

μ C/OS - III 在 Cortex - M3 处理器上的移植

李承创, 陈跃斌, 房晓丽, 王兵

(云南民族大学 电气信息工程学院, 昆明 650031)

摘要: 为了将 μ C/OS - III 移植到 Cortex - M3 处理器上, 选用 RealView MDK 作为软件开发平台, 针对 Cortex - M3 处理器特性编写了移植所需的 C 语言和汇编语言源代码, 并验证了移植的正确性。移植后的 μ C/OS - III 能够稳定运行于 Cortex - M3 处理器上。该移植对大部分 Cortex - M3 处理器具有通用性, 对其他架构处理器的 μ C/OS - III 移植具有参考作用。

关键词: μ C/OS - III; 嵌入式操作系统; ARM; Cortex - M3; 移植

中图分类号: TP316.2

文献标识码: A

Porting μ C/OS - III to Cortex - M3 Processor

Li Chengchuang, Chen Yuebin, Fang Xiaoli, Wang bing

(School of Electrical and Information Technology, Yunnan University of Nationalities, Kunming 650031, China)

Abstract: In order to port μ C/OS - III to Cortex - M3 processor, it uses RealView MDK as the software development platform. With Cortex - M3 processor's features, it writes the required C and assembly source code for porting, and verifies the correctness of the porting. After porting, μ C/OS - III can run on Cortex - M3 processor stably. The porting is universal for most Cortex - M3 processor, which is a reference for μ C/OS - III porting on processors of other architectures.

Key words: μ C/OS - III; embedded operating system; ARM; Cortex - M3; porting

引言

μ C/OS - III 是一款基于优先级调度的抢占式实时内核, Micrium 公司于 2011 年 8 月公开了 μ C/OS - III 的源码, 其源码遵循 ANSI C 标准, 因而具有良好的移植性, 相信其将会被移植到越来越多的处理器体系上。本文主要完成基于 Cortex - M3 处理器的 μ C/OS - III 移植, 通过本次移植, 加深对嵌入式操作系统原理的理解。此外, 在 μ C/OS - III 移植成功的基础上进行嵌入式应用程序开发, 可以把主要精力集中到应用程序上, 而硬件资源交由 μ C/OS - III 管理, 从而使得嵌入式应用程序更易开发和维护, 在嵌入式软硬件结构变得越来越复杂的今天具有现实意义。

1 μ C/OS - III 和 Cortex - M3 特点

相对以前的版本, μ C/OS - III 最大改进之处在于允许多个任务运行于同一优先级上, 相同优先级的任务按时间片轮转调度, 内核对象的数量不受限制, 以及接近于零的中断禁用时钟周期。

Cortex - M3 是 ARM 公司推出的基于 ARMv7 - M 架构的内核, 主要针对高性能、低成本和低功耗的嵌入式

应用。Cortex - M3 拥有固定的存储器映射, 采用更高效的 NVIC(Nested Vectored Interrupt Controller)、更简单的堆栈以及更高性能的指令集, 且 NVIC(包括 SysTick)的寄存器位置固定, 极大地方便了 μ C/OS - III 的移植及在基于 Cortex - M3 内核的处理器之间的迁移。

2 移植

2.1 移植方案

本文移植 μ C/OS - III 内核的版本为 V3.02.00, 其源代码下载地址见参考文献[4]。选用法半导体(ST)公司生产的基于 Cortex - M3 内核的 STM32F103RBT6 微控制器作为硬件实验平台, 而编译环境采用 RealView MDK V3.5。

Cortex - M3 支持两种特权级别: 特权级和用户级, μ C/OS - III 内核和用户代码都运行于特权级下。Cortex - M3 还支持两个栈指针 MSP 和 PSP, μ C/OS - III 内核和 ISR(Interrupt Service Routine)使用 MSP, μ C/OS - III 的任务则使用 PSP。

首先针对 Cortex - M3 处理器的特性编写与内核、CPU 和 BSP(Board Support Package)相关的源代码, 然后创建若干个简单的用户任务, 在具体的硬件平台上测试移

