讲座题目：适用于 ADI Blackfin®处理器的 NI LabVIEW™ 嵌入式模块简介

演讲人：Glen Anderson

第 1 章：产品概述

第 1a 分章：导言

大家好，我是 Glen Anderson，模拟器件公司的处理器核开发工具小组的高级工程师。今天我们将介绍适用于模拟器件公司 Blackfin 处理器的 LabVIEW Embedded。我们将做一系列的演示。向大家展示这一产品，我们将一起学习所有不同的特性，并使你充分了解产品的功能，了解产品是针对什么样的用户而设计的，最后我们将对 LabVIEW Embedded 进行回顾。

首先，我们将从概述开始，我将解释什么是适用于 Blackfin 的 LabVIEW Embedded（LabVIEW 嵌入式模块）？LabView 是从什么基础产品上发展而来的？今天，我们会开发一些应用程序，还会进行一系列的演示。我们将通过建立一个应用程序，来告诉你如何下载，在 Blackfin 上如何运行这些虚拟仪器。我们将着眼于用内置的驱动 VI 的驱动器和外设通信。我们还将着眼于调试的功能，在虚拟仪器上设置断点的能力，还能够设置探针，并通过 VisualDSP 在传统的 C 开发环境进行交互调试。

第 1b 分章： 什么是 LabVIEW?

在深入了解 LabVIEW Embedded 之前，我们将谈的第一件事是：什么是 LabVIEW？
LabVIEW 本身是一个产品，是由国家仪器公司开发的产品。NI 是一家历史悠久的公司，测试测量行业里的领先者。你所遇到的任何一种信号，都可以使用它们所生产的硬件器件进行测试和测量。早期传统的测试工程师为了对 GPIB 器件进行自动测试，通常必须为其编写 C 程序。LabVIEW 本身就是一个图形开发环境，它是由 NI 开发的，能够让工程师（不仅仅是 C 程序员，而是真正懂得测试和测量的工程师，不一定需要懂得嵌入式开发）测试任何类型的器件。因此，LabVIEW 是这样一个图形编程的环境，只需要在其中对数据流进行一些拖放操作，就可以在屏幕上对已有的虚拟仪器进行组织，将它们都连接起来，而不必去掌握像如何对一个特定的虚拟器件进行编程这样的细节。LabVIEW 传统上是以桌面环境为目标的。在最近几年来，它们扩展出了更多的移动应用。现在有了 Compact RIO 框架，使你能够将测量放入更多的移动应用中。它们在工业类型设定和嵌入式应用中被广泛使用。作为一款成熟的产品，LabVIEW 带有数千种信号和数学处理函数，以及能与不同硬件相连的连接函数。实际上，它的主要支柱就是这个连接功能，能够适应多种不同的标准，无论是在视觉系统，还是数据采集系统。

现在国家仪器公司认识到这一不断发展的趋势甚至对测试和测量世界造成了影响，测量工程师不必懂得 C 语言，也可以很方便地使用 LabVIEW 环境来对器件进行编程。他们是自己领域内真正的专家，他们知道如何去测试自己特定的产品，他们能够使用 LabVIEW 轻松地完成测试。与他们的情

况相似的还有一些嵌入式系统的开发人员，他们也是自己领域内的专家，他们对于特定的应用十分了解，但却不一定了解实现的细节，如使用 C 语言进行编程，以及中断、寄存器和许多真正的嵌入式系统开发者逐渐需要去面对的问题。因此，LabVIEW Embedded 在 LabVIEW 图形环境的基础上进行了扩展，它能够适用于嵌入式器件的应用。其第一个产品是一个合作的产品，被称作适用于 Blackfin 的 LabVIEW Embedded，它是模拟器件公司和国家仪器公司联合推出的产品，使你能够将这一存在于测试测量行业里很多年的图形开发环境，应用到 Blackfin 处理器上。因此，我们实际上正在进入一个全新的领域，嵌入式系统开发的王国。适用于 Blackfin 的嵌入式模块是将设计能力和人们所熟知 LabVIEW 的图形编程能力结合在一起的，它适用于模拟器件公司的 Blackfin 处理器，Blackfin 是一款高性能、低功耗的处理器。我们的想法是，要把从设计开始阶段到最终产品代码完成这期间所有的工作都在 LabVIEW 环境下完成。我们在那里已经有了这些虚拟仪器，这可能会对今天将要做的演示有一些影响。但既然我已经有了这些事先制作好的模块，那么就不必再去做重复的工作了，我在产品拆封后立即就获得全部这些功能，因此对于像如何对 Blackfin 进行编程这样的问题，不必去了解太多的细节。

第 1c 分章：谁能从中获益?

