Blackfin在线培训课程

课程单元：Blackfin®系统服务程序
主讲人：David Lannigan

第9b节：API
第10章：结束语
第10a节：服务程序文档
第10b节：总结
第10c节：更多信息

第1章：简介

第1a节：课程安排

大家好，我是模拟器件公司（ADI）的平台工具工程师，我的名字叫David Lannigan。今天我们将讨论面向Blackfin处理器家族的系统服务程序。

今天，我们首先要讨论什么是系统服务程序，以及系统服务程序的优越性。然后，我将逐一介绍每个系统服务程序提供的高级功能，并概述各个系统服务程序的API。其间，我将穿插一些例子，演示如何利用系统服务程序执行电源管理、DMA管理、标志控制和回调管理等。

第2章：系统服务程序

第2a节：概述

那么，什么是系统服务程序？其实，系统服务程序就是一个为嵌入式系统提供各种常见功能的软件库。利用系统服务程序，用户可以便捷、高效地接入Blackfin处理器的硬件子系统，具体而言包括锁相环路（PLL）、直接内存存取（DMA）、中断控制器、标志控制器和定时器等等。我们还提供了一个名为“延迟回调”的服务程序，以帮助用户提高嵌入式系统的中断性能。C语言程序和汇编程序均可调用这个软件库。虽然用户可以自由选择C语言或汇编语言，但如果要使用汇编语言，则必须遵循C运行时间调用规则。所有Blackfin处理器均采用一套通用的API，包括ADSP-BF531、BF532、BF533、BF534、BF536和BF537等单核处理器以及BF561双核处理器。各种应用和设备驱动程序均可使用这些系统服务程序。这些服务程序实际上是可用于任何系统的公用程序，既可在未安装RTOS的独立式环境下运行，又可用于VDK环境。也就是说，如果用户的系统安装了VDK实时操作系统，那么，我们的系统服务程序和API，也能够支持该系统。

第2b节：系统服务程序的优越性

常有用户问我，“使用系统服务程序究竟有什么好处？”我认为，系统服务程序最大的优越性就是能够帮助用户快速将产品推向市场。能够利用现成的代码，干嘛不坐享其成？系统服务程序中包含的软件都是业经测试、稳定可靠的，用户可以轻松上手，直接使用这些程序。利用服务程序，用户可以轻松整合这些模块化软件。服务程序将负责管理硬件资源，因此，将不同开发人员编写的两个程序块整合起来变

得轻而易举。此外，我前面提到过，所有的Blackfin处理器都采用完全相同的API，因此，不论是单核处理器还是双核处理器，其API都一样。我们的Blackfin处理器系列范围广泛，而且还将推出更多型号。由于所有这些Blackfin处理器都采用完全相同的API，因此，用户可以将应用从某个处理器轻松导入另一个处理器。我们的设备驱动程序库也基于这个系统服务程序。因此，用户可以利用系统服务程序，便捷地调用设备驱动程序库。我们还提供了所有的源代码，以便用户通过研究这些代码，深刻理解其功能，并在这个过程中进一步学习。通过研究源代码，用户可以深入理解DMA控制器的工作原理，以及如何调用和管理DMA控制器，有哪些寄存器等等。

第2c节：系统架构

这个框图演示的是一个典型的应用系统架构。在最上面，是用户代码；用户可以选择为系统配置RTOS。然后，是设备驱动程序，负责管理输入和输出系统的数据流。最下面是系统服务程序。服务程序之所以会在这个框图的最底部是因为我们不需要RTOS，RTOS是一个可选项。设备驱动程序、应用、甚至VDK或者任何其他类型的RTOS均可利用这些系统服务程序。

目前，我们提供了8种系统服务程序，包括负责控制Blackfin处理器上的SDRAM控制器的EBIU、动态电源管理、中断控制、DMA管理器、面向Blackfin处理器上的通用IO的标志控制、延迟回调、面向内核定时器、看门狗定时器和通用定时器的定时器控制，以及面向BF534、BF536和BF537等处理器的端口控制。稍后，我将详细解释什么是回调，以及什么是延迟回调。我将高度概括地逐一介绍这些系统服务程序，介绍其API，并举例演示如何轻松使用这些服务程序。

第3章：动态电源管理与EBIU

第3a节：电源管理概述

Blackfin处理器具备一个功能强大的电源管理系统。通过控制内部电压调节器中的锁相环路（PLL），可以设置多种不同的功耗和性能选项。对于Blackfin处理器或任何处理器，其功耗均为工作频率和电压平方的函数。通过控制工作频率和电压，可以控制处理器的功耗。然而，对PLL和电压调节器进行编程没有想象的那么简单。需要对寄存器进行编程、生成中断、等待PLL稳定下来，这其实是一个相当复杂的过程，也非常容易出错。动态电源管理服务程序提供了一个函数，只要调用这个函数就可以改变系统和内核的时钟频率（SCLK和CLCK）、改变运行模式、甚至改变电压电平。通过一次函数调用，就能改变处理器的运行模式，例如工作、待机、睡眠、深度睡眠和冬眠等运行模式。用户还可以改变内核时钟频率和系统时钟频率。当内核和系统时钟频率改变后，电源管理服务程序将自动调节电压电平，以便尽可能最大限度地节省功耗。例如，如果用户选择了一个较低的内核CCLK和系统SCLK频率设置，那么，电源管理服务程序自动将电压电平降至该频率设置下的最低值。相反地，如果要最大限度地提高处理器的性能，则可以调高其CCLK和

