS320C6455 微处理器,对该微处理器汇编成各自的目标代码段,然后用链接器将其链接起来生成可执行文件,

上电后,程序起始标号的长跳转指令被加载到入口地址处并开始执行,接着指针跳转到 WDT 计时器禁用代码,之后再执行拷贝函数,将所有段复制到 RAM 中,所述长跳转指令及 WDT 计时器禁用代码部分分别置于相应的段下,并组合成汇编文件,添加到工程文件的 source 路径下,

并且,拷贝函数可将代码段及各个变量段从 FLASH 拷贝到 RAM,应用程序在复制时,通过链接命令文件中指明的信息调用段的起始地址、长度以及运行地址,同时将拷贝函数链接到代码拷贝段中。

4. 根据权利要求 3 所述的方法,其特征在于,当所有的段都被复制完毕时,通过跳转指令跳转到程序初始化函数,直接进入主函数执行。

5. 根据权利要求 3 所述的方法,其特征在于,在将段进行拷贝时,首先将该段的大小先暂存到寄存器 XAR5,再存储到累加器 ACC 中;将加载地址存储到 XAR6 寄存器中;运行地址存储到 XAR7 寄存器中,再执行 copy 操作即可完成该段的拷贝。

6. 根据权利要求 3 所述的方法,其特征在于,所述链接命令文件分为两部分,第一部分用于指明和定义装载代码所用到的所有存储区域即程序空间和数据空间,以便链接器进行配置;第二部分主要是将每一个输出段映射到所分配的存储空间里面去,并指定段的加载地址和运行地址,

对于赋值段,加载和运行分配到不同的存储空间,对未赋值段的加载和运行只需分配到 RAM 空间,并且在用户代码从 FLASH 拷贝到片内 RAM 之前,将 WDT 计时器禁用段加载和运行于 FLASH 中。

## 嵌入式系统程序执行方法

### 技术领域

[0001] 本发明涉及嵌入式系统程序执行方法。

### 背景技术

[0002] 随着信息技术的发展和行业的需求,智能交通系统(ITS)越来越备受关注。近年来,ITS在城市交通管理方面得到了普遍应用。其中,嵌入式系统在实时、准确、高效的综合交通运输管理系统中起到决定性作用。例如,嵌入式系统中使用专门用于实时数字信号处理的微处理器,其必须具备功能强大,适合做密集计算,拥有强大的运算能力。以TMS320C6455为例,它是专门用于实时数字信号处理的微处理器,采用了多总线结构、流水线技术和专用指令等特殊设计。但这种处理器的时钟频率是有限的,这样运行的程序特别是复杂的代码必须要经过优化,才能完成实时性的要求。

[0003] 在一个独立的嵌入式系统中,所有的代码必须存放在非易失性存储器中。对于TMS320C6455来讲,由于CPU对片内FLASH读写访问时需要插入一定数目的等待周期,所以,应用程序在FLASH中是无法按照CPU的指令执行速率全速运行的。程序在片内FLASH中的运行速度有限,而如果生成的可执行代码加载至片内RAM中,则能够全速运行,真正实现无等待读写访问。因此,在代码运行前,将其从FLASH中拷贝到RAM中运行,能够显著提高程序的运行效率。

[0004] 然而现有的技术文档只提供了部分代码从FLASH拷贝到RAM中的手段。然而,在一些应用领域中,如智能交通中的视频监控等实时性要求较高的领域,需要将整个应用程序都拷贝到RAM中执行,以提高代码整体运行速度。

[0005] 因此,针对相关技术中所存在的上述问题,目前尚未提出有效的解决方案。

### 发明内容

[0006] 为解决上述现有技术所存在的问题,本发明提出了一种嵌入式系统程序执行方法,用于在嵌入式系统中执行应用程序,包括:

[0007] 应用程序编译生成以段为单元的对象文件;

[0008] 在运行程序初始化函数之前,禁用WDT计时器;

[0009] 在RAM中分配空间,并编写拷贝函数,将文件中的各段从FLASH拷贝到RAM。

[0010] 优选地,所述文件以段的形式组织代码和数据,分为赋值段和未赋值段,其中在嵌入式系统上电时,赋值段含有真实有效的参数,未赋值段用于保留程序运行过程中所需开辟的变量的地址空间,在程序运行时,所有的赋值段链接到非易失性存储器中,未赋值段的代码链接到RAM中,