我已经提到过领域专家这个词，我将谈谈什么人能够从 LabVIEW Embedded 中获益。领域专家是那些真正十分了解自己想解决的特定问题的人。他们知道那是一个算法，却不一定也不必懂得如何为一个嵌入式器件编程。LabVIEW 确实是一个很棒的产品，它使得领域专家能够在最高的抽象级别上去做他们最擅长的工作，即他们的算法或正在尝试解决的特定问题。嵌入式系统的开发者，他们和我一样是传统的 C 程序开发者，通过 LabVIEW，不必再去做重复的工作，从 LabVIEW 中获益。由于有了这些可重复使用的元件、驱动和算法，会使开发周期变得更短，并且 LabVIEW 能够立即向我提供很多这样的功能。测试工程师和质量工程师，我们刚才谈到过 LabVIEW 是如何从测试和测量世界中发展而来的，因此，有些公司已经在 LabVIEW 技术上进行了大量的投资。LabVIEW Embedded 能够让你从以前的投资中继续获利，并且使你能够将其扩展到设计周期中去。在此产品之前，这是不可能实现的。因此，质量工程师，你们知道，自始至终只使用同一系列的一组工具。如果我在开发阶段中使用 LabVIEW，那么我在测试和测量阶段也可以使用它，甚至在产品定型阶段，设计周期开始时就能使用 LabVIEW。在整个开发链上自始至终只使用同一系列的一组工具确实会提高质量，因为我不必再花时间将不同的工具集成在一起。即使集成在了一起，也不一定意味着它们能很好地协同工作。

第 1d 分章：结构概述

从较高的层次上来看，LabVIEW 嵌入式模块非常简单。你可以看到，这里是我的用户程序，那是我将要画的框图，我们将做一些示范。LabVIEW 会使用我画的框图，并在内部产生标准的 ANSI C 代码，这与传统的嵌入式开发者所编写的代码相似。在内部，此 C 代码提供给 VisualDSP++的工具链，VisualDSP++是 Blackfin 最优化的编译器、汇编器、连接器、装载器和分割工具，但它们隐藏在内部（你不必去关心它们具体的功能）。实际上我不必进入 VisualDSP 环境，而只需待在使用舒适的 LabVIEW 环境中即可。在这里你可以看见所产生的代码被送入 VisualDSP，与这些元件的驱动进行连接，这些驱动是由 VisualDSP 的系统服务库建立的。代码还被连接到一些适合 Blackfin 的 VI 上，VI 是虚拟仪器的缩写。因此，所有这些在一起构成了我前面所说的可再使用的模块。随后，使用 VisualDSP++内核或 VDK 将它们连接在一起，VDK 是内部的操作系统，将所有的功能结合在一起，并调度所有任务，所有这些一起构成了一个应用程序，它可以被下载到我的 Blackfin 处理器中。

第 1e 分章：演示设置

今天，我将使用 Blackfin EZ-KIT Lite，它是模拟器件公司的标准开发平台。我面前是一个工作在 600MHz 的 ADSP-BF 537 处理器，在这里可以看到这块电路板的一些技术规格，包括 64MB SDRAM、Flash、立体声音频，模数、数模转换器，还有一个高性能的 USB 调试接口，用来调试 Blackfin，通过这个接口能够下载代码到 Blackfin 上并运行。电路板上还集成了以太网和 CAN 接口，以及教育实验室虚拟仪器套件，Elvis。这是一个 NI 的术语，它是一个类似于 PCI 的接口，位于卡的侧面，与产品所提供的适配器一起使用，就能将此电路板与 NI 所有型号的数据采集板以及不同类型的硬件相连。在电路板的底部还有为 EZ-Extender 卡配备的连接器。根据应用的需要，这些不同的卡，可作为音频和视频、以太网、USB 以及其它各种类型的外设来使用。这款电路板具有极强的扩展性，我们可以使用这一低成本的开发系统来开发出最终的产品。

我需要为演示做如下的准备，一台装有 LabVIEW 和 VisualDSP 的计算机，LabVIEW 和 VisualDSP 两者合在一起就是适用于 Blackfin 的 LabVIEW Embedded 产品。在这里，我将用 HPUSB J TAG 仿真器，它是一台高性能的 USB 2.0 仿真器，它能够将程序下载并与 EZ_KIT 电路板相连。这是一个 USB 2.0 件，我在这里使用它主要是因为它的速度较快。我前面提到过，这款开发板已经自带了一个高速 USB 调试接口，但使用的是 USB 1.1 连接。因此拆封后，你就能得到全部的连接功能，只是 USB 1.1 的速度会比 USB2.0 稍慢一些。因此就这样将 PC 机与开发板相连接，由于要做一些音频的演示，我将一条音频输入线与开发板上的多媒体数字信号编解码器相连。

