实时操作系统 μC/OS-Ⅱ 在 LPC2114上的移植

廖文良，褚艺斌，陈文芗
(厦门大学 机电工程系，福建 厦门 361005)

摘　要：介绍了实时操作系统 μC/OS- II 的特点和内核结构，并实现了 μC/OS- II 在 Philips 嵌入式处理器 LPC2114 上的移植。

关键词：μC/OS- II　LPC2114　移植

作为一个实时内核，μC/OS 从 1992 年开始为人们熟悉，到现在已经发展为 μC/OS- II。μC/OS- II 最多支持 56 个任务，其内核为占先式，即总是执行就绪态的优先级最高的任务，并支持 Semaphore(信号量)、Mailbox(邮箱)、Message Queue(消息队列)等多种常用的进程间通信机制。与大多数商用 RTOS 不同的是，μC/OS- II 公开其全部源代码，并可以免费获得，对商业应用只收取少量的 License 费用。

LPC2114 是 Philips 公司开发的一款支持实时仿真和跟踪的 ARM7TDMI- S CPU，并嵌入了 128KB 的高速 Flash 存储器。其内部集成了与片内存储器控制器接口的 ARM7 局部总线、与中断控制器接口的 AMBA 高性能总线(AHB)和连接片内外设功能的 VLSI 外设总线(VPB，ARM AMBA 总线的兼容超集)。LPC2114 将 ARM7TDMI- S 配置为小端(little- endian)字节顺序。128 位宽度的存储器接口和独特的加速结构使 32 位代码能够在最大时钟频率下运行。

将 μC/OS- II 移植在 LPC2119 上不仅有益于 ARM 和 μC/OS- II 在车用控制器上的应用，其成果还可以用于其他嵌入式工业控制领域。本次移植中，使用 CodeWarrior For ARM Developer Suite v1.2 编译调试环境。

1　μC/OS- II 系统结构

图 1 为μC/OS- II 的软硬件体系结构。应用程序处于整个系统的顶层，每个任务都可以认为自己独占了 CPU，因而可以设计成为一个无限循环。μC/OS- II 处理器无关的代码提供了μC/OS- II 的系统服务，应用程序可以使用这些 API 函数进行内存管理、任务间通信以及创建、删除任务等。

大部分 μC/OS- II 代码是使用 ANSI C 语言编写的，因此 μC/OS- II 的可移植性较好。尽管如此，仍然需要使用 C 和汇编语言写一些处理器相关的代码。μC/OS- II 的移植需要满足下列要求：(1)处理器的 C 编译器可以产生可重入代码。(2)可以使用 C 调用进入和退出 Critical Code(临界区代码)。(3)处理器必须支持硬件中断，并且需要一个定时中断源。(4)处理器需要能够容纳一定数据的硬件堆栈。(5)处理器需要有能够在 CPU 寄存器与内存和堆栈交换数据的指令。

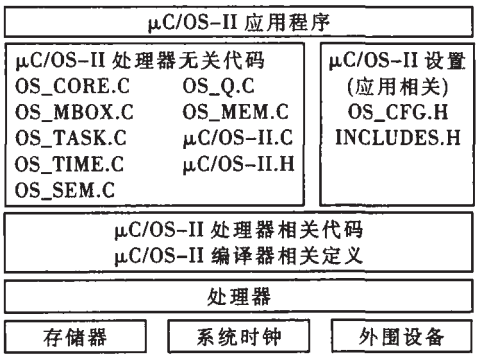

图 1　μC/OS- II 的软硬件体系结构

移植 μC/OS- II 的主要工作涉及处理器及编译器相关代码以及 BSP 的编写。

2　μC/OS- II BSP 的编写

BSP(板级支持包)是介于底层硬件和操作系统之间的软件层次，它完成系统上电后最初的硬件和软件初始化，并对底层硬件进行封装，使得操作系统不再面对具体的硬件。

为 μC/OS- II 编写一个简单的 BSP 的方法是：首先设置 CPU 内部寄存器和系统堆栈，并初始化堆栈指针，建立程序的运行和调用环境；然后使用 C 语言设置 LPC2114 向量中断控制器、GPIO 以及 SRAM 控制器，初始化串口(UART0)作为默认打印口，并向操作系统提供一些硬件相关例程和函数(如 dprintf())，以方便调试；在 CPU、板级和程序自身初始化完成后，就可以把 CPU 的控制权交给操作系统了。