植后的 $\mu\text{C}/\text{OS}-\text{III}$ 。

2.2 内核相关

2.2.1 编写 os_cpu.h

os_cpu.h 头文件主要是对上下文切换函数和时间戳获取函数进行宏定义。 $\mu\text{C}/\text{OS}-\text{III}$ 的上下文切换包括两种类型:任务级上下文切换 OS_TASK_SW() 和中断级上下文切换 OSIntCtxSw()。它们使用相同的代码置位 ICSR.PENDSVSET 以悬起 PendSV 异常,由 PendSV 的 ISR“缓期执行”上下文切换。

OS_TS_GET() 的作用是获取当前时间戳,若使能 $\mu\text{C}/\text{OS}-\text{III}$ 的时间戳功能,则将 OS_TS_GET() 宏定义为 CPU_TS_TmrRd(), 否则简单地宏定义为 0。

2.2.2 编写 os_cpu_a.asm

在 os_cpu_a.asm 文件中需要用汇编指令实现 OSStartHighRdy() 函数和 PendSV 的 ISR。OSStartHighRdy() 函数被内核用于调度第一个最高优先级的就绪任务,以开始多任务运行环境,汇编代码实现如下:

OSStartHighRdy

```
LDR R0,=NVIC_SYSPRI14
LDR R1,=PRI_LOWEST;设置 PendSV 优先级为最低
STRB R1,[R0]
MOV R0,#0 ;设置 PSP 为 0
MSR PSP,R0
LDR R0=NVIC_ICSR ;挂起 PendSV 异常
LDR R1,=PENDSVSET
STR R1,[R0]
CPSIE I ;确保中断使能
```

Cortex-M3 支持 PendSV 异常,而 PendSV 异常的典型应用场合就是上下文切换。得益于 Cortex-M3 的中断机制, $\mu\text{C}/\text{OS}-\text{III}$ 上下文切换只需保存和恢复 R11~R4、PSP,而 PSR、PC、LR、R12、R3~R0 由硬件自动保存和恢复。PendSV 的 ISR 汇编代码如下:

PendSV_Handler ;确保此标识符与中断向量表中的一致

```
CPSID I ;上下文切换期间禁用中断
MRS R0,PSP
CBZ R0,SkipSaving ;如果 PSP 等于 0,则跳过现场保存
STMFD R0!,{R4-R11} ;保存 R4~R11
LDR R1,=OSTCBCurPtr ;保存 PSP
LDR R2,[R1]
STR R0,[R2]
```

SkipSaving

```
PUSH {LR} ;调用 OSTaskSwHook()
LDR R0,=OSTaskSwHook
BLX R0
POP {LR}
```

```
LDR R0,=OSPrioCur ;OSPrioCur=OSPrioHighRdy
LDR R1,=OSPrioHighRdy
LDRB R2,[R1]
STRB R2,[R0]
LDR R0,=OSTCBCurPtr
;OSTCBCurPtr=OSTCBHighRdyPtr
LDR R1,=OSTCBHighRdyPtr
LDR R2,[R1]
STR R2,[R0]
LDR R0,[R2] ;R0=OSTCBHighRdyPtr->StkPtr
LDMFD R0!,{R4-R11} ;恢复 R4~R11
MSR PSP,R0 ;恢复 PSP
ORR LR,#0x04 ;Return Stack 选择 PSP
CPSIE I ;开中断
BX LR ;异常返回
```

2.2.3 编写 os_cpu_c.c

os_cpu_c.c 文件包含了 OSTaskStkInit() 函数和若干钩子函数。OSTaskStkInit() 函数的作用是在创建任务时初始化任务栈,并返回新的栈顶位置。 $\mu\text{C}/\text{OS}-\text{III}$ 基于 Cortex-M3 的任务栈结构如图 1 所示。其中 PSR、PC、LR、R1、R0 五个寄存器应赋予正确的初值,而其他 11 个寄存器的初值无关重要。

