计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(5):117−122 [doi: 10.15888/j.cnki.csa.007406]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

实时操作系统 mbedOS 的移植方法①

刘长勇[1,2,3], 王宜怀[2], 彭　涛[2], 孙亚军[2], 程宏玉[2]

[1](武夷学院 数学与计算机学院, 武夷山 354300)
[2](苏州大学 计算机科学与技术学院, 苏州 215006)
[3](认知计算与智能信息处理福建省高校重点实验室, 武夷山 354300)
通讯作者: 刘长勇, E-mail: 121903852@qq.com

摘　要: MbedOS 是 ARM 公司于 2014 年开始推出的一款面向智能终端与物联网节点的实时操作系统, 主要用于对响应时间有较高实时性要求的嵌入式系统.在深入分析 mbedOS 的基本功能、调度机制、延时函数机制、任务间通信机制等基础上, 以可移植的 mbedOS 工程框架为基础, 分析移植的共性问题, 给出具体的移植方法. 在此基础上, 实现了 mbedOS 在 ARM Cortex-M 系列的不同内核及不同 MCU 上的移植, 还给出了不同开发环境间移植共性问题分析, 为 mbedOS 的应用研究提供了基础, 有效地降低了 mbedOS 的移植难度, 也可为其他 RTOS 的移植提供参考.

关键词: 实时操作系统; mbedOS; 移植; ARM

引用格式: 刘长勇,王宜怀,彭涛,孙亚军,程宏玉.实时操作系统 mbedOS 的移植方法.计算机系统应用,2020,29(5):117–122. http://www.c-s-a.org.cn/1003-3254/7406.html

Transplantation Method of Real-Time Operating System mbedOS

LIU Chang-Yong[1,2,3], WANG Yi-Huai[2], PENG Tao[2], SUN Ya-Jun[2], CHENG Hong-Yu[2]

[1](School of Mathematics and Computer Science,Wuyi University, Wuyishan 354300, China)
[2](School of Computer Science and Technology, Soochow University, Suzhou 215006, China)
[3](Fujian Provincial Key Laboratory of Cognitive Computing and Intelligent Information Processing, Wuyishan 354300, China)

Abstract: The mbedOS is a real-time operating system launched by ARM in 2014 for intelligent terminals and IoT nodes. It is mainly used in embedded systems with the high real-time response time. The study analyzes the common problems of transplantation and gives specific migration steps based on the in-depth analysis of the basic functions of mbedOS, scheduling mechanism, delay function mechanism, and communication mechanism between tasks. This work is based on the portable mbedOS engineering framework. On the basis, the mbedOS is implemented in different cores of ARM Cortex-M series and different MCU transplantation. The analysis of the common problems of transplantation between different development environments are given and the basis for the application research of mbedOS are provided. It effectively reduces the difficulty of mbedOS transplantation and can also provide reference for other RTOS transplantation.

Key words: real-time operating system; mbedOS; transplantation; ARM

实时操作系统 (Real-Time Operating System, RTOS) 的运用不仅能够更有效、更合理的利用现有的 CPU 资源, 而且能够简化应用软件的设计, 缩短应用的开发时间、降低开发费用[1], 保证系统的可靠性和实时性, 那么如何实现 RTOS 在不同的内核、微控制器 (Micro Controller Unit, MCU)、开发环境等方面的移植, 一直是工程技术界研究的共性技术.

目前在 RTOS 的移植上已经有了一些研究, 也提

① 基金项目: 国家自然科学基金 (61672369), 福建省自然科学基金 (2017J01651)
Foundation item: National Natural Science Foundation of China (61672369); Natural Science Foundation of Fujian Province, China (2017J01651)
收稿时间: 2019-09-27; 修改时间: 2019-10-22; 采用时间: 2019-11-20; csa 在线出版时间: 2020-05-07

出相应的移植解决方法. 如常华利等[2]提出了一种基于 MicroBlaze 软核处理器的 μC/OS-II 的移植方案; 唐小平[3]在深入分析 μC/CPU 移植文件编写与修改的细节的基础上, 给出了基于 STM32F103RC 产品平台下的 μC/OS-III成功移植案例; 候海霞[4]对 Free RTOS 操作系统和 LwIP 协议栈进行了深入的研究, 在 STM32F407 芯片上实现了 Free RTOS 与 LwIP 协议栈的移植; 罗名驹[5]研究了如何移植引入设备树的 U-boot 和 Linux 内核到 Samsung Exynos4412 芯片上的关键技术.但是, 这些移植解决方案仅停留在解决如何将 RTOS 移植到某一特定的 MCU 上, 而对不同的内核、MCU、开发环境等的移植情况鲜有研究.同时, 有关 mbedOS 的研究主要集中在通信技术和安全访问服务机制[6]、协议栈和 IP 网络组件[7]、物联网设备平台[8,9]等方面.

