嵌入式操作系统的通用硬件抽象层设计

■同济大学　王力生　仇志付　唐军敏

摘　要　基于嵌入式操作系统硬件抽象层理论，设计一种用于嵌入式操作系统内核开发的通用硬件抽象层平台。通用硬件抽象层能够为嵌入式操作系统内核的设计开发屏蔽硬件平台的特性，提供统一的硬件相关的服务接口，可以使嵌入式操作系统内核的设计开发不依赖于特定的硬件平台，同时开发的嵌入式操作系统内核具有更强的可移植性。

关键词　嵌入式操作系统　通用硬件抽象层(HAL)　BSP　V 开发模式

引　言

为了便于操作系统在不同硬件结构上进行移植，美国微软公司首先提出了将底层与硬件相关的部分单独设计成硬件抽象层美国微软公司提出了将操作系统底层与硬件相关的部分单独设计成硬件抽象层 HAL(Hardware Abstraction Layer)的思想。硬件抽象层的引入大大推动了嵌入式操作系统的通用程度，为嵌入式操作系统的广泛应用提供了可能。然而，目前 BSP 形式的硬件抽象层仅仅能够解决有限的几种操作系统在同样有限的 BSP 所支持的硬件平台上的移植，而对绝大多数需要根据不同嵌入式应用而专门定制的嵌入式操作系统来说能起的作用则非常有限。

1　硬件抽象层原理

1.1　硬件抽象层概念

嵌入式系统是一类特殊的计算机系统。它自底向上包括 3 个主要部分：硬件环境、嵌入式操作系统和嵌入式应用程序。硬件环境是整个嵌入式操作系统和应用程序运行的硬件平台，不同的应用通常有不同的硬件环境；因此如何有效地使嵌入式操作应用于各种不同的应用环境，是嵌入式操作系统发展中所必须解决的关键问题。

硬件抽象层通过硬件抽象层接口向操作系统以及应用程序提供对硬件进行抽象后的服务。当操作系统或应用程序使用硬件抽象层 API 进行设计时，只要硬件抽象层 API 能够在下层硬件平台上实现，那么操作系统和应用程序的代码就可以移植。

这样，原先嵌入式系统的 3 层结构逐步演化为一种 4 层结构。图 1 显示了引入硬件抽象层后的嵌入式系统的结构。

图 1　引入 HAL 后的嵌入式系统结构

在整个嵌入式系统设计过程中，硬件抽象层同样发挥着不可替代的作用。传统的设计流程是采用瀑布式设计开发过程，首先是硬件平台的制作和调试，而后是在已经定型的硬件平台的基础上再进行软件设计。由于硬件和软件的设计过程是串行的，因此需要很长的设计周期；而硬件抽象层能够使软件设计在硬件设计结束前开始进行，使整个嵌入式系统的设计过程成为软硬件设计并行的 V 模式开发过程，如图 2 所示。这样两者的设计过程大致是同时进行的或是并发的，缩短了整个设计周期。

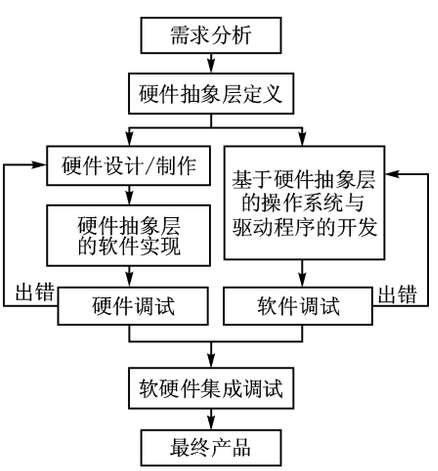

图 2　硬件抽象层引入后的 V 开发模式

1.2 BSP 分析

作为硬件抽象层的一种实现，板级支持包 BSP(Board Support Package)是现有的大多数商用嵌入式操作系统实现可移植性所采用的一种方案。BSP 隔离了所支持的嵌入式操作系统与底层硬件平台之间的相关性，使嵌入式操作系统能够通用于 BSP 所支持的硬件平台，从而实现嵌入式操作系统的可移植性和跨平台性，以及嵌入式操作系统的通用性、复用性。

然而现有应用较为广泛的 BSP 形式的硬件抽象层，完全是为了现有通用或商业嵌入式操作系统在不同硬件平台间的移植而设计的，因此 BSP 形式的硬件抽象层与 BSP 所向上支持的嵌入式操作系统是紧密相关的。在同一种嵌入式微处理器的硬件平台上支持不同嵌入式操作系统的 BSP 之间不仅从组成结构、向操作系统内核所提供的功能以及所定义的服务的接口都完全不同，因而一种嵌入式操作系统的 BSP 不可能用于其他嵌入式操作系统。这种硬件抽象层是一种封闭的专用硬件抽象层。因此，我们提出了为上层嵌入式操作系统内核的开发和构建提供一种开放、通用的硬件抽象层平台，使得在某种硬件平台上的嵌入式操作系统内核的开发能够在支持这种硬件平台的硬件抽象层上进行。