[0011] 所述赋值段包括以下内容段: .text、.cinit、.pinit、.switch、

[0012] .const、.econst,其中.text、.cinit、.pinit、.switch段为程序空间,.const、.econst段为数据空间,

[0013] 所述未赋值段包括以下内容段: .bss、.ebss、.stack、.system、.esystem,均为数

据空间。

[0014] 优选地,所述嵌入式系统基于 TMS320C6455 微处理器,对该微处理器汇编成各自的目标代码段,然后用链接器将其链接起来生成可执行文件,

[0015] 上电后,程序起始标号的长跳转指令被加载到入口地址处并开始执行,接着指针跳转到 WDT 计时器禁用代码,之后再执行拷贝函数,将所有段复制到 RAM 中,所述长跳转指令及 WDT 计时器禁用代码部分分别置于相应的段下,并组合成汇编文件,添加到工程文件的 source 路径下,

[0016] 并且,拷贝函数可将代码段及各个变量段从 FLASH 拷贝到 RAM,应用程序在复制时,通过链接命令文件中指明的信息调用段的起始地址、长度以及运行地址,同时将拷贝函数链接到代码拷贝段中。

[0017] 优选地,当所有的段都被复制完毕时,通过跳转指令跳转到程序初始化函数,直接进入主函数执行。

[0018] 优选地,在将段进行拷贝时,首先将该段的大小先暂存到寄存器 XAR5,再存储到累加器 ACC 中;将加载地址存储到 XAR6 寄存器中;运行地址存储到 XAR7 寄存器中,再执行 copy 操作即可完成该段的拷贝。

[0019] 优选地,所述链接命令文件分为两部分,第一部分用于指明和定义装载代码所用到的所有存储区域即程序空间和数据空间,以便链接器进行配置;第二部分主要是将每一个输出段映射到所分配的存储空间里面去,并指定段的加载地址和运行地址,

[0020] 对于赋值段,加载和运行分配到不同的存储空间,对未赋值段的加载和运行只需分配到 RAM 空间,并且在用户代码从 FLASH 拷贝到片内 RAM 之前,将 WDT 计时器禁用段加载和运行于 FLASH 中。

[0021] 相比于现有技术,本发明的技术方案具有以下优点:将整个应用程序都拷贝到 RAM 中执行,以提高代码整体运行速度。

## 附图说明

[0022] 图 1 是根据本发明实施例的嵌入式系统程序执行方法流程图。

## 具体实施方式

[0023] 多种方式可以用于(包括实施为过程;装置;系统;物质组成;在计算机可读存储介质上包括的计算机程序产品;和/或处理器(诸如如下处理器,该处理器被配置成执行在耦合到处理器的存储器上存储的和/或由该存储器提供的指令))实施本发明。在本说明书中,这些实施或者本发明可以采用的任何其他形式可以称为技术。一般而言,可以在本发明的范围内变更公开的过程的步骤顺序。除非另有明示,描述为被配置成执行任务的部件(诸如处理器或者存储器)可以实施为被临时配置成在给定时间执行该任务的一般部件或者被制造成执行该任务的具体部件。

[0024] 下文与图示本发明原理的附图一起提供对本发明一个或者多个实施例的详细描述。结合这样的实施例描述本发明,但是本发明不限于任何实施例。本发明的范围仅由权利要求书限定,并且本发明涵盖诸多替代、修改和等同物。在下文描述中阐述诸多具体细节以便提供对本发明的透彻理解。出于示例的目的而提供这些细节,并且无这些具体细节中

的一些或者所有细节也可以根据权利要求书实现本发明。

[0025] 本发明的目的在于提供一种嵌入式应用程序的执行方法。片内 FLASH 程序的引导启动流程一般为,程序需要系统独立地运行时,要求将应用程序固化到非易失性存储器 FLASH 中,系统每次上电复位后,便开始 FLASH 程序引导流程,经过引导,最终跳转到应用程序入口地址处。

[0026] BootLoader 是位于 BootROM 中的用于系统引导的程序,可完成芯片复位后的自动引导功能。系统复位后,PC 指针跳转到指定地址处并获得复位向量,该向量在芯片出厂前已固化好,将程序流的执行重新定位到引导初始化函数,从而开始引导过程。

[0027] 引导初始化函数主要把器件初始化成 C28X 工作模式,然后读取安全代码模块的密码,完成对 FLASH、RAM 等片内存储器的解锁操作;接着 Bootloader 将调用引导模式选择函数,检测相应 GPIO 引脚的电平状态,判断为 FLASH 引导模式。引导结束后 PC 指针将跳转至 FLASH 中起始地址单元处,并执行其中的代码,用户需要在该地址存放一条指令。由于安全代码模块密码从起始地址开始,所以只有 2 个字的的空间可用于存放跳转指令,而一个长跳转指令(在汇编代码中为 LB)刚好占用 2 个字的地址空间。