LPC2114 处理器支持七种类型的异常。异常出现后，CPU 强制从异常类型对应的固定存储地址开始执行程序，因此需要在程序头建立起异常向量表，例如：

```
Vectors
        LDR     PC, ResetAddr
        LDR     PC, UndefinedAddr
        LDR     PC, SWI_Addr
        LDR     PC, PrefetchAddr
```

```
        LDR     PC, DataAbortAddr
        DCD     0xb9205f80
        LDR     PC, [PC, #- 0xff0]
        LDR     PC, FIQ_Addr
ResetAddr       DCD     Reset
UndefinedAddr   DCD     Undefined
SWI_Addr        DCD     SoftwareInterrupt
PrefetvhAddr    DCD     PrefetchAbort
DataAbortAddr   DCD     DataAbort
NoUse           DCD     0
IRQ_Addr        DCD     IRQ_Handler
FIQ_Addr        DCD     FIQ_Handler
```

向量从上到下依次为复位、未定义指令异常、软件中断、预取指令中止、预取数据中止、保留的异常、IRQ 和 FIQ。保留的异常向量位置所填的数据 0xb9205f80 是为了使向量表中所有的数据 32 位累加和为 0。这个向量在 ARM 文件中标识为保留，该位置被 Boot 装载程序用作有效的用户程序关键字。当向量表中所有的数据累加为 0(且外部硬件禁止进入 ISP 程序)时，Boot 装载程序将执行用户程序。

从异常向量表可知：芯片复位时程序会跳转到标号 Reset 处。程序首先调用 InitStack 初始化各种模式的堆栈，然后调用 TargetResetInit 对系统进行基本的初始化，最后跳转到 ADS 提供的启动代码 __main。例如：

```
Reset
    BL      InitStack
    BL      TargetResetInit
    B       __main
```

同时在每个硬件时钟到来后，μC/OS- II 会在中断服务例程中调用 OSIntCtxSw()进行任务调度。另外，当某个任务因等待资源而被挂起时，它可以自己主动放弃 CPU，而没有必要等到自己的时间片全都用完。这可以通过调用一个任务级的任务调度函数 OSCtxSw()来实现，其中相对复杂的是 OSIntCtxSw()。由于 OSLickISR()调用了 OSIntExit()，OSIntExit()又再次调用了 OSIntCtxSw()，如果进行任务切换，则二次调用都不会返回，而不同的 C 编译器、不同的编译选项处理 C 调用时对堆栈的使用也不尽相同。因此 OSIntCtxSw()是与编译器相关的。在 ADS 编译环境下，OSIntCtxSw 的软件流程如图 2 所示。

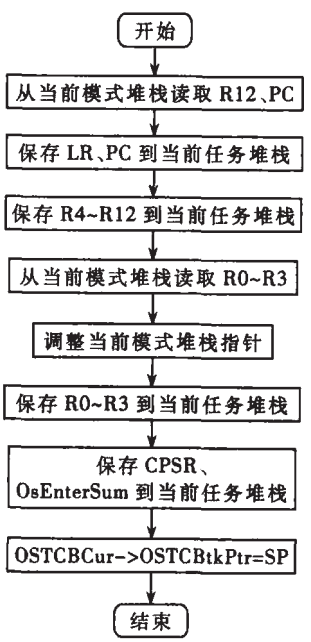

图 2 OSIntCtxSw的软件流程图

3 μC/OS- II 任务堆栈初始化

μC/OS- II 中每个任务都有自己的任务堆栈。在任务创建初期由 OSTaskStkInit()初始化。初始化堆栈的目的就是模拟一次中断。任务堆栈中保存了任务代码的起始地址和一些 CPU 寄存器(初值是无关紧要的)，这样一旦条件满足，就可以执行任务了。LPC2114 在中断发生时，会自动保存程序指针 PC、状态寄存器 SR 以及其他一些信息。图 3 为针对 LPC2114 编程结构设计的堆栈结构。