SCLK设置，从而使服务程序自动调高电压电平。同样地，也可以通过控制电压来调节时钟频率设置。例如，将电压设定为0.9伏，那么，通过一个函数调用，即可将CCLK和SCLK设置为0.9伏电压下在最高值。此外，不论用户在什么时候改变了任何时钟频率设置，电源管理服务程序都会自动调用EBIU服务程序，对SDRAM进行相应的设置。

第3b节：EBIU概述

下面，我们介绍EBIU服务程序。EBIU服务程序其实就是Blackfin处理器的SDRAM控制器的控制器。它包含一个运算逻辑，可以在SCLK频率发生改变后，自动计算出新的SDRAM设置值。EBIU服务程序将与电源管理服务程序配合作用，因此，对用户而言，只需要更改CCLK和SCLK设置，EBIU服务程序就会自动调节SDRAM设置。要利用EBIU和电源管理服务程序，应用必须首先初始化EBIU服务程序和电源管理服务程序，然后，就可以根据需要调用适当的函数和电源管理服务程序。例如，设置处理器的工作频率、设置特定电压下的最高工作频率，或者改变处理器的运行模式。

第3c节：动态电源管理API

这张幻灯片列出了动态电源管理服务程序的API。这里我们不会详细讨论API，不过，需要注意的是，在最上面列出的这些初始化函数和终止函数。每个服务程序都具备一个初始化函数和终止函数。要调用服务程序，必须首先调用其初始化函数。在服务程序调用完毕后，还需要调用其终止函数。一般而言，在嵌入式系统中，系统将以无限循环方式运行，不需要调用终止函数。下面这些是用于控制工作电压和频率的函数，以及设置运行模式的函数。

第3d节：EBIU API

在EBIU API中，最重要的函数依然是初始化函数和终止函数。在与电源管理服务程序配合作用时，应用只需要对该服务程序进行初始化，然后，它将自动运行。

第3e节：举例

这个例子演示的是一个在Blackfin BF537 EZ-KIT Lite评估板上运行的应用。这个主程序的基本任务是初始化电源管理服务程序，然后，更改处理器的SCLK和CCLK频率设置，并切换不同的运行模式，包括工作、待机等等模式。这个应用将调用电源管理服务程序，然后，电源管理服务程序将控制PLL和电压调节器，并调用EBIU服务程序，自动调节SDRAM设置。好了，打开VisualDSP®。我们将通过这个例子，逐行演示如何使用电源管理服务程序。首先，我们要初始化电源管理服务程序。然后，就可以调用其中的任何API函数。现在，执行这个初始化函数。右侧窗口中显示的是CCLK频率、SCLK频率和电压控制振荡器频率的局部变量。这里，最重要的是CCLK频率和SCLK频率。接下来，我们要调用一个被称为

“GetFrequency（设置频率）”的动态电源管理服务程序。这些是用于保存CCLK频率和SCLK频率设置的位置，这些设置将显示在右侧栏的最上方。好了，执行这个函数，这里显示我们的CCLK频率为250 Mhz，SCLKK频率为50 Mhz，电压控制振荡器的运行频率也为250 Mhz。通过执行下面这个函数，我们将命令电源管理服务程序将CCLK频率提高至600 Mhz或者将SCLK频率提高至120 Mhz。现在，单步执行这个函数，命令电源管理服务程序按要求更改这些频率设置；然后，我要进行询问，以确认它是否遵从了这个命令。果不其然，CCLK频率为600 Mhz，SCLK频率为120 Mhz。接下来，我们要将CCLK频率设置为500 Mhz，并命令电源管理服务程序最大限度地提高SCLK频率。也就是说，希望在内核频率为500 Mhz的条件下，尽可能将SCLK设置为最高。请记住，我们不能分别设置这两个频率，它们之间存在一定的关系。如果将CCLK频率设置为特定值，那么，SCLK值只能在特定范围内。因此，这个代码行的意思是告诉电源管理服务程序，将SCLK调至最高，但是CCLK必须为500 Mhz。好了，我们执行这个函数，要求服务程序计算出符合要求的频率值。它将CCLK设置为500 Mhz，并将SCLK设置为125 Mhz。CCLK频率和SCLK频率的组合取决于时钟输入值和处理器采用的振荡器。如需了解关于这方面的更多信息，请参阅硬件参考手册。

下面我将演示如何改变处理器的工作电压。我们要命令电源管理服务程序，将电压调节器将至0.85伏，并让处理器以0.85伏电压下的最高频率运行。所以，我们要调用“SetMaxFrequency（设置最高频率）”函数，设置电压。然后，我要再次询问CCLK和SCLK频率的值。询问结果是，当电压为0.85伏时，处理器的最高CCLK频率为250 Mhz，也就是当前的运行频率，而SCLK频率则为83 Mhz。