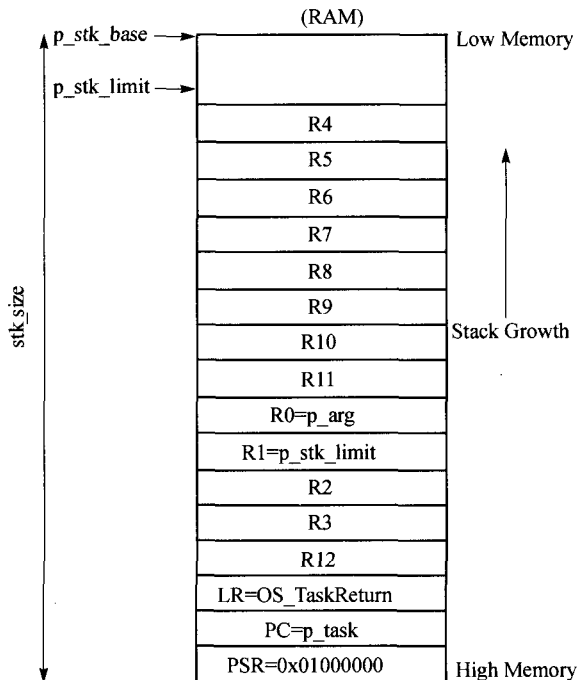


图 1 $\mu\text{C}/\text{OS}-\text{III}$ 基于 Cortex-M3 的任务栈结构

os_cpu_c.c 文件中的钩子函数是 $\mu\text{C}/\text{OS}-\text{III}$ 为了扩展用户功能而定义的。进行 $\mu\text{C}/\text{OS}-\text{III}$ 移植时至少需要定义 OSTaskSwHook()、OSInitHook()、OSTimeTickHook()、OSIdleTaskHook()、OSStatTaskHook()、OSTa-

skCreateHook()、OSTaskDelHook()、OSTaskReturnHook() 八个钩子函数。为了简单起见,本次移植不对钩子函数作功能扩展。

2.3 CPU 相关

2.3.1 编写 cpu.h

cpu.h 头文件主要包括对标准数据类型、字长、栈、临界区的相关定义。标准数据类型与具体的编译器相关,需要查阅相应的编译器手册。Cortex-M3 字长是 32 位,则 CPU_DATA 和 CPU_ADDR 皆定义为 CPU_INT32U 类型。Cortex-M3 使用满降序栈,栈增长方向应为从高地址到低地址。临界区方法选用 CPU_CRITICAL_METHOD_STATUS_LOCAL。

2.3.2 编写 cpu_a.asm

cpu_a.asm 文件的最主要部分是临界区函数的实现。根据所选用的临界区方法,中断使能函数 CPU_SR_Save() 和中断禁用函数 CPU_SR_Restore() 代码实现如下:

CPU_SR_Save

```
MRS    R0, PRIMASK    ;保存 PRIMASK 的值
CPSID  I                ;关中断
BX      LR
```

CPU_SR_Restore

```
MSR     PRIMASK, R0    ;恢复 PRIMASK 的值
BX      LR
```

Cortex-M3 的指令集提供了 CLZ 指令,则可选地使用汇编指令实现 CPU_CntLeadZeros() 函数,以加快 $\mu\text{C}/\text{OS-III}$ 调度器查找最高优先级的就绪任务的速度,CPU_CntLeadZeros() 函数汇编代码实现如下:

CPU_CntLeadZeros

```
CLZ     R0, R0
BX      LR
```

2.4 BSP

Cortex-M3 内核包含了一个 SysTick 定时器,可以用来给 $\mu\text{C}/\text{OS-III}$ 提供系统时钟节拍。SysTick 初始化和 ISR 的源代码实现分别如下:

void SysTick_Init(CPU_INT32U cnts){

```
SysTick-> LOAD=cnts;    //重载值寄存器
SysTick-> VAL=0;        //当前计数值
SysTick-> CTRL|=0x00000007; //控制寄存器
```

