Application Note

在 TM4C MCU 上使用 FreeRTOS 开发高级应用

Ralph Jacobi and Charles Tsai

摘要

FreeRTOS 是用于嵌入式系统的实时操作系统。由于其具有紧凑的尺寸，还可根据免费开源许可分发，已被广泛移植到许多架构平台中。本应用报告是*在 TM4C MCU 上使用 FreeRTOS 开发常见应用*应用报告的续篇，提供了适用于 USB 和以太网外设的更多示例，以便演示如何在德州仪器 (TI) TM4C 系列 Arm® Cortex®-M4F 微控制器上使用 FreeRTOS 内核。

本应用报告中讨论的所有示例工程的源代码均可从以下链接下载：http://www.ti.com/cn/lit/zip/spma087。

内容

商标

1 引言

在微控制器市场中，用于管理应用程序的嵌入式实时操作系统 (RTOS) 的利用率持续稳步增加。由于人们对 RTOS 解决方案的关注度和需求不断增加，出现了许多开源 RTOS 产品/服务，其中包括广泛使用的 FreeRTOS。为了满足人们对 TM4C Arm Cortex-M4F 微控制器系列的 FreeRTOS 支持日益增长的需求，我们发布了一系列应用报告。*在 TM4C MCU 上使用 FreeRTOS 开发基本应用* 是最初发布的应用报告，提供了一组基本应用程序示例。之后发布了*在 TM4C MCU 上使用 FreeRTOS 开发常见应用*，扩展了应用范围，重点介绍了许多常用 TM4C 外设。这是本系列中的最后一份应用报告，将介绍更先进的 USB 和以太网外设。

本应用报告包含适用于 TM4C123x 和 TM4C129x 器件系列的示例工程。这些工程基于 TivaWare 软件开发套件 (SDK) 中提供的裸机示例。TivaWare™ SDK 包含适用于所有外设的驱动程序库 (DriverLib) API，这些外设是 TM4C 微控制器上任何应用程序的构建块。提供的示例工程演示了如何使用 FreeRTOS 内核中的 DriverLib API 来创建实际具有各种器件外设的简单应用程序。本报告中演示的外设如下。

- 通用串行总线 (USB)
- 以太网

2 安装方法

以下步骤演示了如何下载 TivaWare 和 FreeRTOS 的最新版本。如果之前已完成下载，可以跳过此部分。

1. 从此处下载最新的 TivaWare SDK，并按照安装程序说明进行操作。如果使用默认安装路径，TivaWare SDK 将安装在 `C:\ti\TivaWare_C_Series-2.2.0.295`。有关 TivaWare 目录结构的信息，请参阅图 2-1（请注意，已有 `third_party/FreeRTOS` 目录）。TivaWare 库中安装的现有 FreeRTOS 为 8.2.3 版本。需要将其更新到最新版本。

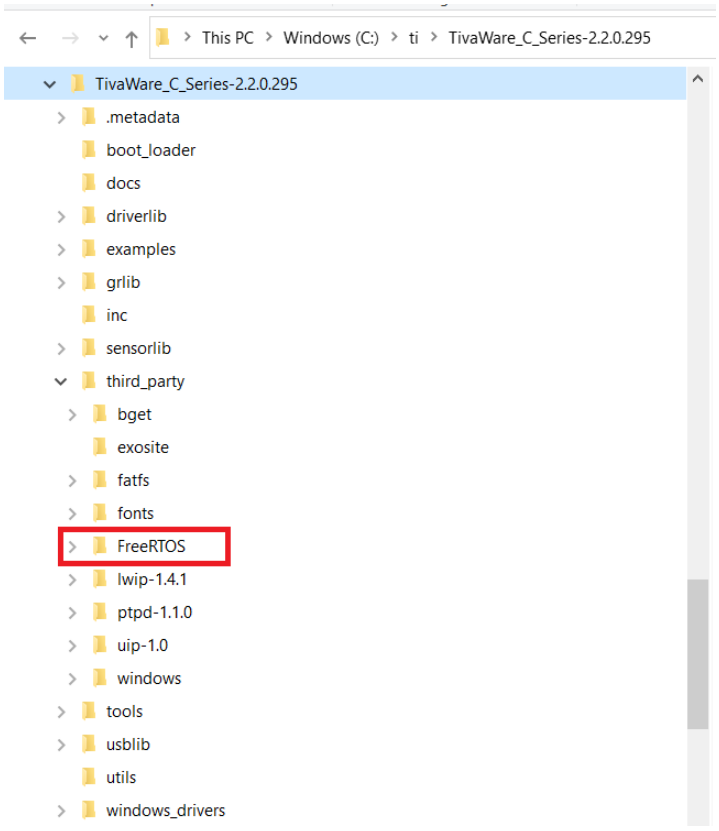

图 2-1. TivaWare 目录结构

2. 从此处的官方下载来源下载最新版本的 FreeRTOS。截至发布本应用报告时，FreeRTOS 的最新版本为 202112.00，如图 2-2 所示。本应用报告中提供的所有示例均已使用该版本创建和验证。