下面演示如何将处理器的运行模式从工作改为待机，以及睡眠模式等等。现在，我们正以工作模式运行，接下来，我们要执行这个函数，使处理器进入待机状态。现在，这个函数运行完毕，我们已经进入待机状态。接着，我要询问它是否真的实现了我的要求。鼠标光标所在位置就是电源模式，这里显示我们正处于待机模式下。现在，我要询问工作频率；结果是，CCLK为25 Mhz，SCLK也为25 Mhz。请记住，在待机模式下，电源管理服务程序将不理会PLL，因此，处理器实际是以时钟输入频率运行。现在，我们处于待机模式；假如我们要返回工作模式，并以更高性能运行，那么，我们需要命令处理器返回工作模式并启用PLL。如果现在进行询问，结果将是我们已经返回了工作模式。询问处理器的工作频率，这里显示CCLK为250 Mhz，SCLK为83.3 Mhz。接下来，我要命令它将CCLK和SCLK均设置为最高，最大限度地提高处理器的性能。然后，再询问，结果是CCLK为600 Mhz，SCLK为120 Mhz。总的来讲，电源管理服务程序的工作机制非常简单，用户可以轻松改变处理器的工作频率和电压电平。

第4章：中断管理器

第4a节：概述

下面，接着讲解幻灯片。现在，我们要讨论中断管理器服务程序。对其他服务程序而言，中断管理器服务程序就像是中央。许多其他服务程序都要调用中断管理器，因为中断是嵌入式应用编程的关键组成部分。中断管理器服务程序将直接利用Blackfin处理器的内核事件控制器和系统中断控制器。请记住，内核事件控制器的工作方式与其他所有具备向量表的通用处理器一样，如果它断言某个向量或者中断，则表示指向表中的某个向量地址。此外，之所以还会涉及Blackfin处理器的系统中断控制器，是因为Blackfin处理器具备丰富的外设。如果处理器具备大量外设，就会需要使用中断向量组。系统中断控制器类似于一个门电路，它将决定是否将中断传输回内核和服务程序。中断管理器将根据用户的要求，操纵内核事件控制器和系统中断控制器。我们可以将中断处理程序挂接至各种中断向量组，或从中断开。如果只有一个中断向量组，则可以实现处理程序链接，即在这个向量组中挂接多个中断处理程序。以错误为例。谁都希望错误越少越好。我们不想为处理器中的每一个潜在错误源单独提供一个中断向量组，因此，我们希望将多个错误挂接至一个中断向量组；处理程序链接就可以帮助我们实现这一点。假如，系统中同时运行着一个PPI设备驱动程序、SPI设备驱动程序和UART设备驱动程序，那么，我们可以将这些外设发生的所有错误映射到一个中断向量组中，并允许每个设备驱动程序将其处理程序挂接到这个链接中。

接下来是系统中断控制器。我刚刚说过，系统中断控制器类似于一个门电路，负责控制是否将外设中断传递给内核事件控制器。中断管理器服务程序具备一个C函数调用接口，允许用户指定希望将哪些中断从系统中断控制器传递至内核事件控制器。用户可以将外设中断映射至不同的中断向量组。如果希望某个外设中断具备高优先级，那么，可以将其映射至高优先级中断向量组。用户可以启用和禁用中断，决定是否将其传递至内核事件控制器，也可控制外设中断是否将处理器从睡眠模式唤醒。我们还提供了公用程序函数，以保护关键代码段。假如用户希望确保自己在更改某段数据时，它不会被应用中的任何其他程序修改，那么，可以借助我们提供的关键代码段开始函数和关键代码段终止函数，将这段敏感代码和这些函数调用包括起来。我们还提供了一个中断屏蔽寄存器控制函数集，允许用户设置和清零中断屏蔽寄存器中的位。我们的中断管理器服务程序中包含了这些彼此独立的控制函数，以支持各种不同的操作系统，如VDK、独立式系统和几乎任何其他类型的RTOS。RTOS不喜欢用户在自己毫不知情的情况下更改中断屏蔽寄存器。因此，通过这些独立的控制函数，中断管理器就可以在更改IMASK寄存器值时，随时通知操作系统。此外，对于关键代码段保护，不同的系统采用了不同的机制来保护关键代码段。得益于这些相互独立的控制函数，用户可以将应用从一个操作系统导入另一个操作系统，而不会影响应用性能。有些RTOS通过禁用调度程序来保护关键代码段，而另一些RTOS则通过禁用中断来保护关键代码段。系统服务程序与操作系统之间存在很强的相关性，应用需要通过RTOS来调用系统服务程序。

第4b节：API

这些是中断管理器的API。上面这些是中断管理器服务程序的初始化函数和终止函数。然后是内核事件控制器函数，可以用于将中断处理程序挂接至中断链接或从中断开。还有系统中断控制器，可用于改变外设中断与中断向量组（IVG）之间的映射关系。用户还可以设置处理器的唤醒条件，以及检测某个外设中断是否断言为中断。在最下面，是我们在前面讨论过的公用程序函数，关键代码段开始函数和结束函数，以及用于设置和清零IMASK位的控制函数。