接着，我将开发板的输出端与扬声器连接。这就是全部的配置。我想最有帮助的还是做一些实际的演示来向各位展示 LabVIEW Embedded 究竟是如何工作的。

第 2 章：开发简单应用程序

第 2a 分章：LVE 环境

这里所显示的是 LabVIEW 7.1 嵌入式版本的窗口，这是第一次启动 LabVIEW 所看见的第一个窗口。如果你之前曾使用过 LabVIEW，它看起来会与你通常所见到的桌面版本的窗口相似，只是在这里的程序启动画面上多出了“嵌入式版本”这几个字。但一个主要的区别是下面的执行目标发生了改变。默认的是适用于 Windows 的 LabVIEW，在此模式下，所开发的是传统的 LabVIEW 桌面代码，与国家仪器公司一直提供的产品是类似的。LabVIEW Embedded 使我能够面向其它的处理器，在本实例中，我们可以选用 ADSP-BF533 和 ADSP-BF537，而我已经与 537 处理器相连，因此就将选用这一型号。首先，我将从一个空白的虚拟仪器（VI）入手，我可以在这里选中它，并对其进行命名。每个虚拟仪器都由两部分组成，一个是前面板，就是我左边这个灰色的窗口，还有一个是框图，就是右边这个白色的窗口。在 C 代码术语中，可以将前面板想象成一个头文件，能在那里定义虚拟仪器的输入和输出。框图实际上就相当于源文件，在那里完成大部分工作，比如开发一个算法或器件驱动，或最终程序所需的一些功能。我们来看一看框图，我们将稍后再讨论前面板，现在让我们先来看看真正用来进行代码开发的地方。现在 LabVIEW 是一种图形化的数据流拖放语言。这就是说我能够使用工具盘，而不用再将代码输入文本编辑器中，在这里你们能看到我的功能工具盘。与其它任何一种语言一样，你对它的结构可能已经非常熟悉，例如，这是一个 for 循环，这是一个 while 循环，case 结构类似于 if 声明，顺序结构类似于 C 中的大括号，局部变量、全局变量、注释、内联 C 代码，稍后我们将对内联 C 代码做一些讨论。接着是数字，我们在这里进行加法、减法和乘法的运算，我们还有许多像分析库这样更复杂的模块，这里是一些信号处理函数，例如信号产生模块。如果想产生一个正弦波，就可以将这个模块从这里拖放到那里。

第 2b 分章：点亮 LED

首先我将选择 Blackfin 工具盘，在这里你能看到，我可以选择查看所有可用的 Blackfin 虚拟仪器。现在当我点击鼠标右键，其中的大多数看起来都是一样的。你能看到，我的结构、数字以及信号处理模块还在这里。现在又多了一个 Blackfin 工具盘，这才是由模拟器件公司和国家仪器公司联合开发的产品所带来的真正的不同之处。现在，这些模块组使我们能够在产品拆封后，立即使用许多为 Blackfin 准备的非常复杂和有效的功能。下面我将对这些功能进行介绍。在这个实例中，我在这里连接上一块 EZ-KIT Lite 开发板，你能看到我有一个 EZ-KIT 工具盘，我能够选择 LED，有一个数

字，这块板上有 6 个 LED，你能看见这里有一些模块，这个模块能让我查询 LED 是否亮着，我能够控制 LED 的开关，使它们周期性地闪烁。我可以在这里控制 LED 的开关，你看我能选择那一项。我只需要对其进行拖放操作。由于在这里我使用的是 537 处理器，因此我选择这个模块的 537 版本，你能看到这个模块里有一些输入和输出。输入在模块的左边，输出在模块的右边。我有一个 LED 的号码，在这里我有 6 个 LED，因此我确信在这里用 0~5 来表示这些 LED。我能在这里点击鼠标右键，我想让 LED 闪烁，控制 1 号 LED 的开关，因此我可以创建一个常数。现在当它被执行时，1 号 LED 应该闪烁，也就是说当它亮了之后，会接着灭掉。其余的输入和输出是错误输入和错误输出，这是一个很常用的 LabVIEW 的范例，能将错误信息送入或输出，你可以将它看作一个函数的返回值。由于我们只是控制 LED 的开关，因此在这里，我不太关心这些错误，这样的话，我现在先不连它们。下面我想使这个 LED 依次闪烁，我可以使用一个 while 循环，通过将这个控制 LED 开关的模块放置到我的 while 循环中，我就能使 LED 依次闪烁。有一个退出条件，在这里被称作循环条件，它位于右下角。我能够将循环条件连入环路中，来确定这个 while 循环将持续多长时间。我想让这个循环一直进行，因此我就连入一个恒定假值，这就等效于一个无限循环。如果我现在就开始运行程序，则由于速度太快，我们将无法看到 LED 的闪烁。因此，我要在这里加一些延时，你看我可以向下进入时间对话框和错误工具盘，这里有一个延时 VI，我将设置延时，我将再创建一个常量，我打算让其延时 500ms。至此，我的整个应用还十分简单，我所做的仅仅是让 1 号 LED 在一个连续的循环中闪烁，在每两次闪烁之间，大约有半分钟的延时。虽然这个程序很简单，但对我们来说是一个很不错的开始。

接着，我将要在这里运行程序。首先，我需要保存刚才所做的工作。我给它起名为 LED 实例，然后点击 OK。这样我就已经保存了 VI，软件告诉我需要一个嵌入式工程来运行这个程序，但我还没有创建这个工程，因此我需要创建一个。因此，我选择 OK 来创建一个新的工程，出现的默认的名字是“LED 实例”，这个名字不错，因此我就使用这个默认的名字，现在出现了一个嵌入式项目的管理窗口。这个窗口与其它你所用过的 IDE 窗口类似。在这里的建立选项中，我可以添加所有的虚拟仪器，并管理所有的代码。看一下这里，我能创建新的工程，打开已有的工程，或关闭已经打开的工程，保存，保存所有，这些都是标准的选项。Target 菜单中有许多有趣的选项。你看我可以建立应用程序，或重新建立所有的应用程序，这会删除所有的东西，在一定程度上又需要从头开始。我可以将完成的应用程序下载到 Blackfin 中，对 Blackfin 进行复位，并运行和调试程序，我们将简单地演示这些过程。我可以设置不同的建立选项，这样能弹出建立选项对话框。我可以做不同的设置来告诉 LabVIEW 如何产生代码。我能够告诉 LabVIEW 是否产生保护性代码，来防止发生除数为 0 或索引数组时越界的情况。我可以选择不同的调试模式，这里有更高级的函数。如果想让编译器执行专门的 VisualDSP 优化方案，可以将它们添加到这里。如果有自己定制的 Blackfin 硬件，可以