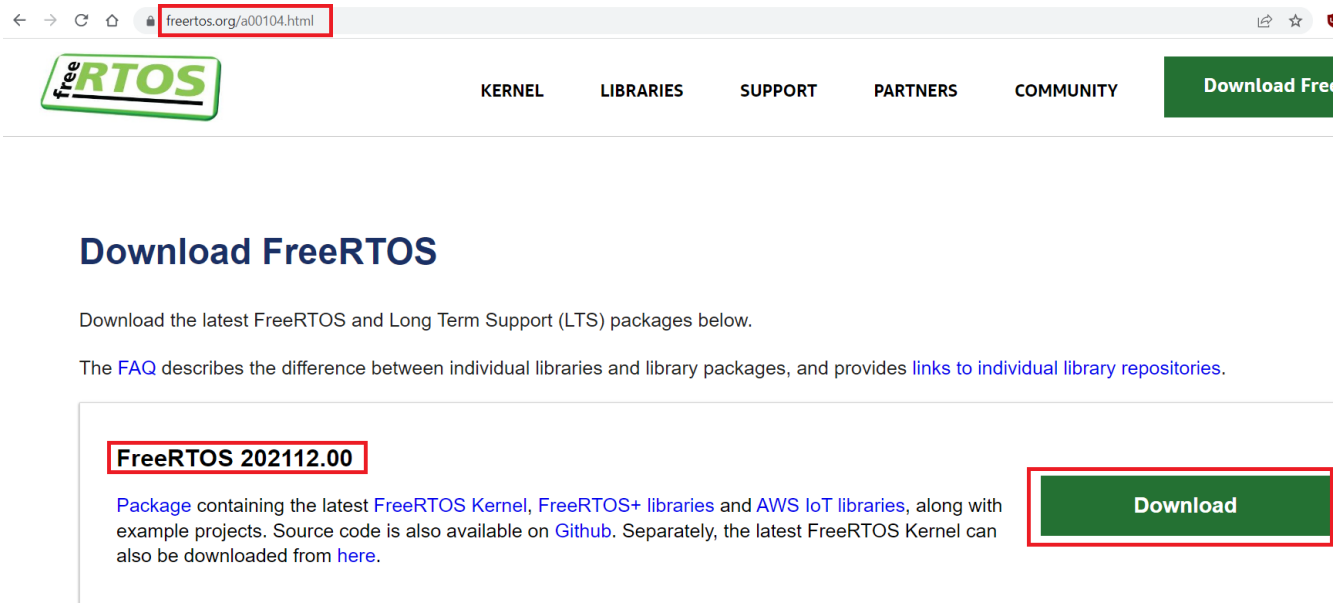

图 2-2. FreeRTOS 下载站点

3. 解压缩 FreeRTOS zip 文件并安装到一个临时目录中。查看 FreeRTOS 目录结构，请注意 FreeRTOS 文件夹以及其他文件夹和文件，如图 2-3 所示。

图 2-3. FreeRTOS 版本 202112.00 目录结构

2.1 更新 TivaWare 目录中的 FreeRTOS 版本

需要将新下载的 FreeRTOS 文件夹移动或复制到 `C:\ti\TivaWare_C_Series-2.2.0.295\third_party` 下的 TivaWare 库中。如果希望保留 TivaWare 库中的现有 FreeRTOS 文件夹作为备份，请确保首先重命名或移动该文件夹，如图 2-4 所示。