2014 年 ARM 公司推出了 mbedOS, 它是一种专为物联网 (IoT) 中的“物体”设计的开源嵌入式实时操作系统[10], 具有任务管理与调度、时间管理、中断处理、内存管理、同步与通信等基本功能, 对外提供统一的底层驱动接口, 能满足多任务的运行, 具有较高的实时性.本文在深入剖析 mbedOS 的基本功能和实时性要求上, 根据嵌入式软件工程的基本思想, 以可扩充、可移植的 mbedOS 工程框架为基础, 分析了移植的共性问题和注意事项, 给出了具体的移植方法, 解决了 mbedOS 在不同的内核、MCU、开发环境等上的移植关键技术问题, 为 mbedOS 的应用研究提供了基础, 为 mbedOS 的移植提供了一条简捷、有效的路径.

1 可移植的 mbedOS 工程框架简介

这里对基于 MKL36Z64VLH4(简称 KL36) 微控制器的无操作系统工程框架 SD-NOS 和可移植的 SD-mbedOS 工程框架的构建方法做个简要的概述.

首先, 根据无操作系统软件最小系统的含义, 建立包括说明文档、内核相关文件、微控制器相关文件、用户板构件、软件构件、中断服务程序和无操作系统主程序等嵌入式系统工程最基本要素的无操作系统工程框架 SD-NOS. 该工程框架能够满足点亮一盏发光二极管的最基本要求, 甚至带有串口调试构件.

其次, 提出了 RTOS 软件最小系统的含义, 它是将任务管理、调度管理、内存管理、中断管理、时间管理以及同步与通信等 RTOS 最基本功能要素, 以构件的形式添加到无操作系统软件最小系统上, 构成能至少实现对两个任务进行调度的工程框架.

最后, 按照“分门别类、各有归处”原则, 对工程框架的文件夹和共性文件进行归纳分类, 以编号的形式建立相应的文件夹存放源程序和头文件, 基于 KL36 微控制器, 构建了可移植的 SD-mbedOS 工程框架. 表 1 给出了利用 SD-mbedOS 工程框架在采用 ARM Cortex-M 处理器的 MCU 上进行移植时, 需要替换、修改或添加的内容, 对不需要改动的内容直接进行复制, 其中底层构件仅包含最基本的通用输入输出 gpio 构件和串口通信 uart 构件.

表 1 基于 SD-mbedOS 工程框架的移植说明

文件夹		操作	移植说明
01_Doc		修改	修改工程信息.
02_Core	chip	复制、修改或替换	需修改堆栈指针、中断向量设置和包含 MCU 头文件; 内核不同时, 还需替换内核头文件和异常处理程序.
	cmsis	复制	不做改动.
03_MCU	Linker_File	替换	MCU 不同时, 需替换链接文件, 并设置 RAM 的起始地址和长度.
	MCU_drivers	替换或添加	MCU 不同时, 需替换 gpio 和 uart 构件; 根据工程需要添中新的构件.
	startup	替换	MCU 不同时, 需替换 MCU 头文件和启动文件.
04_UserBoard		替换	MCU 不同时, 需替换应用构件的头文件.
05_SoftComponent		复制	不做改动.MCU 不同时, 修改公共头文件 common.h.
06_NosPrg		修改	主函数不做改动; 根据工程需要修改中断服务程序和总包含头文件 includes.h.
07_mbedOS	manager	复制	不做改动.
	protect	复制	不做改动.
	TARGET_CORTEX	复制	不做改动.
08_mbedOsPrg		复制、修改、删除或添加	mbedOS 启动文件不做改动; 根据工程需要修改自启动任务程序, 修改、删除或添加用户任务程序.