进入这里，来准确地告诉 LabVIEW 所使用的是哪一款硬件和哪一款 Blackfin 处理器。可以提供的信息有，时钟频率、电压、核心电压、外部电压、封装形式以及芯片版本。现在 LabVIEW 自动监测到我在这里所使用的是 0.2 版本的芯片。如果要使用一块较老的开发板，则需要告诉 LabVIEW，使得编译器能够处理版本 0.1 中所出现的异常情况。我不打算将这些信息保存，我将只使用默认的设置。下面我们来建立程序，进入目标菜单，选择建立所有。LabVIEW 会获得框图，产生 C 代码，现在你能看到它正在将代码发送给 VisualDSP 的代码产生工具、编译器、连接器和汇编器，在这里都隐藏在内部。在整个过程中，在任何地方都看不到 VisualDSP，LabVIEW 为我们处理了一切事务。我已经成功地建立了程序，接下来我们要将应用程序下载到 Blackfin 中，让我们现在进行这一步操作。你能看到 LabVIEW 正在通过这个高性能的 USB 仿真器与 Blackfin 进行通信，它已经连接上了，现在正在将代码下载到 Blackfin 中。现在下载已经完成。我们完成了建立和下载操作，接着我们将试着运行程序，来看看会发生什么。我点击运行，可以看见 LED 1 大约每 500ms 闪烁一次，并且其余所有的 LED 都是熄灭的，非常好。让我们停下来想一想刚才都发生了些什么，让 LED 闪烁不是什么尖端的技术，但想一想，如果要在传统的 C 开发环境下同样实现这一应用，必须做多少工作。这里会需要一个 500ms 的定时器，也就是说我必须在 Blackfin 上使用定时中断，查阅硬件参考指南，来确定需要写哪些寄存器，我还需要写一个中断服务程序，并弄清楚如何维持 LED 的状态。此外，在设置 LED 时还要先查明它是处于输入还是输出，因为在 Blackfin 上输入和输出使用不同的设置寄存器。因此，在这种情况下，就要求我们对 Blackfin 有非常深入的了解。而使用前面我说的那种方法，就不必对 Blackfin 有什么了解，我所要做的就是知道要让 LED 闪烁，并需要一个延时模块。

第 2c 分章：并行循环

下面要演示一些更强大的功能，我们已经能够很好地控制 LED 的闪烁，接下来我想控制更多 LED 的闪烁，我可以简单地拷贝这个模块，我将在这里做 4 个拷贝，4 是一个好数字。我完成了这个模块的 4 个拷贝，我想让 LED 2 每隔 750ms 闪烁一次，让 LED 3 每隔 1s 闪烁一次，让 LED 4 每隔 1.25s 闪烁一次。我将再一次回到工程管理选项来保存我所做的工作，我们将建立程序，然后提示我保存，产生了更新的 C 代码，并将其送入 VisualDSP 编译器，现在正在连接我的最终应用程序。都完成了，我再次将程序下载到 Blackfin 上。现在，当我运行程序时，我看见 4 个 LED，LED 1、2、3、4 都开始闪烁了，大致看上去是正确的。现在，如果我们停下来，花几分钟时间考虑一下，在 C 语言环境中，仅仅是让一个 LED 闪烁，所需要做的工作就已经十分繁琐了。现在，要同时实现 4 个循环，而且定时器和 LED 的值还不相同，但看看现在我所需要做的只是拷贝一些模块并建立程序。如果要在 C 语言环境中实现，实际上会特别复杂，这需要我在某种程度上掌握隐藏的

操作系统，用 VDK 的术语来说，就是要能够管理这些并行的任务和调度。但在这里，我就不需要懂得调度，我所需要知道的只是要让 4 个不同的任务同时运行，因此我只需在同一图中放置 4 个不同的循环。LabVIEW Embedded 帮助我将问题变得抽象化，这样就不需要真正了解所有 VDK 和 Blackfin 硬件上那些低层次的细节。以上是一个实例。

第 3 章：外设通信

第 3a 分章：音频实例

使 LED 闪烁还不能算是一个真实世界的例子，我曾在前面提到过，LabVIEW Embedded 在拆封后能提供许多功能强大的 VI。现在，我们就来看看其中的一个。这是一个音频例子，在这里我所要做的设置是将音频信号从我的台式 PC 传到开发板上，随后通过扬声器播放出来。让我们来看看如何实现这一应用，在这个工程中也只有一个 VI，它比前面让 LED 闪烁的例子更复杂，但还远没有在 C 中实现这一应用来得复杂。现在看这里，LabVIEW VI 通常是从左到右的，因此我们从这里开始，你可看到我在这里分配了一些缓存器。一看到这个图标我就知道它是缓存器，但你们却不一定知道。LabVIEW 中一个十分有用的特性是上下文相关的帮助。我可以打开这个上下文帮助窗口，现在，只需将鼠标移到不同的模块上，就能得到每个模块的帮助信息。因此这是一个查看和理解代码的好途径。你在这里可以看到，这是一个初始的数组模块，数组长度为 512，这是一个常量，因此我创建了一个 512 字节的数组。接下来在这里我将 512 除以 2，得到 256 字节，我在那里创建两个数组。让我们将这个窗口关闭。我在这里对 Blackfin 音频进行初始化，你可以看到其中的一个输入就是在音频播放时我所要收集的采样中的一个。这是一个接收缓存器和一个发送缓存器，这里是数据输入和输出并处理数据的地方，这些是它的连线。