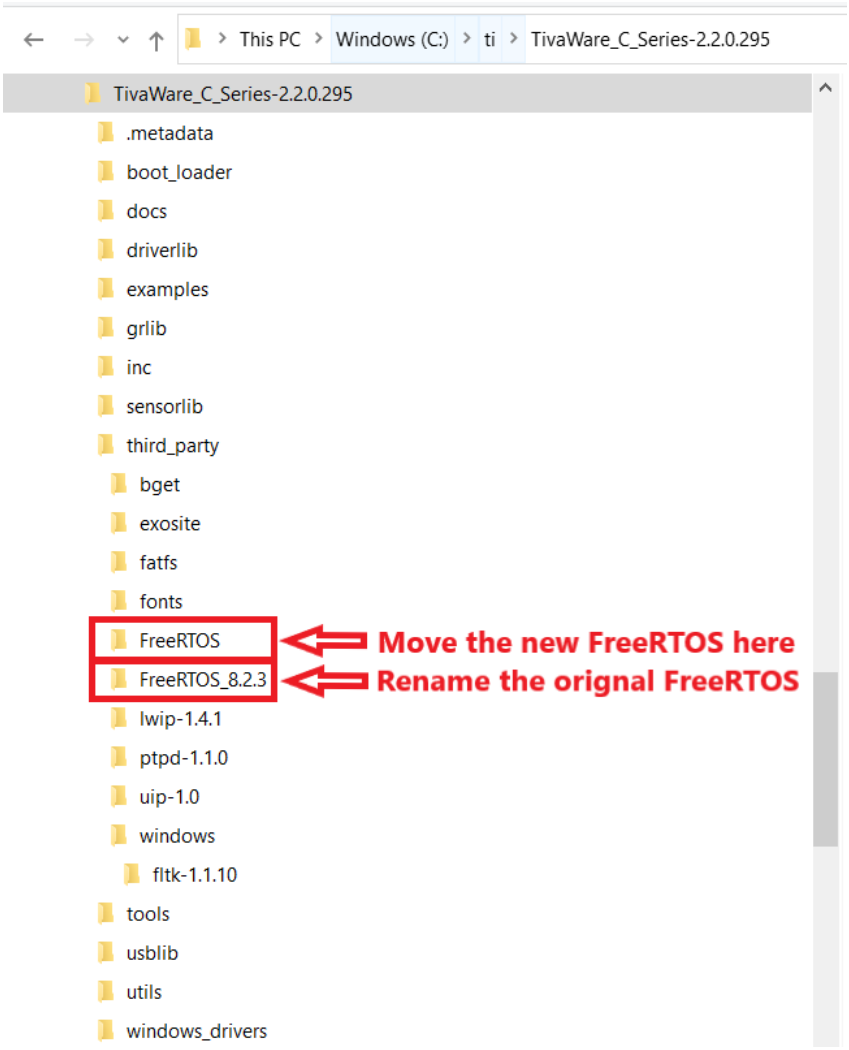

图 2-4. 更新 TivaWare，用于新 FreeRTOS 的安装

2.2 为 TM4C LaunchPad 添加 FreeRTOS 硬件驱动程序文件

提供的配套资料包括新的硬件驱动程序文件，这些文件旨在为每个 LaunchPad 评估套件的基本引脚排列配置提供单一来源。这些文件位于 `Driver Files` 目录下。使用器件引脚的所有示例中都引用了这些文件，因此在首次设置时需要执行此步骤。

每个特定 LaunchPad 的 *rtos_hw_drivers.c* 和 *rtos_hw_drivers.h* 文件应拷贝到该 LaunchPad 的 TivaWare 文件夹中，该文件夹位于 `drivers` 目录中。对于 EK-TM4C123GXL，此路径应为 `TivaWare_C_Series-2.2.0.295\examples\boards\ek-tm4c123gxl\drivers`，对于 EK-TM4C1294XL，此路径应为 `TivaWare_C_Series-2.2.0.295\examples\boards\ek-tm4c1294xl\drivers`。

3 TM4C FreeRTOS 示例的架构

提供的示例工程旨在提供一个紧凑、简化的基础，以便用于开始应用程序开发。为了实现此目标，FreeRTOS 内核将与 TivaWare 驱动程序库 (DriverLib) 结合使用，无需添加任何额外的抽象层。特定外设的配置和处理包含在专用任务文件中，可在多个工程之间轻松地重复使用。

目前，POSIX 未包含在此架构中。目前只部分实现了 FreeRTOS + POSIX 解决方案，仅通过 GitHub 提供。如需了解这些产品/服务的详细信息，请访问此链接。

除了 DriverLib 之外，TivaWare 还包含一些实用程序，可作为特定应用程序的补充，例如通过终端输出 UART 消息，或使用外部存储器源。请注意，这些实用程序是为裸机应用程序创建的，包含的 API 不能保证线程安全。为了简化演示，示例工程中有时会使用某些非线程安全 API。这些 API 将被明确地突出显示为非线程安全，只作为示例工程的一部分用于演示所执行的操作。

3.1 适当的时钟配置

为了确保 FreeRTOS 内核以正确的时钟频率运行，必须注意将 TM4C 微控制器的正确系统时钟频率分配给 FreeRTOS 内核。`FreeRTOSConfig.h` 中的 *configCPU_CLOCK_HZ* 变量保存系统时钟频率的值。无论系统时钟频率发生任何变化，都必须相应地更新此变量。提供的所有示例均使用目标微控制器支持的时钟速度上限，TM4C123x 器件为 80MHz，TM4C129x 器件为 120MHz。

