

ARM7 嵌入式系统中启动程序的实现

王京林 岳春生 苏洁

(解放军信息工程大学 郑州 450002)

[摘要]: 本文给出了基于 ARM7 嵌入式系统的启动程序的实现流程,并针对存储器控制单元的使用以及目标文件的分布装载等技术难点进行详细分析。

[关键字]: 嵌入式系统、启动程序、ARM7

[Abstract]: This article has provided the realization flow of ARM7 startup process in embedded system. Two technical difficulties were analysed detailedly: How to use MMU to manage memory map and how to scatter loading the image file.

[Keywords]: Embedded system, Startup process, ARM7

嵌入式系统被定义为:以应用为中心、以计算机技术为基础、软件硬件可裁剪、适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。嵌入式系统的核心部件是各种类型的嵌入式处理器,随着嵌入式系统不断深入到人们生活中的各个领域,嵌入式处理器得到前所未有的飞速发展。

典型的 32 位 RISC 芯片——ARM 处理器,不论是在 PDA, STB, DVD 等消费类电子产品中,还是在 GPS, 航空, 勘探, 测量等军方产品中都得到了广泛的应用。越来越多的芯片厂商早已看好 ARM 的前景,如 Intel, NS, Atmel, Philips, NEC, CirrusLogic 等公司都有相应的产品。在 1999 年,ARM 突破 1.5 亿个,市场份额超过了 50%,已经成为业界的龙头。

在我们研制开发基于 ARM7 的嵌入式系统过程中,发现技术难点主要在于系统启动程序的编写,为此本文详细论述了在 ARM7 基础上开发嵌入式系统时启动程序的实现。

1. 启动程序流程

嵌入式系统的资源有限,程序通常都是固化在 ROM 中运行。ROM 中程序执行前,需要对系统硬件和软件运行环境进行初始化,这些工作由用汇编语言编写的启动程序完成。

启动程序是嵌入式程序的开头部分,应与应用程序一起固化在 ROM 中,并首先在系统上运行。它应包含进各模块中可能出现的所有段类,并合理安排它们的次序。

2. 详细步骤

(1) 设置入口指针

启动程序首先必须定义入口指针,而且整个应用程序只有一个入口指针。

(2) 设置中断向量

ARM7 要求中断向量表必须设置在从 0 地址开始,连续 8×4 字节的空间,分别是复位、未定义指令错误、软件中断、预取指令错误、数据存取错误、IRQ、FIQ 和一个保留的中断向量。

如果 ROM 定位于 0 地址,向量表包含一系列指令跳转到中断服务程序,否则向量必须被动态初始化。可以在启动程序中添加一段代码,使其在运行时将向量表拷贝到 0 地址开始的存储器空间。

对于各未用中断,使其指向一个只含返回指令的哑函数,以防止错误中断引起系统的混乱。

(3) 初始化堆栈和寄存器

系统堆栈初始化取决于用户使用了哪些中断,以及系统需要处理哪些错误类型。一般来说管理者堆栈必须设置,如果使用了 IRQ 中断,则 IRQ 堆栈也必须设置。

如果系统使用了 DRAM 或其它外设,需要设置相关的寄存器,以确定其刷新频率,数据总线宽度等信息。

(4)初始化存储器系统

有些芯片可通过寄存器编程初始化存储器系统,而对于较复杂系统通常集成有 MMU 来管理内存空间。

(5) 如有必要改变处理器模式、状态

如果系统应用程序是运行在用户模式下,可在此处将系统改为用户模式并初始化用户堆栈指针。

(6)初始化 C 语言所需的存储器空间。

为正确运行应用程序,在初始化期间应将系统需要读写的数据和变量从 ROM 拷贝到 RAM 里;一些要求快速响应的程序,如中断处理程序,也需要在 RAM 中运行;如果使用 FLASH,对 FLASH 的擦除和写入操作也一定要在 RAM 里运行。ARM 公司软件开发工具包中的链接器提供了分布装载功能,可以实现这一目的。

(7)呼叫 C 程序。

ARM 有两种指令集:16 位 THUMB 指令集和 32 位 ARM 指令集。使用 16 位的存储器可以降低成本,在这种情况下,Thumb 指令集的整体执行速度比 ARM 32 位指令集快,而且提高了代码密度,所以一般用 Thumb 编译器将 C 语言程序编译成 16 位的代码。处理器一开始总在 arm 状态,可使用 BX 指令转换到 thumb 状态呼叫 C 程序。要注意的是用 C 语言编写嵌入式程序时,要避免使用不能被固化到 ROM 中的库函数。

3. 技术难点分析

(1) MMU 的使用

MMU 是存储器管理单元的缩写,是用来管理虚拟内存系统的器件。MMU 通常是 CPU 的一部分,本身有少量存储空间存放从虚拟地址到物理地址的匹配表。此表称作 TLB(转换旁置缓冲区)。所有数据请求都送往 MMU,由 MMU 决定数据是在 RAM 内还是在大容量存储器设备内。如果数据不在存储空间内,MMU 将产生页面错误中断。