我们从这里进入这个 while 循环，这个 while 循环与前面 LED 实例中的 while 循环是相同的，但明显内部多了不少内容。当我进入 while 循环时，我已经在这里对我的音频进行过初始化了，我将等待这个标志，数据准备就绪的标志。现在，隐藏在内部的 Blackfin 一直在收集数据，并将其存入存储器中。现在我拖入这个初始的音频模块，输入的采样值个数为 512。也就是说，输入 512 个采样值后，程序就返回。在数据缓存被装满之前，将会一直等待，只要数据缓存已满，就会继续执行。因此一旦我从这里出来，能得到数据，并且将返回这个数组，我将数据分开， 使它们符合标准的音频格式，I^2S 模式。我将一个大缓存分成独立的左、右声道，你看这里我既有右声道输出，又有左声道输出，现在我还没有对数据进行任何处理，只是将它直接送入这一组合模块中。你可以看到 BF 组合声道，我给出左、右声道输入，随后将它作为一个缓存器的输出发送出去。然后，我将它发送给芯片，这个模块显示工作已完成，将会返回并开始等待下一轮 256 个采样值的输入。现在，

我返回到工程管理菜单，让我们来建立程序，并保存，软件让我进行保存，现在我已经完成了建立工作，接着要进行下载，好的，下载也完成了。现在我将运行程序，如果我播放音频，我们就应该能够听到，我听到了贝多芬第 9 交响曲。现在我要停止播放音乐了。

第 3b 分章：前面板控制

当我得到数据后，并没有对它进行任何处理，因此让我们来做一些更有趣的操作。数据进入到一个大缓存器中，我将它分成左、右两个声道。我想要去掉这两条线，现在我将转到前面板，我们还没有讨论过前面板，前面我提到过可以使用前面板来定义虚拟器件的输入和输出。它的功能还不止这些，你可以看到前面板上的工具盘与框图中所看到的工具盘是不一样的，在它上面有许多不同的控制，看这些按钮。在这里它们几乎是图形化的用户接口元件，字符串控制，这里有图表和各种类型的数据可视化选项。我将选择一个数字控制功能，并取名为增益。你可以将它看作一个变量，现在我的增益是 0。现在，当我将前面板上的增益模块放到框图中时，在我的代码里就会自动出现一块代表这个模块的代码，并且它在这里的名字也是增益。现在，像传统的编程环境那样，LabVIEW 具有了类型的概念。我能够看到是因为它变成了橙色，表示这是一个双精度类型，即一个 64 位的浮点数。我可以选择来改变这一类型，比如说我想将它变成一个 32 位的整数，或一个长整型，我先选择，然后你看到它变成了蓝色，表示这是一个 I32 类型的整数，这正是我所要的。我要将进入的采样数据乘上一个固定的增益因子，因此点击鼠标右键，进入数字工具盘，在这里我们能找到加、减、乘、除运算工具，选择乘法器，将它连接到那里，将右声道连接到乘法器上，并将输出放在右边。现在要对左声道作同样的操作，按住控制键并将乘法器拖出，我要连接左声道，但这有些费事，让我们来看一下，我把输出放到了左边。因此我让右声道乘上增益，再将它送回，我又让左声道乘上相同的增益因子，再将它送回左边，如果我想交换左、右声道的位置，只需简单地将两个乘法器的输入交换即可。可视化的编程环境确实使得这类操作看起来比在其它环境下更为直观。我将上面的操作保存，并返回到工程窗口。

第 3c 分章：运行时间调整

我们已经在建立选项中完成了一些操作。现在建立选项看上去和前面有所不同，其中有调试设置一栏，而且现在已经出现了几种不同的调试模式可供选择。 而在 LED 的实例中却没有调试模式可供选择，因为我们并没有真正进行调试，而只是假设程序已经可以工作，然后就直接运行了。可以通过串行端口来调试，EZ-KIT Lite 上有一个 RS232 端口，可以将其连接到 PC。如果我使用 JTAG 仿真器或硬件内置的 USB 接口连接，我可以选择这个非插入式模式，我已经选择了这个模式。在那里选择 OK。现在，我将保存，并重新建立，现在建立完成，可以进行下载。前面，我总是点击

这个运行按钮或在目标菜单里选择运行选项，现在我先不选择运行，而是选择调试，这样就需要进行不同的操作。现在，你会注意到前面板弹出到前面了，并且也不再是通常所看到的那种灰色，变成了一种固态的灰色。那里有一个终止按钮，还有一个暂停按钮，因此我能有效地对应用程序进行调试，并且交互地为增益设置不同的值。让我返回这里，来确定音乐正在播放。现在选择的增益值为 1，能够听到音乐被重播。让我们返回，选择增益值为 2，声音变响了，来回调节几次。这里，改变增益的值起到了音量调节的作用。增益只是一个方面，滤波器参数也可以像增益一样被调节，假设我正在 LabVIEW 中开发一个滤波器算法，我将所有不同的滤波器的参数都列在这里，实际上当程序在硬件上运行时，我能够对这些参数进行调节，并实时地观察不同值的效果。LabVIEW 中的这一强大功能在传统的开发环境中是难以实现的，因为它们无法提供这样的交互性。我将停止调试，并且返回。