在每个 TM4C 微控制器的器件特定数据表的*时钟控制*部分，提供了有效时钟速度选项的相关信息。根据时钟分频器设置的频率的精度是存在限制的。如果在 TivaWare 时钟配置函数中输入了不支持的频率，通常会配置为最接近的频率。如果时钟配置不正确（无法准确地反映外部晶体，或应使用的振荡器不正确），也会锁定器件，直到复位为出厂状态。最后，某些外设为了按照规格运行，对最低系统时钟频率有要求，这些外设包括 ADC（用于高速采样）、USB 和以太网（仅限 TM4C129x 系列）。

两个 TM4C 器件系列处理时钟配置的方式存在差异。

对于 TM4C123x 器件，系统时钟频率使用 `SysCtlClockSet` API 设置，只由通过函数调用传递的分频器确定。系统时钟的实际速率可由 `SysCtlClockGet` API 获得。但是，由于 `FreeRTOSConfig.h` 文件需要时钟速率的硬编码定义，因此对系统时钟频率的任何更改在插入 *configCPU_clock_Hz* 的新值之前，需要在内核未运行的情况下进行验证。

对于 TM4C129x 器件，系统时钟频率使用 `SysCtlClockFreqSet` API 进行设置，需要输入所需的系统时钟频率。然后它会返回实际系统时钟频率，该频率是根据与请求值最接近的值设置的。在提供的每个示例中，该值均存储在 *g_ui32SysClock* 中，可用于验证是否设置了正确的频率。提供给 `SysCtlClockFreqSet` 的时钟频率输入是 *configCPU_CLOCK_HZ* 变量。因此，只要定义了有效的 TM4C129x 时钟频率，工程就只需要进行一次更改。

3.2 硬件中断与 FreeRTOS 内核配合使用

在微控制器上管理 RTOS 的一个重要因素是正确处理微控制器硬件中断的使用，不会干扰 RTOS 内核和调度程序。由于 FreeRTOS 的目标是与芯片和编译器无关，因此没有处理特定器件硬件中断的独特插件。硬件中断处理发生在内核和调度程序之外，必须使用非常精简的中断服务例程 (ISR) 来处理。最大限度地减少执行 ISR 所花费的周期，可使调度程序更快地控制应用程序。

由于大多数 ISR 需要处理已触发的特定事件，因此 FreeRTOS 内核提供了将事件处理从 ISR 推迟到一项任务的方法，该任务是内核的一部分，由调度程序安排运行。这些推迟的任务已配置了优先级，使开发人员能够获得与 ISR 处理相同的灵活性，可以设置优先级，确保首先处理最重要的中断。通过这种方法，可以最大限度地减少硬件 ISR 处理，同时仍确保应用程序所需的优先级。

要了解有关推迟中断处理以及 FreeRTOS 中断安全 API 的更多详细信息，以及内核如何执行上下文切换，请参阅掌握 FreeRTOS 实时内核中的*中断管理*章节。

从 TM4C 微控制器的角度来看，配置中断包括将 ISR 映射到中断矢量表，以便在触发硬件中断时执行 ISR。有两种方法可以将新中断正确登记到中断矢量表中。*TivaWare™ 用户入门指南*的 *TivaWare 矢量表和 IntDefaultHandler* 部分对这些方法进行了详细说明。

3.3 Code Composer Studio 对 FreeRTOS 的调试支持

使用本应用报告中提供的固件时，Code Composer Studio 中可用的调试功能仅限于传统的裸机调试方法。TM4C 微控制器目前不支持包括实时对象视图 (ROV) 在内的高级 RTOS 特定方法。此类功能通常会随附完整的软件开发套件 (SDK) 版本，通常会提供额外的驱动程序、更广泛的 IDE 支持以及用于全面评估微控制器的各种工具。

可以使用能够与 FreeRTOS 内核连接的第三方工具来获取信息，包括任务的堆栈使用情况以及队列状态、信标等。但是，目前还没有官方支持的 FreeRTOS 和 TM4C 微控制器组合解决方案。如果 TM4C 微控制器需要全面的 ROV 支持，另一种 RTOS 选项是使用适用于 Tiva-C 的 TI-RTOS。借助 TI-RTOS，Code Composer Studio 中的 ROV Classic 工具可与 TM4C 微控制器配合使用。