2 通用硬件抽象层总体设计

2.1 通用硬件抽象层的功能结构设计

通用硬件抽象层需要为上层操作系统内核提供统一的硬件相关功能服务；而嵌入式操作系统内核主要的硬件相关部分包括系统启动初始化、任务上下文管理、中断异常管理以及时钟管理。因此，通用硬件抽象层对嵌入式操作系统内核所相关的硬件平台的基本硬件组成部分进行抽象，提供嵌入式操作系统内核硬件平台的相关功能，并设计相应的通用硬件抽象层 API 接口。通用硬件抽象层的总体功能结构如图 3 所示。

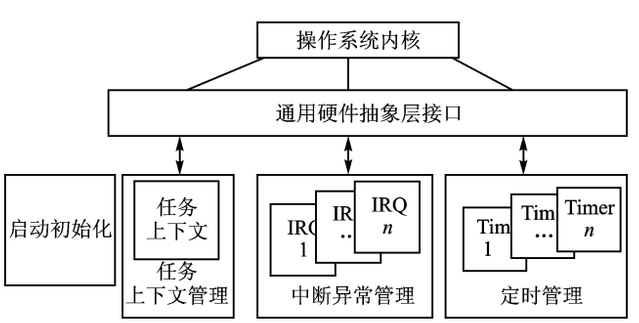

图 3 通用硬件抽象层总体功能结构示意图

(1) 系统启动初始化

启动初始化功能为操作系统的启动和运行提供了必要的软硬件环境。启动和初始化过程中，对硬件平台的直接访问包括对 CPU 内核的寄存器的初始化设置，以及对于起系统控制作用的端口寄存器的设置。通过启动初始化过程，为整个操作系统内核的运行提供了必要的运行环境与基础，隔离了不同硬件平台上嵌入式微处理器总线结构、存储系统结构的差异。

(2) 任务上下文管理

任务上下文管理负责嵌入式操作系统内核中任务管理部分中对任务寄存器上下文的创建、删除以及切换等操作。任务的寄存器上下文是操作系统内核所管理的任务的重要组成部分，是 CPU 内核的寄存器中内容的映像，因此上下文管理的实现依赖于 CPU 内核中寄存器的组织，是与体系结构密切相关的。通用硬件抽象层的任务上下文管理统一定义体系结构中的寄存器上下文的保护格式，提供了任务管理对任务上下文的基本操作的 API 接口。

(3) 中断异常管理

中断异常管理是嵌入式操作系统内核中的重要组成部分。中断异常机制是操作系统内核实现与外部设备通信、任务系统调用、进行出错处理以及能够实现对任务的实时调度的重要手段。因此，硬件抽象层中断系统的管理部分是整个硬件抽象层中的关键。

通用硬件抽象层中为中断异常处理进行了必要的包装，向嵌入式操作系统内核屏蔽底层的中断异常处理；同时，由于中断管理必须涉及对中断控制器的操作。因此，通用硬件抽象层的设计中，将中断控制器控制的外设请求抽象成为统一的 IRQ 设备，嵌入式操作系统通过操作抽象 IRQ 设备来管理外设的中断服务程序以及进行对中断控制器的操作，从而为操作系统内核屏蔽了中断控制器的直接操作。

(4) 定时管理

定时管理负责为操作系统内核中的时钟滴答处理提供必要的定时机制，同时也为内核之外的系统功能提供定时服务，如 TCP/IP 协议栈等。操作系统内核通过时钟滴答处理来执行重要的定时任务(如任务时间的分配、任务运行时间统计、任务定时等待更新等)，因此定时功能是硬件抽象层需要为操作系统内核提供的最为基本和重要的功能之一。

通用硬件抽象层根据对硬件定时器的抽象为操作系统内核提供统一的抽象定时器设备，并且对定时中断服务程序进行了包装，从而使嵌入式操作系统内核直接面对的是统一、通用的抽象定时器设备，通过对抽象定时器的操作来实现定时服务，而不必直接操作硬件定时器。

2.2 通用硬件抽象层的层次结构设计

通用硬件抽象层的设计是为在各种不同硬件平台上的嵌入式操作系统内核的开发提供统一的硬件平台相关的功能，因此这就要求硬件抽象层本身能够易于扩展和移植到不同的硬件平台之上，才能为这种硬件平台上嵌入式操作系统内核的开发提供支持。与硬件平台相关的软件分为体系结构相关以及外围端口寄存器操作相关部分。体系结构相关软件部分能够用于与 CPU 内核体系结构兼容的不同嵌入式微处理器上，而对外围端口寄存器的操作，则每种嵌入式微处理器都不同。因此，通用硬件抽象层功能的实现设计成为图 4 所示的 3 个层次的结构：通用层、体系结构层以及外围层。通过这 3 个实现层次的划分尽可能地实现代码的可复用性。