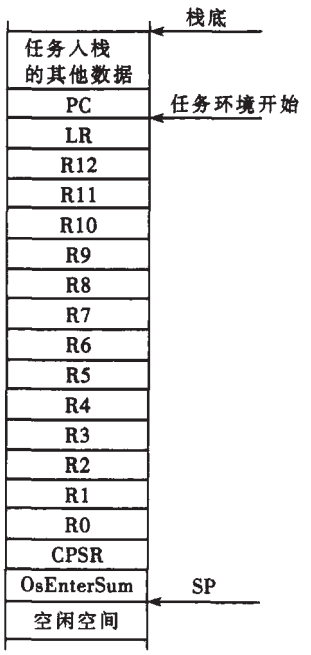

图 3 任务堆栈结构

本次移植的函数 OSTaskStkInt()代码为：

```
OS_STK * OSTaskStkInit(void( * task)(void * pd), void * pdata,
OS_STK * ptos, INT16U opt)
{
    OS_Stk * stk;
    opt=opt; /* opt 没有使用。作用是避免编译器警告 */
    stk=ptos;  /* 获取堆栈指针 */
    /* 建立任务环境，ADS1.2 使用满递减堆栈 */
 * stk=(OS_STK)task; /* pc */ - - stk=(OS_STK)task; /* lr */
 * - - stk=0;     /* r12 */
 * - - stk=0;     /* r11 */
 * - - stk=0;     /* r10 */
 * - - stk=0;     /* r9 */
 * - - stk=0;     /* r8 */
 * - - stk=0;     /* r7 */
 * - - stk=0;     /* r6 */
 * - - stk=0;     /* r5 */
 * - - stk=0;     /* r4 */
 * - - stk=0;     /* r3 */
 * - - stk=0;     /* r2 */
 * - - stk=0;     /* r1 */
 * - - stk=(unsigned int)pdata;  /* r0，第一个参数使用 R0 递推 */
 * - - stk=(USER_USING_MODE|0x00);
                                  /* spsr，允许 IRQ、FIQ 中断 */
 * - - stk=0;                    /* 关中断计数器 OsEnterSum */
return(stk);
}
```

4 μC/OS- II 系统时钟管理

μC/OS- II 需要在系统初始化时开始一个系统时钟节拍，它是 OS 系统的时间基准。该时钟节拍一般由时间中断

产生。LPC2114 中可产生时钟节拍的模块很多，本次移植采用定时器 0 异常。因为它与外部中断使用不同的异常向量，便于对异常事件的管理，有利于提高 OS 的稳定性。32 位定时器 TC 的计数频率由 plck 经过 PR 分频控制得到，而定时器的启动/停止、计数复位由 TCR 控制。当有捕获事件或比较匹配事件发生时，IR 会设置相关的中断标志，若已打开中断允许，则会产生中断。

本次移植设置系统时钟频率为 11.0592MHz，代码在时钟初始化和每次进入定时器 0 异常时，将定时器 0 的计数器 PWMTC 设置为 11.0592M/OS_TICKS_PER_SEC，这样可使 OS 每秒钟产生 OS_TICKS_PER_SEC 的时钟节拍。

5 应用方法

在使用移植后的 OS 时，用户需要编写自己的主程序 main()，其流程图如图 4。在适当的初始化后即可启动 OS。

另外，用户需在 TaskStart 任务中启动时钟节拍，调用 OS_StartInit()函数初始化统计任务，创建所需的其他任务，最后调用 OSTaskDel () 函数删除 TaskStart 任务。OS 在该函数调用结束后，会自动允许异常和中断，OS 正常运转，不断调度任务，响应中断。