第 3d 分章：器件驱动

现在，在离开这个音频实例之前，让我们返回，来更仔细地看一看这里究竟发生了什么。让我们来回顾一下，如果我查看 Blackfin 下的工具盘，初始的音频模块、等待标志、获取缓冲器以及许多器件都是可用的，你能看到器件，这里有一个通用的工具盘，使我能够打开和关闭一个器件，向一个器件发送数据，或从一个器件得到数据。适用于 Blackfin 的 LabVIEW Embedded 的强大特性之一就是所有这些器件模块。例如，我在使用 1854 和 1871，这两个模块在这里是 537 EZ-KIT 上的 ADC 和 DAC，我使用它们输入和输出音频信号。我可以简单地打开它们，读取一些数据，写入一些数据，并设置不同的参数。假设其中一块芯片上有一个音量寄存器，我就可以用这一控制方法来设置它的值。因此我不必亲自编写这些驱动，模拟器件公司在产品拆封时已经把它们提供给了我，我只有一件事需要担心，那就是我前面谈到过的这些可重复使用的元件。现在，假设有一个器件没有在这个工具盘中表示出来，如一些其它的编解码器，则会有一些工具组，能让你访问 Blackfin 上所有的外设。在本实例中包括 PPI 、SPI、SPORT 和 UART，因此我就可以假设有自己的编解码器，并且它通过 PPI 连接到硬件上。因此，无论器件在什么地方，我都可以在 LabVIEW 中使用这些低层次的 PPI 模块，来实现器件的驱动，而不必到 C 环境中去做这些事。我可以使用这些基本的构造单元，来开发许多复杂的驱动，如处理 DMA 和中断以及类似的操作。在有些应用中，这些即拆即用的功能确实非常好用。还有一些模块可以用来控制 Blackfin 的功耗，我提到过 Blackfin 是一款低功耗、高性能的产品，这里我实际上指的是能够改变频率，我可以让 Blackfin 工作在一个特定的频率下，可以通过施加某一个电压来让 Blackfin 以最快的速度工作，假设它使用在一个无线或手持产品中，这一功能能够最大限度地利用电池的能量。

还有一些分析库。至此，我们已经看到，LabVIEW，无论是适用于 Windows 的版本，还是适用于 Blackfin 的版本，都有内置的信号处理模块。这是我的信号产生模块，这是一些时域函数和不同的滤波器（巴特沃思型和切比雪夫型），这些模块都已经被 LabVIEW 的用户使用过很长时间，并且已经有很多利用这些模块优势的现成的代码。有了这些以后，将现有的应用程序从 LabVIEW Windows 上移植到 LabVIEW Blackfin 上将十分容易。但是，这些程序可能并没有专门为 Blackfin 进行过调整和优化，因此模拟器件公司开发了这些 Blackfin 的分析库，并且在不断地增加库的内容，你可以看到这是一些信号处理模块，这是一些滤波器，这是一个复杂的 FIR，这是一个常规的 FIR，那里还有 IIR 和一些不同类型的滤波器。模拟器件公司的开发工具小组对这些滤波器模块进行过手动的调节，使它们能真正地发挥出 Blackfin 结构的优势，真正得到尽可能高的性能。因此，我可以使用现有的 LabVIEW 模块来搭建系统原型，能够先让系统迅速地运行起来，为了得到最高的性能，可以选用 Blackfin 的专用库，来保证系统在 Blackfin 上能够真正出色地工作。

第 4 章：调试功能

第 4a 分章：断点和探针

现在，让我们来看一看调试。我将保存并关闭我的工程。任何一种开发环境都无法避免的一个问题之一就是无论你编写什么样的代码，都需要去调试。这也是生命的实情之一。因此 LabVIEW 内建了一些功能，使你能够深入地查看你的应用程序正在做什么，无论它出现了错误，还是虽然是在做正确的事，但却不是按照你所期待的方式进行。LabVIEW 内建了许多工具来让你进行查看。这里是一个调试的例子，我们将打开前面板，可以看到在前面板上有一些元件。我们看看下面的代码，这张图不是十分复杂，但却实现了不少功能。从左向右看，我在这里初始化了一个 5 维数组，它是一个有 5 个元素的数组，这个数组是橙色的，因此我知道它是双精度类型的。我将这个由 5 个双精度的值所构成的数组传递到我的 while 循环中，我要让这个数组进行循环，我可以按 Ctrl+H 来查看上下文的帮助文档。我将这个数组移动一位，将其中的一个元素替换成一个随机数，这个随机数的范围从 0 到 1，再将它乘以一个前面板上的因子 X，然后用它来取代数组的子集。因此，实际上我仅仅产生了一个随机数，我对数组进行移位，将新的数插入到数组的第一个位置上，随后在这下面计算数组的平均数，再将它输出到这里的平均数模块。你可以看到，这是 X，这是平均数，因此在这个特定的实例中，实际上有一个输入和一个输出。假设我确实十分想查看这里发生了什么。我可以选择设置一个断点，假设我实际上想逐步地运行我的程序。现在，LabVIEW 使我不需要进入 C 环境，因为所有的 C 代码都是隐藏在内部的，如果我不想那样做，就不必陷入低层次的细节中。我可以到建立选项中选择非插入式模式，这样就能够进行调试了。我要下载，软件提示我要先建立程序，那我就建立一下，好的，现在可以下载了，我可以从目标菜单中，或者从这里上方的工具条中