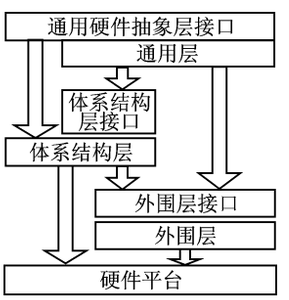

图 4 通用硬件抽象层层次结构示意图

(1) 通用层

通用层是以 C 语言编写的、不涉及体系结构及外围端口寄存器具体操作的、能够通用于各种硬件平台的一层。通用层内包括：对统一的与编译器无关的数据类型、抽象设备的数据结构定义，以及提供给嵌入式操作系统内核的对抽象设备的各种统一的操作服务的接口通用的实现部分。

通用层中抽象设备操作的实现中需要涉及的对 CPU 内核寄存器的操作以及对外围 I/O 端口寄存器的操作，是通过调用体系结构层以及外围层中统一定义的接口进行的。当扩展或移植到其他硬件平台上时，上层无须修改，而只须进行下层替换。

(2) 体系结构层

针对各种嵌入式微处理器 CPU 内核的体系结构，体系结构层需要分别设计实现。体系结构层中对体系结构相关的数据类型以及数据结构进行定义，包括寄存器上下文保存格式的定义以及对中断异常向量起始地址、各种异常和中断处理的入口偏移等，并负责通用硬件抽象层功能中体系结构相关部分的实现。实现的内容主要是对 CPU 内核中各个寄存器的访问，对于中断异常向量表的操作以及底层的中断和异常处理。

体系结构层的实现是按照上层规定的调用接口来进行的，因而针对不同的体系结构，上层通用层无须进行修改。体系结构层中对有关 I/O 端口寄存器的操作通过对外围层接口的调用来实现。

针对某种体系结构设计实现的体系结构层能够通用于 CPU 内核体系结构兼容的嵌入式微处理器的硬件平台上，从而易于硬件抽象层在体系结构兼容的嵌入式微处理器硬件平台上的扩展和移植。

(3) 外围层

外围层是针对各种嵌入式微处理器而分别设计实现的。外围层主要包括对外围 I/O 接口和设备属性的定义（包括中断控制器连接的外设个数、定时器个数等），并且负责对各个外围 I/O 设备端口寄存器的访问操作。外围层的实现需要根据上层定义的接口进行。

通用硬件抽象层的外围层必须提供对存储控制、总线控制、中断控制器、定时器控制器、UART 等基本 I/O 接口和设备的 I/O 端口寄存器的访问功能。外围层是与各种嵌入式微处理器一一对应的，在采用不同的嵌入式微处理器的硬件平台之间，外围层是无法通用的。因此针对新的嵌入式微处理器的通用硬件抽象层的扩展或移植，外围层都需要重新设计实现。

(4) 层次间接口的设计

通用硬件抽象层除了为嵌入式操作系统内核提供统一的功能服务接口外，为了便于扩展和移植到其他硬件平台，还在各层的调用之间设计了统一的调用接口。下层的功能实现需要按照与上层确定的接口规范来进行。其中某些上下层之间的接口，尤其是外围层与上层之间的接口是使用宏定义的方式进行的。宏定义在预编译时进行替换，没有执行时的性能损失。相反，对于底层的操作直接使用宏定义能够提高执行效率，尤其对外围端口寄存器的操作，由于操作本身的执行时间短，而一般函数调用则需要返回地址、参数压栈等过程。这些开销可能超过这些 I/O端口寄存器的访问时间，使用宏定义则没有调用开销，从而能够直接实现接口对底层端口寄存器的访问而不损失操作的效率。

参考文献

[1] Fernando Friedrich L, John Stankovic, et al. A Survey of Configurable, Component-based Operating System for Embedded Applications, IEEE MICRO, 2001,5~6:54-68.

[2] Labrosse J. 嵌入式系统构件. 第 2 版. 袁勤勇，黄绍金，唐青，译. 北京：机械工业出版社，2001.

[3] 罗蕾. 嵌入式实时操作系统及应用开发. 北京：北京航空航天大学出版社，2005.

[4] 王涛，张伟良，冯重熙. 嵌入式系统硬件抽象层原理与实现. 电子技术应用，2001(10).

王力生（教授、硕士研究生导师），主要研究方向为嵌入式系统及计算机网络；仇志付、唐军敏（硕士研究生），主要研究方向为嵌入式系统。

（收稿日期：2006-05-26）