[0028] 一般情况下,执行跳转指令将会跳转到执行程序的 C 环境初始化函数 int00,只有当 int00 运行后,才开始执行主函数。其中,int00 是 C 程序的入口点,主要实现全局和静态变量的初始化,即将 .cinit 段(存放用来对变量初始化的常数)复制到片内 RAM 中的 .ebss 段(为变量分配地址空间)来实现的。当用户程序中包含大量的已初始化全局和静态变量时,在程序初始化函数执行结束并调用主函数前,WDT 计时器计数器可能溢出。在程序调试阶段,由于将 .cinit 段链接到 RAM 中,上述拷贝操作相当于是从 RAM 到 RAM,速度是比较快的,这个问题一般不会出现。但在系统应用阶段,用户代码需要固化到 FLASH 存储器中,.cinit 段链接至片内 FLASH,而从 FLASH 中复制数据需要占用多个时钟周期,速度较慢,WDT 计时器可能会溢出,导致复位。所以本发明在运行程序初始化函数之前先禁用 WDT 计时器,然后执行至主函数时,根据需要使能或继续禁用 WDT 计时器的状态。

[0029] 本发明通过向工程源文件中添加拷贝程序,引导结束后对其执行,由于程序复制阶段,WDT 计时器不能使能。因此,从 FLASH 引导之后,经过禁用 WDT 计时器操作后,“拷贝程序”才可被执行,FLASH 空间的代码会被拷贝到 RAM 中。最后,PC 指针指向 RAM 中存放代码的首地址处开始执行主程序,从而程序就在 RAM 中运行起来。应用程序调试成功后,将程序固化在 FLASH,上电后自动将 FLASH 中程序拷贝到 RAM,并在 RAM 中运行。

[0030] 图 1 是根据本发明实施例的应用程序执行方法流程图。如图 1 所示,实施本发明的具体步骤如下:

[0031] 步骤一:应用程序编译生成段空间形式的文件,该文件包括赋值段和未赋值段,其中赋值段用于提供有效参数,未赋值段用于预留程序运行时所需的变量地址空间。

[0032] 应用程序经过编译、链接后生成通用对象文件。该文件以段的形式组织代码和数据,可分为赋值段和未赋值段。在嵌入式系统上电时,赋值段含有真实有效的参数,例如指令代码和常量。未赋值段用于保留程序运行过程中所需开辟的变量的地址空间,在上电调用初始化库前,未赋值段并没有真实的内容。

[0033] 当彻底脱离仿真环境运行程序时,所有的赋值段必须链接到非易失性存储器中,通常是片上 FLASH。未赋值段不需要初始化值,如变量,在程序运行的时候,代码会频繁地对

其进行读写和实时修改。因此,未赋值段必须链接到 RAM 中。同样,用户创建的赋值段必须存放到 FLASH;未赋值段必须存放在 RAM 中。

[0034] 在一个实施例中,赋值段包括 .text、.cinit、.pinit、.switch、

[0035] .const、.econst 段。而 .text、.cinit、.pinit、.switch 段为程序空间,.const、.econst 段为数据空间。.bss、.ebss、.stack、.system、.esystem 为未赋值段,这些未赋值段均为数据空间。

[0036] 步骤二:在运行程序初始化函数之前,禁用 WDT 计时器,并利用拷贝函数,用于各个变量段从 FLASH 拷贝到 RAM。

[0037] 仍以 TMS320C6455 为例,C 编译器允许对其独立编写 C 及汇编程序,并分开编译或汇编成各自的目标代码段,然后用链接器将其链接起来生成可执行文件。在工程中,汇编源代码用简单的汇编或伪指令编写,用于定位段、指针跳转等。为了能够实现上述启动功能,最简单的方法是使用汇编代码。

[0038] 上电后,程序起始标号的长跳转指令被加载到入口地址处并开始执行,接着指针跳转到 WDT 计时器禁用代码,之后再执行拷贝函数,将所有段复制到 RAM 中。将跳转指令及 WDT 计时器禁用代码部分分别置于相应的段下,并组合成汇编文件,添加到工程文件的 source 路径下。

[0039] 拷贝函数可将代码段及各个变量段从 FLASH 拷贝到 RAM,以实现程序的整体拷贝。

[0040] 以赋值段 text 为例来详述设计过程。程序在复制时可通过链接命令文件中指明的信息调用段的起始地址、长度以及运行地址。该例程中,将 text 段的大小先暂存到寄存器 XAR5,再存储到累加器 ACC 中;加载地址存储到 XAR6 寄存器中;运行地址存储到 XAR7 寄存器中,再执行 copy 操作即可完成 text 段的拷贝。

[0041] 同时应将拷贝函数链接到代码拷贝段中,其他的赋值段“.cinit、const、econst、pinit、switch”也按照类似的方法去实现,一旦所有的段都被复制完毕,可通过一个指令跳转到程序初始化函数。