2 mbedOS 移植涉及的文件变动及共性技术分析

移植必然涉及文件覆盖与改动, 也涉及到开发环境、编译器、内核等共性问题, 下面给出简要分析.

2.1 工程框架内的文件变动问题

SD-mbedOS 工程框架是基于 ARM Cortex-M0+内核的 KL36 微控制器进行构建的, 当采用该框架在 ARM 内核的 MCU 上进行 mbedOS 移植时, 会涉及到不同内核、不同 MCU、不同开发环境等因素, 其共性问题分析如下:

(1) mbedOS 文件的添加

由于要将 mbedOS 移植到特定 MCU 的嵌入式系统工程中, 因此需将 SD-mbedOS 工程框架下的 02_Core 和 07_mbedOS 文件夹内容添加到指定的工程中.

(2) MCU 相关文件的替换

不同的 MCU, 其内核、异常处理程序、链接文件、MCU 头文件、MCU 启动文件、底层硬件驱动构件、应用构件都有所不同. 因此, 需要根据实际采用的 MCU 和应用需求, 替换这些文件. 若采用的 MCU 为 KL36, 则不需进行替换.

(3) 内核与 MCU 相关文件的修改

不同 MCU, 其 RAM 的起始地址和大小是不一样的, 中断向量的个数也不相同, 堆栈指针的栈底地址也不同. 因此, 需要对这些内容进行重新设置, 以符合特定 MCU 的要求. 若采用的 MCU 为 KL36, 则不需进行修改.

(4) 头文件的设置

由于 MCU 发生了改变, 需要将 CMSIS 编译器、内核头文件以及 MCU 头文件加入到相关的文件中, 这样在编译时才能找到指定的文件.

2.2 移植的共性技术分析

本文涉及到的 MCU 虽然内核不相同、所采用的开发环境也不一样, 但都属于 ARM Cortex-M 处理器范畴, 都遵循微控制器软件接口标准 (Cortex Microcontroller Software Interface Standard, CMSIS). mbedOS 是由 ARM 公司开发, 其内核程序也遵循 CMSIS, 且采用 C++语言和汇编语言混合编程, 这就要求开发环境集成的编译器不仅能处理 C++语句, 而且还能编译汇编语言指令, 故一般都采用 GCC(GNU Compiler Collection, GNU 编译器套件) 编译器. 因此, 当在不同 MCU 上进行 mbedOS 移植时, 必须分析开发环境、编译器和内核等方面的设置和使用问题.

(1) 与开发环境相关问题分析. 由于开发环境不相同, 会出现有些宏、数据类型、函数调用、静态内联函数等未定义情况, 一方面是要在工程属性中设置各文件夹及其子文件夹的路径, 另一方面则需要在相关文件中添加宏定义或头文件. 例如, 在 MSP432 开发环境中, 需要在 cmsis.h 中添加“__WEAK”宏定义解决弱定义错误、在 cmsis_gcc.h 中添加“sys/_stdint.h”解决数据类型未定义问题、在链接文件中修改 heap 段的结束标志“__end”为“__end__”; 在 S32K 开发环境中, 需要在 mbed_critical.c 中添加“cmsis_gcc.h”解决__disable_irq 等函数调用问题、在 os_systick.c 中添加“core_cm4.h”和“system_S32k144.h”解决 OS_Tick_Setup 等函数调用问题、在 rtx_core_cm.h 中添加“core_cm4.h”解决静态内联函数未定义问题.

(2) 与编译器相关问题分析. mbedOS 是采用 C++和汇编语言混合编程, 不仅开发环境的编译器要设置成能同时处理 C++语句和汇编指令, 而且有的开发环境还要求对.c 的源文件要进行特殊处理, 才能满足其要求. 例如, 在 MSP432 开发环境 CCS 中, 默认编译器采用“TI V5.2.5”版本, 对编译汇编文件和 C++文件支持不足, 故建议改为“GNU V7.2.1”, 运行支持的类文件选择“libsupc++.a”, 同时添加名为“c”的 libc.a 库文件, 这样才能更好地实现对汇编指令和 C++的编译; 在 S32K 和 MSP432 开发环境中, 对.c 程序的编译时会出错, 要进行特殊处理: 一方面可以采用直接将其扩展名改为.cpp 的方式, 另一方面若该.c 文件同时存在同名的.h 文件时, 则可以通过“ifdef __cplusplus”预处理告之编译器要以 c 的形式处理这些文件.