选择调试，这次我从工具条中来选择。现在程序已经开始运行，LabVIEW 正在更新这个初始的数组模块，由于我设置了断点，因此程序在那里停了下来，我现在有许多选项，在一个传统的开发环境中都可以找到它们。我可以逐步地运行我的代码，可以暂时离开整个 VI，可以选择继续运行，或中断调试。我将选择逐步执行这个初始的数组，你可以看到它将我带入了 while 循环，我想进入这个 while 循环，来查看在这里发生了些什么，你可以看见我在逐步前进，现在我处于循环 1-D 数组的模块，再前进一步，现在我在乘法模块中。你看我可以在这里设置探针。我实际上想查看乘数的值，因此我选择设置了一些探针。你能看到第 5 号探针显示的是 X 的值，这个值是 5，与前面板上所给出的一致。这个由我的随机数模块所产生的随机数看起来像一个处于 0 到 1 之间正确的数，因此使用探针就能查看具体的数值。现在，如果我再前进一步，可以看到 5 乘以 0.83，4.15 看上去是正确的。因此，这使我能够真正进入到程序内部，去分析在代码中究竟发生了些什么。我可以选择在这里，平均值模块，再设置另一个断点，随后我可以选择继续。现在每次我点击继续，程序都将停在这个平均值模块，这是因为每次循环时，都会在那里遇到断点，每当我点击继续时，同样的情况都会再次发生。因此，断点确实能让我到达那个位置，并查看在我的程序内部发生了什么。

第 4b 分章：同步调试

关于 LabVIEW Embedded 的一个十分重要的问题就是，我觉得很多公司已经在传统的技术上作了很大的投资。实际上，LabVIEW 在很大程度上允许你重新使用许多传统的技术。让我停止调试，让我们来快速浏览一下那些能使我们真正进入更低层的特性。这里你能看见我在浏览框图，我能查看这些线，我能进入不同的 VI 内部，进行浏览，再出来。假设在内部发生了一些问题，使我必须真正进入 C 层次来检查。LabVIEW 允许你真正进入到 C 层次，在那里进行无缝地调试。我将其保存。进入目标设置菜单，现在出现了一个我不曾见过的对话框，我能在对话框里告诉 LabVIEW 我将如何与 Blackfin 进行通信。在本实例中，我选择 537 EZ-KIT Lite via HP-USB Ice 一项，这就是我的连接方式，这里有调试选项。它们使我能够真正控制不同的调试特性。这个前面板更新周期选项能让我设置 LabVIEW 多久访问一次那个对象，并查询一个特定的值——假设有一个探针窗口。我能够让 LabVIEWS 允许我探测数组和簇，簇类似于 C 中的结构体。我可以使用 VisualDSP++ IDDE 来进行调试，我在前面几次提到了 VisualDSP，它是更基本的技术，使用 C 代码来产生优化的 Blackfin 应用程序。VisualDSP IDDE 或集成开发和调试环境是 VisualDSP 中的应用程序，它使得传统的 C 开发者能够逐句通过它们的代码。因此，假设我想进入 C 的层次，我可以选择使用 IDDE 进行调试，并选择 OK，接下来，我所要做的就是释放 IDE，是的，这就是 VisualDSP IDDE，现在这里的许多东西与传统的 C 开发环境都很相似。以一个工程组为例，其中有存储器窗口和寄存器窗口，现在我正与 Blackfin 相连，如果我愿意，就可以选择去查看累加器以及不同类型

的寄存器和存储器，我能够运行程序，逐步地来查看。我将在这里使用相同的调试实例，因此不需要再重新建立，我现在所要做的，就是回去确定在这个初始数组模块中仍有断点，接着将进行下载。这里所发生的已经不仅仅是直接和 Blackfin 进行通信，LabVIEW 现在正在将代码装载到 VisualDSP IDDE 中，因此当我选择调试时，你可以看到我在这里被暂停。在 IDE 中，你能看见我的“Debug Example.c”，它与这里的“Debug Example.vi”相对应，你能看到真正的 C 代码。在本实例中，它就在数组初始化模块的前面。因此，现在我已经处于 C 程序中，我可以选择查看存储器窗口，可以打开一些寄存器窗口，可以查看表达式窗口，在那里仔细地对不同的事件进行考察，现在 N 的值是 1，因此，我可以依照自己的需要进入尽可能低的层次。我甚至能够在这里逐步地将 C 代码执行几次，随后返回 LabVIEW，将它最小化，我还能逐步执行，让我们看看是否能同时将这两个窗口都显示出来。你可以看到这里是我的 C 代码，这里是我的 LabVIEW 框图，我可以逐步执行到那里，你可以看到 C 代码实际上与框图是一起逐步执行的。因此这是 LabVIEW 提供的一种方法，使你能够进入一个更低的层次，你需要明白，这样做会花费大量的时间。