}

void SysTick_Handler(void){

```
CPU_SR_ALLOC();
CPU_CRITICAL_ENTER();
OSIntEnter();
CPU_CRITICAL_EXIT();
OSTimeTick();
```

OSIntExit();

}

$\mu\text{C}/\text{OS-III}$ 新增了时间戳功能,用于测量中断禁用时长、代码执行时长和确定事件发生时间等。时间戳定时器可以由 DWT(Data Watchpoint and Trace)的时钟周期计数器 CYCCNT 充当,该计数器是一个自由运行的 32 位递增计数器,溢出时自动重载为 0,周而复始。时间戳定时器初始化和读取函数源代码实现分别如下:

void CPU_TS_TmrInit(void){

```
DEM_CR |= TRCENA;    //置位 DEM_CR. TRCENA
DWT_CYCCNT = 0u;
DWT_CTRL |= CYCCNTENA; //使能 CYCCNT 计数器
```

}

CPU_TS_TMR CPU_TS_TmrRd(void){

```
return (CPU_TS_TMR)DWT_CYCCNT;
```

}

此外,本移植过程的 BSP 还涉及 RCC、GPIO、NVIC 和 LED/LCD 等硬件的初始化函数和驱动程序。

3 测试

首先不加任何用户任务来测试移植好的 $\mu\text{C}/\text{OS-III}$ 内核自身运行情况,待验证内核正常运行之后,编写 TaskLed1、TaskLed2、TaskLed3、TaskProfile 四个任务,其中前 3 个任务被赋予相同的优先级(本移植是假设使能了 $\mu\text{C}/\text{OS-III}$ 的轮转调度功能),实现对 3 盏 LED 灯不停地闪烁;而 TaskProfile 的功能是在液晶屏上显示上下文切换次数。

运行结果如图 2 所示。

图中 3 盏 LED 灯不停地闪烁,验证了 $\mu\text{C}/\text{OS-III}$ 的相同优先级任务轮转调度的特征;LCD 上显示 Ctx-SwCtr 的值一直在增加,指示不断发生上下文切换。系统连续稳定运行 5 个小时以上没出现任何问题,可见本移植是成功的。

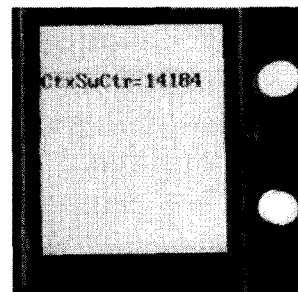


图 2 运行结果

结 语

本文主要论述了基于 Cortex-M3 内核处理器上 $\mu\text{C}/\text{OS-III}$ 的移植过程并给出关键代码,移植后的 $\mu\text{C}/\text{OS-III}$ 能够稳定运行于 STM32F103RBT6 处理器上。本移植能通用于大部分 Cortex-M3 内核的处理器,并对于将 $\mu\text{C}/\text{OS-III}$ 移植到其他体系结构的处理器上具有参考价值。

速率快,广泛应用于移动设备。本手持终端采用了 SD 卡存储汉字字库、界面图片和射频卡中读取的数据。SD 卡通过 SPI 总线与 STM32F103VET6 进行通信,经实验证明每秒可以传输 2 MB 以上的数据,可以满足手持终端对数据传输速率的要求。另外,由于 SD 卡可以很方便地从手持终端中取出,也可以使用上位机的通用读卡器对 SD 卡进行读写,实现手持终端和上位机的数据交换。

2.5 数据通信部分的设计

RFID 手持终端使用 STM32F103VET6 芯片内部集成的 USB 总线与上位机完成有线通信。USB 总线支持即插即用和热插拔,使用方便。同时,USB 2.0 全速总线支持 480 Mbps 的传输速率,可以快速将手持终端中的信息传输到上位机。为了满足手持终端的移动使用需求,采用了 Simcom 公司生产的 GPRS 模块 SIM300,它的工作频率为 GSM/GPRS 900/1800 MHz,可以在低功耗的条件下,完成手持终端数据的无线传输。在使用时,通过 STM32F103VET6 的 USART 串口与 SIM300 模块连接,通过 AT 指令实现网络连接、数据发送等功能。