(3) 与内核相关问题分析. mbedOS 的 SVC、PendSV、SysTick 等中断服务程序是按照 CMSIS 标准采用 C++和汇编语言混合编程, 对寄存器的使用遵循 AAPCS 规范 (ARM Archtecture Procedure Call Standard, ARM 架构过程调用标准). 当 MCU 的内核为 Cortex-M0+时, 采用 R7 寄存器保存实际调用函数的入口地址; 当采用 Cortex-M4 或 M4F 内核的 MCU 时, 则实际调用函数的入口地址保存在 R12 寄存器中. 因此, 在编写汇编程序时, 要注意避开这两个寄存器, 改用其他寄存器, 以避免程序执行时出现无法预计的情况.

3 mbedOS 的移植方法

在已有的无操作系统工程的基础上, 要将 mbedOS 移植到特定的嵌入式系统应用开发中, 可以采用以下 3 种方法进行移植: 第 1 种方法是自行分析 mbedOS 代码结构, 根据应用开发的实际从中抽取相关文件加入到工程中, 然后在工程进行编译, 修改相关的错误, 对 mbedOS 的移植有较大难度; 第 2 种方法是根据 SD-mbedOS 工程框架内的 02_Core 和 07_mbedOS 文件夹的内容, 从 mbedOS 代码中直接抽取这些源文件, 加入到工程中, 然后进行编译修改, 对 mbedOS 的移植有一定难度; 第 3 种方法是直接利用 SD-mbedOS 工程框架, 将其中的 02_Core 和 07_mbedOS 文件夹加入到工程中, 根据工程具体情况只需少量的修改就可以使用, 这样可以少走弯路, 更容易一些. 下面介绍 mbedOS 移植的第 3 种方法.

3.1 几款 MCU 比较

本文分析的是基于 KL36[11]的 SD-mbedOS 工程框架在 S32K144[12]和 MSP432[13]等 MCU 上的移植方法, 这几款 MCU 均为 ARM Cortex 处理器, 但采用的内核、开发环境、Flash、RAM、中断向量数以及生产厂家等都有所不同, 具体对比如表 2 所示.

表 2 不同 MCU 的基本情况对比表

MCU 参数	KL36	S32K144	MSP432
内核	Cortex-M0+	Cortex-M4	Cortex-M4F
Flash 大小 (KB)	64	512	256
RAM 大小 (KB)	8	60	64
中断向量数 (个)	48	256	57
生产厂家	NXP	NXP	TI
开发环境	KDS	S32DS	CCS

3.2 mbedOS 的移植步骤

当采用的 MCU 为 KL36 时, 可以直接使用 SD-mbedOS 工程框架, 只需根据实际应用的功能需求, 修改 08_mbedOsPrg 文件夹内的相关内容即可.当采用不同的 MCU 时, mbedOS 的移植步骤分析如下:

(1) 构建无操作系统工程框架. 按照 SD-NOS 工程框架的结构, 根据所采用的 MCU(其内核文件、链接文件和启动文件等在购买该 MCU 时厂家会提供或可从网络上下载获取), 使用相应的开发环境搭建无操作系统工程框架.

(2) 复制 mbedOS 文件. 由于 02_Core 文件夹包含的内核头文件、内核函数访问头文件、内核指令访问头文件以及编译器头文件等在 mbedOS 中已有且略有不同. 因此, 先删除这些文件, 然后将 SD-mbedOS 框架中的 02_Core 和 07_mbedOS 文件夹的内容复制到无操作系统工程框架中, 形成带操作系统的工程框架.

(3) 设置内核与 MCU 相关文件. 由于 MCU 不同, 其中断向量个数、中断向量表在 RAM 的起始地址、RAM 的起始地址与大小、堆栈指针等也不一样, 要根据特定的 MCU 进行重新设置; 需要将 MCU 的头文件包含在微控制器软件接口标准头文件 cmsis.h 和公共头文件 common.h 中; 由于中断向量表要从 Flash 复制到 RAM 中, 需要在 RAM 中预留出中断向量数×4 的字节数来, 故在链接文件中 RAM 的起始地址要在原起始地址的基础上加中断向量数×4. 具体需要设置的文件和内容如表 3 所示.