4 示例工程演练

本应用报告包含一个 zip 文件，其中包含 EK-TM4C123GXL LaunchPad 评估套件和 EK-TM4C1294XL LaunchPad 评估套件的示例工程。.这些 Code Composer Studio™ 工程演示了如何将 TivaWare SDK 与 FreeRTOS 内核配合使用，以便在 TM4C 微控制器上创建应用。它们重点演示了 FreeRTOS 功能基本和 Arm 微控制器的基本外设。表 4-1 展示了示例列表。

表 4-1. FreeRTOS 应用示例

TM4C129	TM4C123
usb_dev_bulk	usb_dev_bulk
usb_dev_cdcserial	usb_dev_cdcserial
usb_dev_keyboard	usb_dev_keyboard
enet_lwip	
enet_io	

4.1 下载并导入示例

本应用报告提供的配套资料可以解压缩到本地目录中，也可以保留为 zip 格式。两种格式都可以导入 CCS。

1. 若要将工程导入 CCS，请首先选择“File”->”Import”。

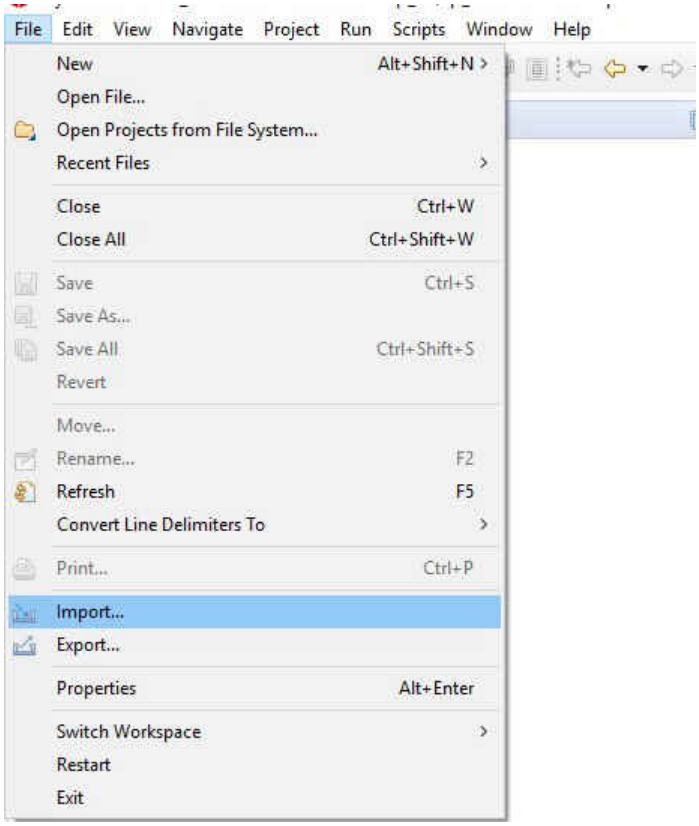

图 4-1. 导入 CCS 工程步骤 1

2. 选择“CCS Projects”以导入示例，然后点击“Next”。

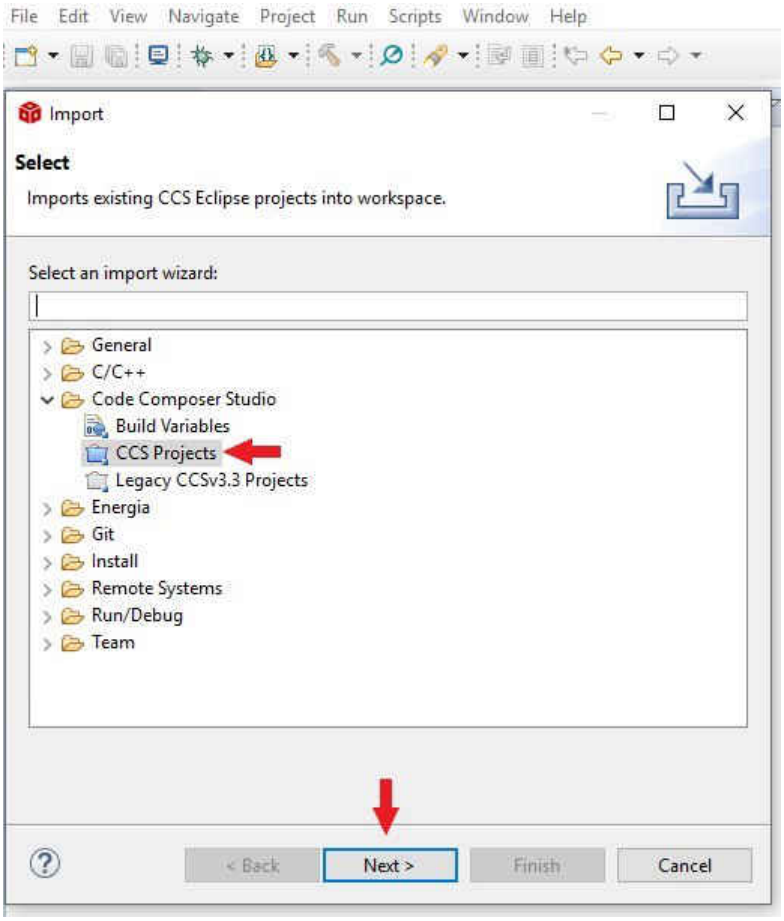

图 4-2. 导入 CCS 工程步骤 2

3. 接下来，提供解压缩工程（选择第一个单选按钮）或直接导入 zip 文件（选择第二个单选按钮）的路径。点击“Copy projects into workspace”。