第 4c 分章：代码再用

适用于 Blackfin 的 LabVIEW Embedded 所提供的与已存在的 C 代码进行连接的机制之一，就是调用库函数注释，它由这个高级的工具盘提供。假设在一个库中已经有了我自己在 VisualDSP 下开发好的代码，我可以选择将其中的一个放到框图上，对它进行双击，此时就会允许你设置一些选项。假设库文件为“My Library.dlb”，DLB 是 VisualDSP 库文件的扩展名，假设在这里我要保留这个加法的题目，给这个函数取名为“MyAdd”。假设返回类型是一个数字类型的 32 位整数，我可以添加一个参数，你可以看见这里的下面提供了 C 的函数原型，我已经添加了一个名为 arg1 的变量，假设我要在内联的 C 代码中将它换作 X，并且我还要添加另一个名为 Y 的变量。当我点击 OK 时，可以看到模块的输入端实际上发生了改变，这个输入被称作 X，这个输入被称作 Y，这是我的输出，在这里被标记上了结果。因此，这是另一种方法，我在这里编写 C 代码，来与已有的代码连接到一起。

LabVIEW 使我能够保留已做的投资和已有的传统工具的第三种方法就是，在嵌入式工程管理器中可以直接添加文件，我可以在这里添加 VI，还能添加外部文件，你看，我实际能够将.C 和.lib 或.dlb 文件直接添加到我的工程中去。假设我有一个库或一个.C 文件，或甚至有一个汇编文件，并假设我还购买过一个算法，但是用汇编语言写的，并且我有一个 a.asm 的文件。我能够将它直接添加到我的工程中去，随后使用这个调用库函数节点，我也能调用它。因此，事实上 LabVIEW 使得

这些传统工具的使用变得简单，你可能已经有了这些工具，无论它们是算法还是驱动，也无论这些代码有什么功能。因此这是 LabVIEW 另一个十分重要的特性。

第 5 章：总结

第 5a 分章：产品详细信息

我们已经讨论了很多关于 LabVIEW 本身的话题和它所具有的特性，调试、驱动和分析库。下面将简要说明当你购买适用于 Blackfin 的 LabVIEW Embedded 时，实际上将得到些什么。这个产品本身由模拟器件公司和国家仪器公司共同提供。在两家公司都可能够购买这一产品。本产品的技术支持服务由国家仪器公司提供，现在，这种由国家仪器一家公司统一提供技术支持的服务结构实际上是非常好的，如果你对于产品有什么问题或需要帮助，可以与国家仪器公司联系。假设一个问题最后发现交给模拟器件公司处理会更好，国家仪器公司会通过两家公司之间的协调机制来分配解决这些类型的问题，而不会让您奔走于不同的公司之间，去经历不同的技术支持机制，NI 和 ADI 已经建立起许多这样无缝的内部联系机制。因此支持服务完全由国家仪器公司提供。在产品的包装内，就有很多内容，你会得到国家仪器公司的 LabVIEW 产品的全部会员资格和模拟器件公司的 VisualDSP++产品的全部会员资格，你会得到这款适用于 Blackfin 的 LabVIEW Embedded 模块，这款软件是 LabVIEW Embedded 最完善的版本，提供了经过优化的库和调试功能。你会得到我在今天的演示中所使用的 ADSP-BF537 EZ-KIT Lite 开发板。与产品一起附带的还有内建的 USB 调试端口，因此你就不必再购买像高性能的外部 JTAG ice 这样的设备。你会得到所有的电缆，耳机以及 EZ-KIT Lite 开发板的包装内通常会带有的附件。你会得到这个数据获取适配器，它可以插在 NI Elvis 接口上使用，在卡的边上还有一个类似于 PCI 的接口，使你能够与国家仪器公司大量不同类型的数据获取硬件相连。并且你还将通过国家仪器公司获得自动的软件更新服务，当新版本发布时，如果需要，我们也会提供相应的技术支持。

第 5b 分章：总结

总之，我们已经讨论了许多关于 LabVIEW Embedded 的话题，实际上关键点是：更短的上市时间，我们相信，使用这种层次更高、用法更简单的数据流语言，通过将实现与问题本身分离，能使得我们可以缩短产品上市的时间。由于使用了拖放的框图，LabVIEW Embedded 使我们能够更快地适应设计和市场压力的改变。随着适用于 LabVIEW Embedded 的新算法和驱动被不停地开发出来，我们能够更快地将这些先进特性和函数加入到自己的产品中去。如果今后你的需求发生变化，就可以直接使用这些已有的模块。最后，在整个产品寿命周期内使用同一系列的工具十分重要，传统的 LabVIEW 已经在产品的原型阶段、测试和测量阶段都建立了稳固的基础，我们现在可以将其延伸到实际的开发阶段中，从设计的开端到产品发布，自始至终使用同一系列的工具确实是非常有效的，也会使得产品的质量有很大的提高。

欲获取关于 LabVIEW Embedded 更多的信息，请访问国家仪器公司的网站 NI.com，你可以在 NI.com/LabVIEW 上看到更多关于 LabVIEW 的信息。你能在所提供的这个链接上看到更多关于 VisualDSP++的信息，那是我们的 CrossCore 工具的网站，你还能访问 Analog.com。如果你还有任何问题，请点击屏幕下方的“询问问题”按钮。我希望今天的讲座能对大家有所帮助，谢谢大家的参与。

文档结束