表 3 mbedOS 在不同 MCU 上移植主要内容修改对比表

文件	修改说明	KL36	S32K144	MSP432
cmsis.h	MCU 头文件	MKL36Z4.h	S32K144.h	msp432p401r.h
cmsis_nvic.h	中断向量数	48	256	57
	中断向量表在 RAM 的起始地址	0x1FFFF800	0x1FFF8000	0x20000000
链接文件	RAM 起始地址	0x1FFFF8C0	0x1FFF8400	0x200000E4
	RAM 大小	0x00001F40	0x0000EC00	0x0000FF1C
mbed_rtx.h	堆栈指针	0x20001800	0x20007000	0x20010000
common.h	MCU 头文件	MKL36Z4.h	S32K144.h	msp432p401r.h

(4) 设置工程属性. 为了便于在函数调用中能找到相关的文件声明位置, 可以在工程配置文件中添加各个文件夹及子文件夹的路径, 同时添加相关宏定义, 以便在函数调用时能找到这些头文件和宏.

(5) 与应用开发相关的内容设置. 在实际的应用开发中, 若有新增底层硬件构件、应用构件、软件构件、用户任务程序、中断服务程序等, 需要在总包含头文件 includes.h 中要包含各类构件头文件、用户函数声明和中断声明. 这一步根据需要进行设置, 若无增加则不需要进行设置.

(6) 编译工程. 在开发环境中对工程进行编译, 检查并修改错误, 编译成功之后下载到开发板上运行, 观察运行结果.

3.3 移植测试

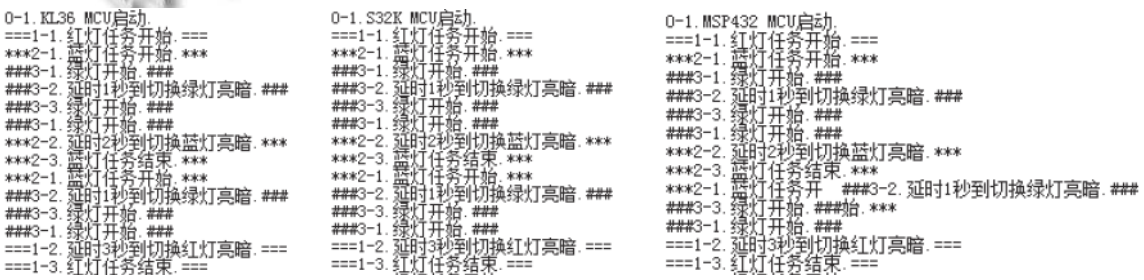
通过以上移植步骤, 按照表 3 的修改内容以及共性技术分析的要点, 就可以利用 KL36 的 SD-mbedOS

工程框架将 mbedOS 移植到以 S32K 和 MSP432 为 MCU 的工程上. 由于是针对 mbedOS 进行移植测试, 因此, 在三款 MCU 的工程中实现相同的任务功能, 可以将 KL36 工程中的蓝灯任务、绿灯任务和红灯任务这 3 个任务程序代码直接复制到 S32K 和 MSP432 工程中进行移植测试. 移植测试工程实现蓝灯任务、绿灯任务和红灯任务分别每 2 s、每 1 s 和每 3 s 闪烁 1 次, 由于篇幅有限, 仅给出蓝灯任务的程序代码, 绿灯任务和红灯任务的程序代码与蓝灯任务的程序代码基本一样, 只是延时时间不同而已, 更加详细的工程代码与移植说明可到苏州大学嵌入式学习社区网站 (网址: http://sumcu.suda.edu.cn) 的“教学培训-教学资料-mbedOS”位置, 下载“The Protability of mbedOS in MCUs_190928”查看.