第4c节：举例

在这个简单的例子中，我将演示如何利用中断管理器和实时时钟。在主程序中，我们将实时时钟设置为每分钟生成一个中断。每隔一分钟，它就会生成一个中断。我们将调用中断管理器，要求它允许将实时时钟生成的中断传递至内核事件控制器。此外，我们要将中断处理程序挂接至内核事件控制器。这个简单的例子演示了如何利用中断服务程序，处理中断。好了，首先打开VisualDSP。关闭刚才演示电源管理所用的项目，打开中断管理举例项目，并构建这个项目。这就是主程序。我们将逐行解释这个程序，不过我首先要概要介绍整个主程序。首先，设置预定标器，以便时钟以正确的频率运行。我们要清除任何未决中断，然后，启用实时时钟，以在分寄存器发生改变时生成一个中断；也就是说，实时时钟将每分钟生成一个中断。接着，我们要询问中断管理器服务程序，查明实时时钟映射至哪个中断向量组。如果需要，也可以改变该映射，不过，在本例中我们将采用这个默认映射。现在，我们要调用钩子函数，挂接处理程序。然后，我们要启用中断。利用这4个简单的函数，我们可以启用中断并处理中断。好了，我们直接跳到下面的中断处理程序。首先，我们要确认该中断是不是面向这个处理程序的。还记得我们刚刚讨论过的中断处理程序链接吗？在利用处理程序链接时，需要注意的是，可能有面向其他处理程序的中断错误地调用这个处理程序。必须询问该处理程序支持的器件，查明这个中断是不是发送给该处理程序的，因为在处理程序链接中还有其他的处理程序。所以，我们先要核实这个中断是不是发送给这个处理程序的，弄清楚之后，就可以告诉中断管理器服务程序，已经处理了这个中断，该中断确实是发送给这个处理程序的，不需要再调用处理程序链接中的其他处理程序。

接下来，回到上面的程序。我们刚刚跳过了初始化步骤。现在，我们要通知实时时钟，我们要启用预定标器，让实时时钟以正确的频率运行，每隔60秒更新一次状态。然后，我要清除任何未决中断，以确保当前没有任何其他中断。然后，我们将命令实时时钟每分钟生成一个中断。现在，我要询问中断管理器服务程序，查明这个实时时钟映射至哪个中断向量组。执行这个函数，这里显示实时时钟映射至IVG8中断向量组。现在，我要将我的处理程序挂接至IVG8。这就是实现这个目的的调用函数。这是需要挂接的中断向量组，这是我们的处理程序的地址，这是所谓的客户端句柄。每一次调用处理程序时，都要将客户端句柄传递给处理程序，这样，系统就可以知道是哪个器件在调用中断处理程序。最后一个参数表明了是否需要启用中断嵌套。借助中断嵌套特性，应用可以通过更高优先级的中断，中断另一个较低优先级的中断。可以将这个参数设置为“真”或“假”，在本例中，我们将

其设置为假。执行这个代码段，现在，处理程序已经挂接到处理程序链接中。接下来，我们要命令系统中断控制器，允许将实时时钟中断传递至内核事件控制器。不过，在此之前，我要在中断处理程序中插入一个断点，以便在中断结束时，让处理器停止运行。按下F5键，处理器开始运行，并迅速停在中断处理程序上。在执行其他函数时，分钟中断结束。现在，我要检查它是不是一个实时时钟中断，结果是，接下来，我们要将这个中断从实时时钟中清除。清除这个未决中断，并向中断管理器报告已经处理完毕该中断。现在，单击工具栏中的“运行”图标，让处理器开始运行。其实处理器只是呆在这个循环中，等待下一个中断发生。在接下来的60秒中，将出现另一个中断，然后再一次达到断点。两次相隔约60秒。再一次检查这个中断是不是发送给这个处理程序的。确实是。于是，我们再次清除该中断，并向中断管理器报告已经处理完毕该中断。就是这样。这个例子简单地概括了如何使用中断服务程序；如何命令外设生成一个中断；如何将中断传递至系统中断控制器；如何将处理程序挂接至内核事件处理器。非常简单的操作步骤。利用3个函数调用，就完成了这操作。挂接处理程序，启用处理程序，并将中断从系统中断控制器传递至内核事件控制器，非常简单的过程。

第5章：延迟回调

第5a节：概述

我们接着来讲解幻灯片。在开始讨论延迟回调服务程序之前，我想首先应该介绍什么是回调。回调其实就是在一个正常的执行流程之外，发起一个函数调用，例如，响应某个异步事件。回调也是一个标准的C语言可调用函数，它将根据发生的事件，采用某种行动。回调分为两类，实时回调和延迟回调。实时回调是在收到中断后立即发起的回调，因此，实时回调通常在硬件中断时立即执行。而延迟回调则在软件中断时执行。两者之间的差别非常重要。在执行实时回调时，即在硬件中断时执行回调，较低优先级的中断将被禁用，直至该回调函数执行完毕。而在执行延迟回调时，由于回调服务程序以较低优先级执行回调，因此，较低优先级的中断将保持启用状态。延迟回调管理器提供了一种机制，可以将在高优先级硬件中断时发起的回调，延迟至较低优先级的软件中断时。也就是说，在发生某个事件时，延迟回调管理器通常会将该回调列入排队，换句话说，就是通过一个注释，阐明其将在稍后进行回调，并退出高优先级中断服务程序。然后，在开始执行软件中断时，延迟回调管理器将对刚才的事件进行回调。借助延迟回调管理器，用户可以创建多个回调排队。每个排队都具备各自不同的中断向量组级别，也就是说，每个排队都拥有不同的优先级。在每个排队中，用户可以指定该排队中的各个回调的相对优先级。例如，如果有一个回调排队是在IVG14中断向量组执行，那么，用户可以设置一个紧急回调，只要开始执行这个回调排队，就首先执行这个紧急回调，然后再执行其他也映射至IVG14中断向量组的优先级较低的回调。我们尽量利用操作系统提供的所有资源；在一个独立式操作系统中，我们一般是在运行标准用户代码之前，执行IVG14回调。而在一个基于VDK的操作系统中，我们则通过软件中断线程执行回调。