[0042] 完成上面流程后,C 编译环境被设置好后便可直接进入主函数去执行。

[0043] 步骤三:编写链接命令文件,在 RAM 中分配空间,将应用程序段拷贝到 RAM 中。

[0044] 将链接命令文件分为两部分。第一部分用于指明和定义装载代码所用到的所有存储区域,即程序空间和数据空间,以便链接器进行配置;第二部分主要是将每一个输出段映射到所分配的存储空间里面去,并指定段的加载地址和运行地址。

[0045] 3.1 考虑到赋值段加载在 FLASH 中而运行在片上 RAM 中,因此对这些段的加载和运行必须分配到不同的存储空间。以初始化程序代码段“.text”为例,在链接命令文件第二部分中按照如下步骤,便可实现段的拷贝:

[0046] 将 .text 段装载至程序空间的 FLASH\_AB;

[0047] 将 .text 运行于程序空间的 RAM;

[0048] 确定 .text 段拷贝的起始地址、运行地址和长度。

[0049] 同理,其他赋值段也可按照相似的方法去配置,来指明段的加载和运行信息。

[0050] 3.2 未赋值段的加载和运行均在片上 RAM 中,所以对 these 段的加载和运行只需分配到 RAM 空间即可。以未赋值段“.bss”为例,在链接命令文件第二部分中按照如下命令配置即可。

[0051] .bss :> RAM\_L3PAGE = 1

[0052] 3.3 在用户代码从 FLASH 拷贝到片内 RAM 之前, WDT 计时器禁用段必须加载和运行于 FLASH 中。

[0053] 优选地, 如果应用程序代码较大, 而 RAM 空间有限, 使得用户代码无法完全加载至片内 RAM 中, 通常可将那些需要频繁调用、实时性要求高的代码拷贝到片内 RAM 中运行, 或者外部扩展高速的片外 RAM。

[0054] 综上所述, 本发明将整个嵌入式系统的应用程序都拷贝到 RAM 中执行, 保证了程序运行的实时性, 适用于交通视频监控等高可用需求的应用领域。

[0055] 显然, 本领域的技术人员应该理解, 上述的本发明的各模块或各步骤可以用通用的计算系统来实现, 它们可以集中在单个的计算系统上, 或者分布在多个计算系统所组成的网络上, 可选地, 它们可以用计算系统可执行的程序代码来实现, 从而, 可以将它们存储在存储系统中由计算系统来执行。这样, 本发明不限制于任何特定的硬件和软件结合。

[0056] 应当理解的是, 本发明的上述具体实施方式仅仅用于示例性说明或解释本发明的原理, 而不构成对本发明的限制。因此, 在不偏离本发明的精神和范围的情况下所做的任何修改、等同替换、改进等, 均应包含在本发明的保护范围之内。此外, 本发明所附权利要求旨在涵盖落入所附权利要求范围和边界、或者这种范围和边界的等同形式内的全部变化和修改例。

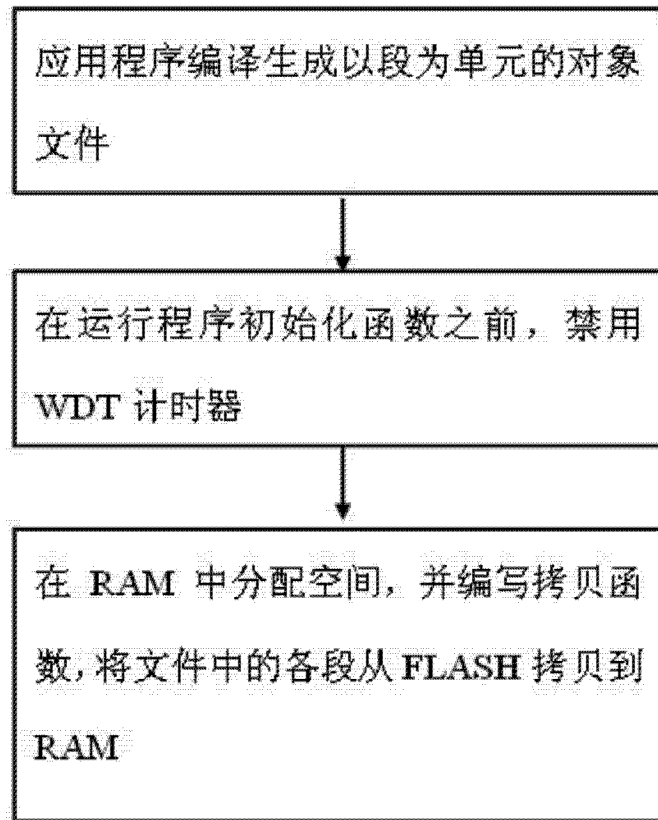


图 1