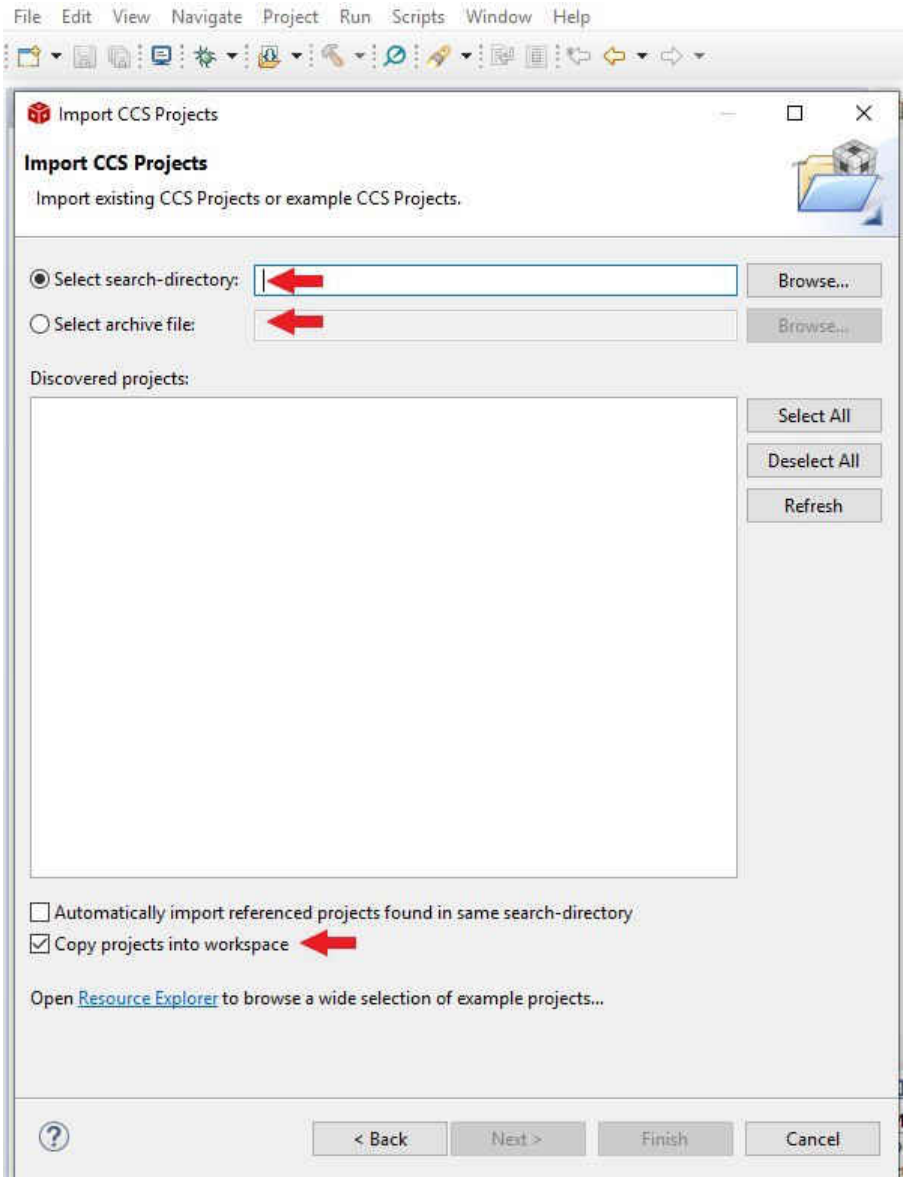

图 4-3. 导入 CCS 工程步骤 3

4. 提供工程路径后，总共会显示五个适用于 TM4C129 的已发现工程。首先点击“Select All”按钮，然后点击“Finish”按钮完成导入。

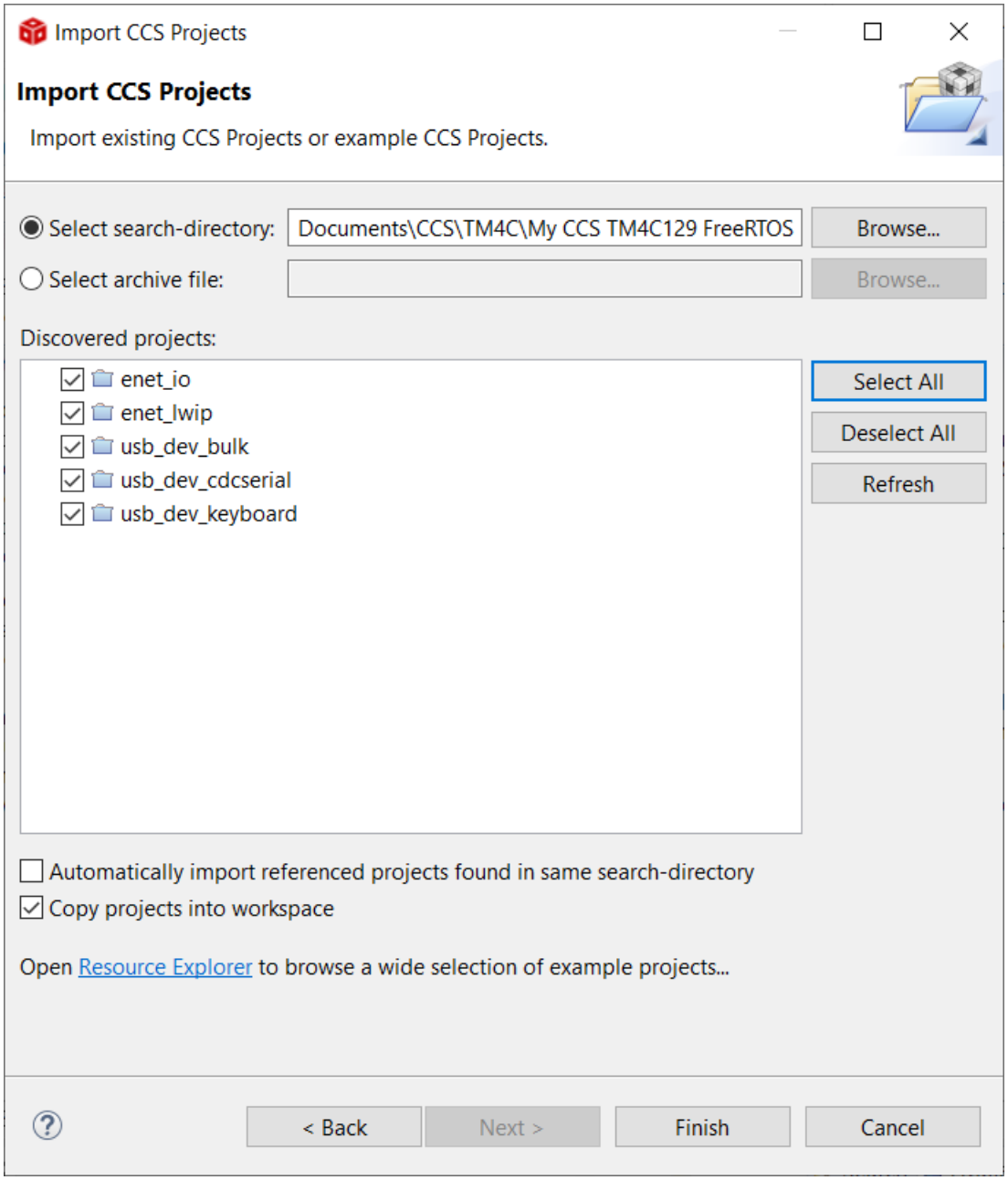

图 4-4. 导入 CCS 工程第 4 步

4.2 USB 示例

大多数 TM4C 微控制器都包含 USB 外设，但功能因器件而异。对于 TM4C129x 器件，所有器件都提供 USB 2.0 全速主机和设备支持，可以将高速 USB 2.0 用于外部 USB PHY。对于 TM4C123x 器件，所选的型号决定了是否有 USB 外设，或仅支持 USB 2.0 全速设备模式，或支持 USB 2.0 全速主机和设备。

两个 TM4C LaunchPad 使用的器件都提供 USB 2.0 全速主机和设备支持，但 EK-TM4C123GXL LaunchPad 需要对硬件进行修改才能使用 USB 主机模式，详细信息记录于*在 EK-TM4C123GXL LaunchPad 上使用 USB 主机模式*。要访问 USB 外设，必须将第二根电缆连接到 LaunchPad 上的辅助 USB 端口，或者必须调整电源配置，通过 USB 为电路板供电，在 EK-TM4C123GXL LaunchPad 上可使用开关完成电源调整，在 EK-TM4C1294XL LaunchPad 上可通过 JP1 上的跳线完成电源调整。