第5b节：API

用于调用回调服务程序的API相当简单。首先还是初始化函数和终止函数，然后是用于打开排队、关闭排队以及控制排队的函数，例如，将回调映射至特定中断向量组。再下面，是用于将回调加入排队的函数；在此之后，用户也可以通过调用这个函数将其从排队中删除。一会儿讨论标志服务程序时，我将以标志服务程序的回调为例，进行演示。

第6章：DMA管理器

第6a节：概述

我们还提供了DMA管理器服务程序。Blackfin具备一个功能强大的DMA控制器，既可支持外设DMA又可支持存储器间DMA。外设DMA主要用于设备驱动程序；设备驱动程序利用外设DMA向Blackfin处理器内存读写数据。DMA管理器负责控制和调度外设DMA和存储器间DMA。设备驱动程序在传输DMA数据时，将适当地调用DMA管理器。应用也可以利用DMA管理器在不同存储空间转移数据，或者利用存储器间DMA，将数据转移至不同的存储位置。此外，DMA管理器还可以控制各种DMA通道映射。每个DMA通道都具备特定的优先级。DMA管理器可以将不同优先级的通道映射分配给不同外设；例如，可以通过通道映射，将PPI DMA通道的优先级设置为高于UART通道的优先级。我们还提供了一种用于流量控制的机制，可以操纵内核接入DMA总线或内存总线与DMA控制器接入内存总线之间的仲裁逻辑。DMA管理器可以支持所有主要的控制器模式，如描述符链接——不论描述符的长短，和自动缓冲——也就是DMA管理器中的循环缓冲。此外，我们还实现了高级内存拷贝函数，允许通过DMA而不是内核接入，将数据从内存中的某个位置转移至其他位置。我们可以充分利用DMA控制器提供的一维和二维传输机制，因此，我们可以实现线性或XY型数据传输。此外，还可以在结束时执行回调。假如要调度一次内存转储，并且希望在数据转移结束后收到通知，就可以利用回调。利用回调机制，可以发出事件通知，既可以是实时通知，也可以是延迟通知，主要取决于用户的要求。

第6b节：API

这些就是DMA管理器的API。和所有其他服务程序一样，也有初始化函数和终止函数。这些是用于管理每个通道的函数，包括打开、关闭、控制、将描述符加入某个通道中的排队，以及高级内存控制函数等。此外，利用内存数据流，用户可以将数据从某个存储空间转移至其他存储空间，或者从存储器中的某个位置转移至其他位置。再下来是我刚才提到过的通道映射，可用于将某个通道映射至其他的外设。

第6c节：举例

下面，我们演示使用内存DMA服务程序的例子。在本例中，一个主程序要将一段数据从源位置拷贝至目标位置。我们将打开内存数据流，并命令DMA管理器将该数据从内存中的这个位置移动至另一个位置。DMA管理器将实现这个转移过程，对Blackfin处理器上的DMA控制器进行必要的编程，以执行这个移动操作。然后，DMA管理器将回调这个主程序，并报告转移已经顺利完成。在回调函数中，主程序将调度另一个内存转储。在视频处理应用中，这种情况很常见。在视频处理应用中，通常需要处理大量的数据，这些数据都保存在SDRAM中，因此，需要将这些数据映射到L1片上内存中，以进行处理，比如压缩，然后，再将处理完毕的数据发送至诸如录制设备等其他器件。要将数据从外接存储器或者SDRAM存储器映射至片上内存，通常需要使用DMA“memcopy”（内存拷贝）函数。

现在，我们在VisualDSP中演示本例。我要打开一个项目，然后构建这个项目。在右侧栏，显示了一些存储位置，分别是源存储位置和目标存储位置。我将这个窗口调大一点，以便大家看得更清楚。在右上角，是源存储位置，而在右下角，则是目标存储位置。我们要利用“memcopy”函数，将上面这个源存储位置中的内容转储到下面这个目标存储位置中。现在，执行这些代码，看看结果会怎么样。首先，我们要初始化系统服务程序，然后，我们就可以使用任何服务程序API函数。接着，我要在上面这个源数据中创建一些哑值。从0数到32，这样就在这个存储器中创建好了需要拷贝到目的存储位置的数据。好了，运行。我们可以看见，已经有一些值保存在这个存储位置。我把窗口拉过来一些，以便大家看得更加清楚。现在，我们要调用DMA管理器，并命令DMA管理器打开一个内存数据流。BF531、BF532、BF533、BF534、BF536和BF537处理器都可提供2个内存数据流，而BF561双核处理器则可提供4个内存数据流。现在我们要命令DMA管理器打开一个数据流，以便使用。接着，我们要运行这个函数，不过首先我要解释这些参数。第一个参数是DMA服务程序的句柄。应用在初始化DMA管理器时，将向其提供一个句柄，用于标识该服务程序。我们要利用该服务程序，打开内存数据流。这是我们即将打开的内存数据流的ID——“memDMA index 0”。下面这个参数是客户端句柄，服务程序在回调应用时，必须向应用传递其客户端句柄。应用可以为客户端句柄任意赋值，对DMA服务程序而言，这个值无关紧要，只要应用自己理解这个值的含义就可以了。总之，可以是任意值。客户端句柄是回调函数中的一个参数。打开内存数据流后，DMA管理器将向应用提供该数据流的句柄。DMA服务程序通过这个句柄可以识别出这个内存数据流，因此，只要利用DMA服务程序调用这个内存数据流，就要将该句柄返回给DMA管理器，以通知它我们正在做什么。