蓝灯任务代码:

```
void run_bluelight(void)
{
  while (true) {
   printf("    ***2-1.蓝灯任务开始.***\n");
   light_change(LIGHT_BLUE); //切换蓝灯的亮暗
   Thread:: wait(2000);              //延时 2 秒
   printf("    ***2-2.延时 2 秒到切换蓝灯亮暗.***\n");
   printf("    ***2-3.蓝灯任务结束.***\n");
  }
}
```

图 1 给出了 mbedOS 在 KL36 上的运行结果, 以及在 S32K 和 MSP432 上的移植测试结果, 从中可以看出在 mbedOS 的调度下任务运行正常、程序执行逻辑准确, 本测试工程涉及到 RTOS 的任务调度、时间嘀嗒、延时函数、事件、消息队列等基本要素, 这些要素具有普适意义, 程序的正常运行, 表明移植成功.



(a) 在 KL36 上的运行结果 (b) 在 S32K 上的移植测试结果 (c) 在 MSP432 上的移植测试结果

图 1 mbedOS 在多个 MCU 的移植测试结果

4 结论与展望

实时操作系统 mbedOS 具有任务管理与调度、时间管理、中断处理、内存管理、同步与通信等基本功能, 能提供调度的实时性、响应时间的可确定性、系统高度的可靠性, 在物联网终端、工业控制设备、军事设备、航空航天等领域得到广泛应用. 本文通过对 mbedOS 的基本功能和实时性等方面进行深入剖析, 根据嵌入式软件工程的基本思想和嵌入式软件构件设计的基本原则, 以可移植、易扩充的 SD-mbedOS 工程框架为基础, 分析了移植的共性技术和注意事项, 给出了 mbedOS 在 ARM Cortex-M 系列的不同内核、MCU、开发环境等的移植解决方案, 并完成了移植测试, 为广大嵌入式系统开发者在 mbedOS 移植上少走弯路, 提供了简洁、方便的路径, 也可为其他 RTOS 的移植提供参考. 后续将针对 mbedOS 的启动、系统服务调用、任务调度等做进一下的研究探讨.

参考文献

1 刘凤. 基于软件构件技术的软件化雷达. 现代雷达, 2016, 38(5): 12–15.

2 常华利, 尹震宇. 基于 MicroBlaze 的 μC/OS-Ⅱ操作系统移植. 计算机系统应用, 2017, 26(5): 239–246. [doi: 10.15888/j.cnki.csa.005742]

3 唐小平. μC/OS-Ⅲ在 STM32F103RC 上的移植. 兵工自动化, 2016, 35(7): 62–65.

4 候海霞. FreeRTOS 和 LwIP 的移植与系统内存分配策略的比较[硕士学位论文]. 北京: 华北电力大学, 2017.

5 罗名驹. 基于 ARM Cortex-A9 的嵌入式 Linux 内核移植研究与实现[硕士学位论文]. 广州: 广东工业大学, 2017.

6 Persson P, Angelsmark O. Calvin-merging cloud and IoT. Procedia Computer Science, 2015, 52: 210–217. [doi: 10.1016/j.procs.2015.05.059]

7 Bock C, Marquardt M, Martens A, *et al*. Smart sensors and actors with BACnetTM and mbed OS on cortex-M microcontrollers. Proceedings of the IEEE 5th World Forum on Internet of Things (WF-IoT). Limerick, Ireland 2019. 937–942.

8 Muhammad A, Afzal B, Imran B, *et al*. oneM2M architecture based secure MQTT binding in mbed OS. Proceedings of 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). Stockholm, Sweden. 2019. 48–56.

9 Balsamo D, Elboreini A, Al-Hashimi B M, *et al*. Exploring ARM mbed support for transient computing in energy harvesting IoT systems. Proceedings of the 7th IEEE International Workshop on Advances in Sensors and Interfaces. Vieste, Italy. 2017. 115–120.

10 ARM limited. Mbed OS. https://www.mbed.com/zh-cn/development/mbed-os/. [2019-05-05].

11 NXP Semiconductors. KL36 sub-family reference manual. https://www.nxp.com/docs/en/reference-manual/KL36P121-M48SF4RM.pdf. (2013-07-03).

12 NXP Semiconductors. S32K1xx series reference manual. https://www.nxp.com/docs/en/reference-manual/S32K-RM.pdf. [2019-06-11].

13 Texas Instruments Incorporated. MSP432P4xx SimpleLink™ microcontrollers technical reference manual. http://www.ti.com/lit/ug/slau356i/slau356i.pdf. [2019-07-16].