MMU 的两个主要功能是:1、将虚地址转换成物理地址。2、控制存储器存取允许。MMU 关掉时,虚地址直接输出到物理地址总线。

在实践中,使用 MMU 解决了如下几个问题:

①使用 DRAM 作为大容量存储器时,如果 DRAM 的行列是非平方的,会导致该 DRAM 的物理地址不连续,这将给程序的编写调试造成极大不便,而适当配置 MMU 可将其转换成虚拟地址连续的空间。

②ARM 内核的中断向量表要求放在 0 地址,对于 ROM 在 0 地址的情况,无法调试中断服务程序,所以在调试阶段有必要将可读写的存储器空间映射到 0 地址。

③系统的某些地址段是不允许被访问的,否则会产生不可预料的后果,为了避免这类错误,可以通过 MMU 匹配表的设置将这些地址段设为用户不可存取类型。

启动程序中生成的匹配表中包含地址映射,存储页大小(1M,64K,或 4K)以及是否允许存取等信息。

例如:目标板上的 16 兆 DRAM 的物理地址区间为 0xc000,0000~0xc07f,ffff 和 0xc100,0000~0xc17f,ffff;16 兆 ROM 的虚拟地址区间为:0x0000,0000~0x00ff,ffff。匹配表配置如下:

```
0x0000, 0000 0xc000, 0000 r/w
0x0010, 0000 0xc010, 0000 ...
0x0020, 0000 0xc020, 0000 ...
.....
0x0070, 0000 0xc070, 0000 r/w
```

```

0x0080, 0000 0xc100, 0000 ...
.....
0x00f0, 0000 0xc170, 0000 ...
0x0100, 0000 0x0000, 0000 ro
0x0110, 0000 0x0010, 0000
0x01f0, 0000 0x00f0, 0000
0x0200, 0000 0x0200, 0000 inaccessible
.....

```

可以看到左边是连续的虚拟地址空间，右边是不连续的物理地址空间，而且将 DRAM 映射到了 0 地址区间。MMU 通过虚拟地址和页面表位置信息，按照转换逻辑获得对应物理地址，输出到地址总线上。

应注意到的是使能 MMU 后，程序继续运行，但是对于程序员来说程序计数器的指针已经改变，指向了 ROM 所对应的虚拟地址。

(2) 目标文件的分布装载分析

首先创建一个文本文件，称为分布装载描述文件。它为应用程序的各部分指定装载区间和执行区间。

举例如下：

```

FLASH 0x01000000 0x011ffff ; 2M FLASH
{
FLASH 0x01000000
{
boot.o(BOOT, +First)
* (+RO)
}
DRAM 0x00000000
{
vector.0(VECTOR, +First)
int_handler.o (+RO)
* (+RW, +ZI)
}
}

```

在 ARM 链接器的命令行里加入”-scov description-file -scf”或”-scatter description-file”，编译链接后，将产生一个分布装载文件。

链接器同时产生一组符号，给出每个分布描述文件中命名的区间的长度，装载地址和执行地址。由于链接器和 C 库都没有将代码从它的装载区间拷贝到执行区间，或创建一个零初始化区域的功能，所以要由应用程序员利用这组符号产生的信息完成这项工作，这是在呼叫 C 程序之前必须完成的，举例如下：

```

LDR r0, = |Load$$DRAM$$Base|
LDR r1, = |Image$$DRAM$$Base|
CMP r0, r1 ; 检查装载地址和执行地址是否相同
BEQ do_zi_init ; 相同，则不拷贝该区间，初始化零数据区
MOV r2, r1 ; 不相同，将装载区拷贝到执行区
LDR r4, = |Image$$DRAM$$length|
ADD r2, r2, r4

```

```
BL copy
do_zi_init
LDR r1, =|Image$$DRAM$$ZI$$Base|
MOV r2, r1
LDR r4, =|Image$$DRAM$$ZI$$length|
ADD r2, r2, r4
MOV r3, #0
BL zi_init ; 调用零初始化子程序
```

结语:

本文介绍的启动程序已经在以 Cirrus Logic 公司的 EP7211 和 Ateml 公司的 AT91M40400 开发的系统上运行并测试通过。今后可以在这一基础上添加串行通信模块和 FLASH 操作模块, 开发系统监控程序, 从而实现应用程序的在线升级。

[参考书目]

- [1] 《嵌入式微处理器及其应用开发》 姜桥 罗蕾 计算机世界第四十三期
- [2] 《面向二十一世纪的嵌入式系统综述》 吕京建 肖海桥
中国单片机公共实验室(BOL)
- [3] 《EP7211 数据手册》
- [4] 《ARM7TDMI 数据手册》

本文内容来自互联网, 著作权归原作者所有。由电子零件城 (<http://www.epcity.com/>) 整理并制作成 PDF 文件, 仅供个人学习之用, 不得用于任何商业目的, 否则后果自负。如果您认为本 PDF 文件侵犯了您的任何权利, 请来信 epcity@epcity.com 通知, 本站立即删除。

搜集整理: 电子零件城-笨笨兔 (QQ: 154502842) 2004-03-28