为了让本例简单明了，我们要采用实时回调，所以，调用句柄服务程序的值设置为“空”（NULL）。现在，运行这个函数，告知DMA管理器，我们需要使用DMA服务程序，我们已经打开了内存数据流，并且DMA管理器已经找到了需要拷贝的内存数据流。

这就是利用DMA管理器执行内存拷贝的函数。这是DMA管理器向我们提供的内存数据流的句柄。这是目标位置，这是源位置，每个元素的宽度为1个字节8位。数据

宽度为32位，源阵列和目标阵列都是32位数据。执行调用后，DMA控制器将移动该数据，这时，我们只需要呆在一个无限循环中就可以了。DMA管理器在完成转储后，将向我们发出回调。现在，我要在回调函数中添加一个断点，这样，在DMA控制器完成数据转储后，将回调我们，报告它已经圆满完成任务。现在，按下F5键，开始执行这段代码，接着，我们停在了回调函数的位置。现在来查看存储位置窗口，我们会看见目标存储位置中的值与源存储位置中的值相同。DMA控制器已经成功地将源阵列中的所有数据拷贝至目标阵列，这个过程是利用DMA在后台完成的，也就是说，是与内核代码执行并行执行的。现在，我们正停在回调函数上。DMA控制器已经完成了数据转储，正要通知我们，以便我们执行其他任务。记得吗，这个客户端句柄就是刚才提到的“0x12345678”；客户端句柄是回调函数中的一个参数。这个句柄非常有用。如果应用有多个回调函数，那么，就可以为每个回调函数设置不同的客户端句柄，这样，就可以知道所发生事件的来龙去脉。现在，我们要测试这个事件。所有回调都有3个参数，第一个参数是客户端句柄，这是一个仅对客户端或应用有意义的参数。第二个参数是实际发生的事件的事件ID，在本例中，事件ID是“0x30001”，是一个已经处理了其描述符的DMA事件。最后一个参数取决于发生的具体事件，是一个会随发生的事件而改变的参数。例如，对于设备驱动程序，在缓冲区处理完毕事件中，最后一个参数可能是指向业已处理完毕的缓冲区。而在本例中，对于DMA控制器，当DMA描述符处理完毕时，返回的参数是指向目标地址的指针。这个地址是“0xFF801144”，毫无疑问，这就是目标地址。通过这种简便的方式，服务程序可以通知应用，事件已经结束或发生，特别是哪个存储位置受到了影响。我做了一个小测试，以确认这个DMA事件确实已经完成了内存转储。

现在，在这个回调函数中，我们调度了另一个内存拷贝。这个步骤可谓易如反掌，并且演示了回调函数的一些功能，即，应用还可以在中断时启用另一个转储。例如，在视频数据处理中，通过这种方法可以通知应用，有一个存储位置需要进行处理。应用可以在处理原有的存储位置的同时，调度转储另一个存储位置。下面，我们将再次执行同一个转储，以证明我们确实是在对其进行转储。首先，我们要更改目标地址中保存的这些值。然后，我们要再次执行拷贝。DMA服务程序会将源数据拷贝至目标数据存储位置。好了，按下F5键，开始执行，数据拷贝完毕。在我们创建的这个循环中，我们将不断地将数据从这个源位置拷贝至目标位置。这个机制非常简单，演示了利用DMA拷贝内存的过程。很简单，打开一个数据流，然后，为其设置一个DMA描述符，阐明存储位置、转储任务、数据长度以及在完成任务后回调主程序。非常简单的过程，通过两个函数，就能利用DMA控制器。简单是系统服务程序的关键特性之一。服务程序旨在令应用运行更加快捷。

第7章：标志控制

第7a节：概述

我们继续讲解幻灯片。Blackfin处理器具备所谓的通用输入和输出标志或通用可编程标志。在嵌入式控制系统中，标志非常有用。标志其实就是可以配置为输入和输

出的管脚。通过将这些管脚设置为不同的值，可以触发外接器件或者检测外接器件发生的活动。标志控制服务程序提供了一个非常简单的接口，用户可以操纵和配置这些管脚，检测这些管脚的值，设置标志的方向，如输入标志或输出标志。用户可以将标志值设置为逻辑1和逻辑0，或者在二者之间进行切换。此外，标志控制服务程序还支持回调。在达到触发条件时，将自动调用回调函数。触发条件由用户阐明或指定。标志控制服务程序可以支持电平敏感转换条件，例如，标志管脚升高或降低；或者边缘敏感转换，实际就是检测边缘升高或降低；或者，用户也可以设置为不论边缘升高或降低，均可触发调用回调函数。不论在什么情况下使用回调，都可以选择使用实时回调或延迟回调，标志控制服务程序也不例外。标志控制服务程序可以在某个可编程标志达到特定触发条件时，发起实时回调或者延迟回调。