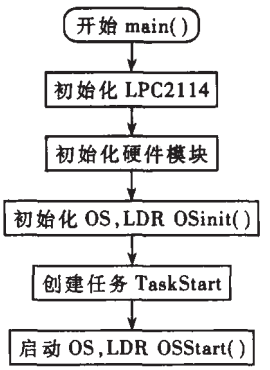

图 4 主程序流程图

参考文献

1 LABROSSE J J.μC/OS- II: the Real Time Kernel.RS: R&D Books, 1999

2 周立功.ARM 微控制器基础与实战.北京：北京航空航天大学出版社，2003

3 Labrosse J 著，邵贝贝译.嵌入式实时操作系统 μC/OS- II（第二版）.北京：北京航空航天大学出版社，2003

（收稿日期：2005- 02- 20）

（上接第 31 页）

的染色体作为解输出，若多个染色体的综合评价值相等，则选择 $w/x_{i,0}$ 值最小的染色体。算法结束。

3 算法性能分析

网络拓扑模型如图 4 所示。应用本算法对其进行仿真实验。假设源节点为 A，目的节点为 H，链路特性用五元组$(x_1, x_2, x_3, x_4, x_5)$描述，分别表示带宽、时延、成功率(=1- 丢包率)、成本和时延抖动。QoS 路由要求为：所需带宽 b=2bps，最大时延 d=8μs，时延抖动 j≤10μs，成功率≥80%，成本≤10，综合评价值≤1。编码长度为 9 位，群体大小为 100，交叉概率为 0.5，采用自适应变异概率，由式(1)得出，初始群体随机产生。

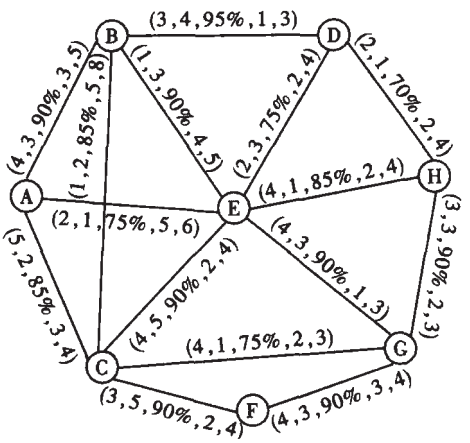

图 4 网络拓扑模型

仿真结果如表 2 所示。由表 2 可以看出，第 2、3 个解的综合评价值最小，符合 QoS 要求且达到负载分布均衡，但第 3 个解的 $w/x_{i,0}$ 值最小，所以第 3 个解(路径 A- C- E- H)为最优解。

4 结 论

本文在单点投递的情况下，利用改进的遗传算法求解具有多个 QoS 约束的路由选择问题。该算法主要有以下特点：(1)采用预处理机制，有效地减少了算法编码空间和搜索空间，提高了算法的搜索效率。(2)特殊的树结构编码，使染色体长度达到固定，简化了编码操作，省略了复杂的编码和解码过程。(3)通过对交叉操作和变异操作的改进，使算法迅速跳出局部最优解，使算法向全局最优的方向发展，同时加快了算法收敛的速度。(4)采用的综合评价指标，能满足多个 QoS 要求且负载分布均衡。

表 2 仿真结果

序号	结 果	路 径	瓶颈带宽/bps	时延/μs	平均成功率/%	平均成本	平均时延抖动/μs	综合评价
1	11010000000	AEH	2	2	80	3.5	5	0.13
2	11010000011	AECGH	2	7	80	2.8	4	0.01
3	11001000001	ACEH	4	5	83	2.3	3.6	0.01
4	11001000001	ACGH	2	6	83	2.3	3.6	0.02

参考文献

1 Wang Z, Crow C J.Quality- of- service for Routing Supporting Multimedia Applications.IEEE Journal of Selected Areas in Communications, 1996; 14(7)

2 欧阳森，宋政湘，王建华等.一种快速收敛的遗传算法.计算机应用研究，2003; 20(9)

3 栋朝雅晴，高井昌彰，佐藤義治.一种适应负载分布均衡的路由遗传算法.信息处理学会论文志，1998; 39(2)

4 Xian G F, Zhou J, Jie Y W.QoS Routing Based on Genetic Algorithm.Computer Communications, 1999; 22(15)

5 Ravikumar C P, Bajpai R.Source- based Delay- bounded Multi- casting in Multimedia Networks.Computer Communications, 1998; 21(2)

6 何小燕，费翔，罗军舟等.Internet 中一种基于遗传算法的 QoS 路由选择策略.计算机学报，2000; 23(11)

（收稿日期：2005- 02- 15）