3 系统测试

设计了 RFID 手持终端的 PCB 板,其主板大小约为 16 cm × 9 cm,可以满足手持终端的便携需求。使用 STM32F103VET6 自带的 ISP 下载工具通过 USART 串口将程序下载后,使用本 RFID 手持终端对符合 ISO/IEC 14443 和 ISO/IEC 15693 标准的射频标签进行读写,操作距离均不小于 8 cm,读卡及显示速度均满足使用需求。将读卡得到的数据存储到 SD 卡中,通过 USB 总线或 GPRS 模块发送到上位机,上位机可以接收到卡号、扇区、数据等信息以便进行进一步的数据处理。

结 语

本文详细介绍了基于 STM32F103VET6 的 13.56MHz RFID 手持终端的硬件设计方法。该读卡器具有处理速度快、功耗低、人机交互友好、与上位机通信方便等特点,适用于多种需要移动应用的场合,尤其适用于物流行业,具有广阔的应用前景。

参考文献

- [1] 刘军. 例说 STM32[M]. 北京:北京航空航天大学出版社, 2011:11-18.
- [2] STMicroelectronics. High-density performance line ARM-based 32-bit MCU with 256 to 512KB Flash, USB, CAN, 11 timers, 3 ADCs, 13 communication interfaces[EB/OL]. [2011-11-25]. <http://www.st.com/internet/mcu/product/164491.jsp>.
- [3] NXP Semiconductors. Multiple protocol contactless reader IC (MIFARE/I-CODE1) (v. 3.5)[EB/OL]. [2011-11-25]. http://www.nxp.com/documents/data_sheet/CLRC632.pdf.
- [4] Philips Semiconductors. AN Micore Reader IC Family; Directly Matched Antenna Design (v. 2.05)[EB/OL]. [2011-11-25]. http://www.nxp.com/documents/application_note/AN077925.pdf.
- [5] NXP Semiconductors. Directly matched Antenna Excel calculation (v 1.0)[EB/OL]. [2011-11-25]. http://www.nxp.com/documents/application_note/149110.zip.
- [6] 李新春,于永鑫. 移动式 13.56MHz RFID 读卡器的设计[J]. 计算机系统应用, 2011, 20(8): 229-232.

陈博(硕士研究生),研究方向为嵌入式技术、射频识别技术;刘开华(教授),研究方向为射频识别理论与应用、数字信号处理理论与应用、数字电视系统技术。

(责任编辑:杨迪娜 收稿日期:2011-11-29)

参考文献

- [1] Matt Gordon. Micrium makes μ C/OS-III source available [EB/OL]. [2011-11]. <http://micrium.com/page/press-room/news/id:65>.
- [2] Brian Nagel, Micrium. Advantages of the Cortex-M3[J]. Information Quarterly, 2008, 7(4): 27-32.
- [3] ARM 中国. 针对 ARM Cortex-M3 平台的代码移植[J]. Information Quarterly, 2007(6): 44-46.
- [4] Micrium. Download μ C/OS source code[EB/OL]. [2011-10-23]. http://micrium.com/page/downloads/source_code.

- [5] Joseph Yiu. The definitive guide to the ARM Cortex-M3 [M]. 2th ed. Burlington: Newnes, 2010: 127-129.
- [6] 方安平,蔡俊宇. Cortex-M3 的异常处理机制研究[J]. 单片机与嵌入式系统应用, 2009(2): 15-16.
- [7] 邵贝贝. 浅谈 μ C/OS 任务调度算法的硬件实现[J]. 单片机与嵌入式系统应用, 2010(9): 5-7.

李承创(硕士研究生),主要研究方向为嵌入式操作系统。

(责任编辑:杨迪娜 收稿日期:2011-11-15)