第7b节：API

这些是标志控制服务程序的API。最上面同样也是初始化函数和终止函数。下面的这些是用于控制各个独立标志的函数。可以打开标志、关闭标志、设置标志方向、设置标志值、清零标志值、切换或检测标志值以及标志方向。再下面，是回调函数，可以为标志设置回调函数，触发条件等等。

第7c节：举例

现在，举例说明标志控制服务程序。我们要在主程序中将一个标志配置为输入标志，并为其设置一个回调，当该标志的状态发生改变时，将回调主程序。标志控制服务程序将负责控制Blackfin处理器的可编程标志子系统。在本例中，我们是在EZ-KIT Lite评估板上运行，我们要控制的标志实际映射至EZ-KIT Lite评估板上的一个按钮。因此，当我按下EZ-KIT Lite评估板上的这个按钮时，就会形成触发条件，从而调用回调函数。我现在打开了VisualDSP，关闭刚才那个项目。打开一个标志项目，并构建该项目。这是我们的源文件。首先，我们要初始化系统服务程序，执行这些代码。然后，我们要打开标志PF2。PF2映射至BF537 EZ-KIT Lite评估板上的一个按钮。好了，打开这个标志，并命令标志控制服务程序，将这个标志设置为输入标志。现在，我们要命令服务程序，为这个标志设置一个回调。下面这个就是回调函数，我们要为标志PF2设置这个回调。这是用于触发回调的中断，在本例中，就是标志中断A。我们要触发的事件就是在标志边缘升高时，发起回调。当标志从0变成1的时候，就会触发这个回调。下一个参数为真。这个参数既可以为真，也可以为假，它是处理器的唤醒标志。利用标志服务程序，用户可以配置该参数，当处理器处于睡眠模式或者深度睡眠模式时，如果达到触发条件，应用将唤醒处理器，使之返回工作模式。

接下来，是一个客户端句柄。和前面的回调例子一样，客户端句柄是回调函数中的一个参数。我们将采用实时回调，如果要采用延迟回调，只需要将这个客户端句柄传递给标志管理器需要使用的延迟回调服务程序。最后，是回调函数的地址。在执行这个函数调用之前，我们先来看看这个回调函数。在调用这个函数时，我们将收到一个客户端句柄，也是刚才提到的“0x12345678”。事件参数将表明，标志已经

触发，在本例中这个指针自变量实际就是已经触发的那个标志的标志ID。现在，我们要设置回调，然后返回这个函数，并执行。好了，已经挂接了这个回调。回调函数设置完毕，接下来，我们只需要呆在这个循环里，等待标志达到触发条件。现在我们要在回调函数中插入一个断点。然后，按下F5键，处理器开始运行，主程序呆在这个“while”循环里，等待发生回调。现在，只要我按下EZ-KIT Lite评估板上的这个按钮，就能达到触发条件，并调用回调函数。现在，我按下这个按钮，我们立即停在了回调函数上。不出所料，标志ID是PF2，也就是说，我们将PF2标志配置为生成回调。此外，由于该标志是映射至EZ-KIT Lite评估板上的这个按钮的，所以当标志从低位变为高位时，就达到了我们设置的触发条件，从而生成回调。这是一个非常简单的方法。在本例中，我们仅使用了3个函数，就配置了一个标志，或者应该说是打开一个标志，将其配置为输入标志，并为其设置一个回调。易于使用是系统服务程序的目标之一。

第8章：定时器控制

第8a节：概述

继续讲解幻灯片。下面，我们讨论定时器控制。Blackfin处理器具备3种类型的定时器：内核定时器、看门狗定时器和几个通用定时器。内核定时器就像是一台心跳监视器，是一个定期生成一个增量的节拍型系统定时器。看门狗定时器通常用于标识超时事件。有时候，看门狗定时器可以用作事故自动刹车开关，因此，需要经常重设，否则会引发其他事件。对于Blackfin处理器，这个事件可能是重设事件，即重设看门狗定时器，或者是一般中断，或者任何其他类型的事件；看门狗定时器自动重设，而不会引起其他操作。Blackfin处理器具备多种通用定时器，可以脉宽捕捉（CAP）模式或脉宽调制（PWM）模式运行。PWM模式非常适用于电机控制，通过同时启用或禁用成组的定时器，辅助电机控制应用。利用定时器控制服务程序，可以调用Blackfin处理器为所有这些定时器提供的各种特性。此外，和其他服务程序一样，定时器服务程序也可支持回调，用户可以将定时器服务程序配置为当定时器终止时，触发一个回调函数。取决于用户的选择，该回调函数既可以是实时回调，也可以是延迟回调。

第8b节：API

这些是定时器控制的API，上面同样也是初始化函数和终止函数。这些是用于打开、关闭和控制这些不同类型的定时器的函数，包括内核定时器、看门狗定时器和通用定时器。再下来是用于为定时器设置和删除回调函数的API。这些API也非常简单，易于使用。

第9章：端口控制

第9a节：概述

最后一种服务程序是端口控制服务程序。端口控制的作用是控制分配Blackfin处理器的多路复用管脚。Blackfin处理器拥有众多外设，而外设一般都具备许多管脚。如果为所有外设单独配置管脚，那么，处理器上将布满管脚，并增加成本，因此，我们采用了多路复用技术，将不同外设发出的多路信号，复用到一个管脚中。换句话说，一个管脚在某种情况下可以用作PPI，也可以在其他情况下，用作通用I/O或者通用标志。在BF531、BF532和BF533单核处理器和BF561双核处理器上，硬件将自动实现管脚多路复用。诸如BF534、BF536和BF537等更新型号的处理器具备更多外设，并利用一个硬件子系统来控制管脚的多路复用，因此，我们也提供了一个端口控制服务程序，以便控制这个硬件机制。这个服务程序的大部分运行情况对应用而言都是透明的。应用通常只需要初始化这个服务程序，然后，其他服务程序将自动调用端口控制，对管脚进行适当的控制。与设备驱动程序一样；例如，配置为外部帧同步信号的PPI驱动程序将自动调用端口控制服务程序，启用这些管脚，以用于帧同步服务程序。定时器也一样。如果将一个PWM定时器配置为生成一个输出信号，定时器服务程序将自动调用端口控制服务程序，将管脚配置为输出管脚。标志也是这样，如果打开一个标志，并设置好标志方向，然后，标志控制服务程序将自动调用端口控制服务程序，将管脚配置为通用标志。对应用而言，端口控制服务程序基本上是透明的；只要实现了初始化，应用通常不再需要返回端口控制。

第9b节：API

和其他服务程序的API一样，端口控制服务程序API也包括初始化函数和终止函数。然后是启用各种外设的各种管脚的函数，包括PPI、SPI、SPORT、UART、CAN、定时器和GPIO等管脚。我们还提供了一个机制，可以一下子就检测出管脚的多路复用情况。用户可以利用配置表，换句话说，可以查看所有管脚的状态，并将其状态保存到配置中，之后，还可以设置配置表。不必通过单独的外设单独配置这些管脚，可以利用配置表，为Blackfin处理器上的所有的管脚指定一套全面的多路复用配置。

第10章：结束语

第10a节：服务程序文档

从哪儿可以获得关于系统服务程序的更多信息？嗯，API以及所有这些服务程序的包含文件都位于“Blackfin/Include/Services”目录下。在典型的默认VisualDSP安装中，该目录应当为“c:\Program Files\AnalogDevices\VisualDSP 4.0\Blackfin\Include\Services”。所有面向系统服务程序的“.h”文件都在这个目录下。API包含在“.h”文件中，各个服务程序都有自己的“.h”文件。例如，“adi_dma.h”是DMA服务程序的API；“adi_flag.h”是标志控制服务程序的API，依此类推。

我先前提到过的源文件位于“lib”目录下，或者服务程序源文件目录下（c:\Program Files\AnalogDevices\VisualDSP 4.0\Blackfin\lib\src\services）。在这个

目录下，还有adi_flag.c、adi_dma.c等所有服务程序的源文件。系统服务程序库与其他运行时间库一样，都位于同一个目录下：c:\Program Files\AnalogDevices\VisualDSP 4.0\Blackfin\lib。我们提供了多个范例，这些范例均可在EZ-KIT评估版上运行。在Blackfin EZ-KIT目录，有BF533、BF537和BF561等子目录，其中都包含一个名为服务程序的子目录（c:\Program Files\Analog Devices\VisualDSP 4.0\EZ-KITs），用户可以在这里找到关于使用系统服务程序的范例。

我们还提供了非常全面的用户手册。用户可以登录模拟器件公司网站（www.analog.com）的技术资料库页面，下载《设备驱动程序和系统服务程序手册》（Device Driver and System Services Manual）。2005年9月，我们发布了一些新的服务程序、更新了用户手册并发布了用户手册附录，可以登录如下网址：ftp://ftp.analog.com/pub/tools/patches/Blackfin/VDSP++4.0，下载这些技术文档。

第10b节：总结

希望通过今天的课程，大家能够认识到系统服务程序的优越性。重复性工作量显著减少，利用我们提供的系统服务程序，开发人员不必再亲自动手编写这些代码。这是一个非常稳定的软件基础，业已在各种不同型号的Blackfin处理器上完成测试。要求开发人员自行编程的代码更少了，例如，我们编写了DMA控制器代码，这样，开发人员就不必为不同的应用重新编写这段代码。模块化软件，模块化程度越高，集成越轻松。服务程序将负责管理Blackfin处理器的硬件资源，所以，客户可以直接将利用系统服务程序的不同软件模块整合到一个统一的应用中。此外，系统服务程序还实现了杰出的可移植性，各个处理器都采用完全相同的API，因此，客户可以轻松地将在现有处理器上运行的应用移植到新一代Blackfin处理器中；或者如果应用需要更强大的处理能力，则可以将其从单核处理器移植到双核处理器中。所有API完全相同，开发人员只需要将应用导入新的处理器，并完成相应的重新编译。

第10c节：更多信息

如需了解更多信息，请参阅技术文档。可以在模拟器件公司网站（www.analog.com）的技术资料库页面下载我们提供的用户手册。强烈建议用户阅读每个服务程序用户手册的简介及前几个章节。如有任何其他疑问，请单击下面的“ask a question（我有疑问）”按钮。谢谢参加系统服务程序课程。

再次感谢！