提供的三个示例仅适用于 USB 设备模式，但也可以使用 USB 主机模式的架构。这些示例涵盖的 CDC、批量和 HID 模式是最常用的 USB 模式。可以通过引用相应的 TivaWare 工程，添加对属于 TivaWare USB 库的其他 USB 模式的支持。

4.2.1 usb_dev_bulk

此示例将 USB 外设配置为批量模式的 USB 设备。为了观察 USB 批量传输，需要一个与批量模式 USB 设备通信的附加程序。TivaWare 中提供了 USB 批量通信示例程序，可从 SW-TM4C 工具页面下载。用于安装 USB 批量通信示例程序的软件包名为 *SW-USB-win-2.2.0.295.msi*。安装该程序后，导航至安装文件夹并找到 *usb_bulk_example.exe*。该应用程序在打开时会尝试连接到 TM4C USB 设备，因此在启动演示应用程序之前请加载并运行 usb_dev_bulk 示例工程。

在此示例中，只创建了一个任务来处理 USB 批量数据传输。计数信标用于支持 USB 处理程序指示何时通过批量通信接口接收到数据。接收到数据后，该任务将被释放，它将读取接收到的数据，然后通过批量模式 USB 设备返回这些数据，同时反转任何字母的大小写。与 *usb_bulk_examples.exe* 程序配合使用时，发送的数据将以 255 字节数据包的形式显示，因为这是 USB 2.0 全速标准下单个 USB 批量数据包的大小上限。可以发送大于 255 字节的消息，将通过多个返回数据包接收。

4.2.2 usb_dev_cdcserial

此示例将 USB 外设配置为 CDC 模式的 USB 设备，并将 UART0 外设配置为能够发送和接收数据。该示例演示了能够双向通信的 UART 转 USB 桥接器。要运行完整演示，需要两根 USB 电缆。一根电缆将连接到通过 UART0 运行的 Stellaris 虚拟 COM 端口，另一根电缆将连接到 TM4C 微控制器的 USB 总线（枚举为用作 CDC 设备时的串行 COM 端口）。使用 COM 端口终端软件，可以在 UART 和 USB 之间来回发送消息。

在此示例中，创建了三项任务来管理 UART 转 USB 桥接器。UART 转 USB 任务阻止了 UART RX 中断发出的一个任务通知。该任务释放后，会在读取 UART 数据的字节以通过 USB 进行回写之前，检查 USB TX 缓冲区的可用空间大小。它还会收集接收到的任何 UART 错误的相关数据，如果发生任何错误，则通过队列将信息发送给错误处理程序。

在 USB RX 处理程序发出任务通知以指示接收到数据之前，USB 转 UART 任务会一直被阻止。它将从 USB RX 缓冲区中读取 USB 数据并发送 UART 的字节。为了确保 UART TX FIFO 有足够的时间发送消息，在这个过程中会启用和监控 TX 中断。此外，互斥量用于保护对任务的访问，因为接收和处理 USB 消息的速度比完成 UART 传输的速度快。

最后一个任务是错误处理程序等待错误数据通过队列到达，如果收到数据，它会处理发生的错误并向 USB 堆栈报告，使 CDC 驱动程序能够知道发生了错误。

4.2.3 usb_dev_keyboard

此示例展示了如何将 USB 外设配置为 USB HID 设备模式，同时将 HID 设备枚举为 USB 键盘外设。将微控制器枚举为键盘，可以在屏幕上输出文本，就像用户使用键盘一样。输出的消息由 LaunchPad 上的按钮控制。一个按钮将始终输出默认的问候消息。另一个按钮可以输出第二条默认消息，可以使用 COM 端口终端向 Stellaris 虚拟 COM 端口发送新消息，以更新默认消息。该消息必须附加 CR-LF，用于标识消息已完成。默认情况下，消息大小限制为 255 字节，发送超过 255 字节的消息将导致消息被拆分。可以通过编辑 *UART_BUFFER_SIZE* 变量来增加工程中消息的默认大小。如果连续发送三条或更多条消息，而未输出任何消息，则只会存储前两条消息以供显示。

在此示例中创建了两项任务。第一项任务通过 UART 接收数据并通过队列将已完成的消息发送到 USB 输出任务，来管理动态消息的更新。检查消息是否超出大小上限，或者接收到的最后两个字节是否对应于回车换行 (CR-LF) 组合。一旦这些条件中有一个被满足，将使用队列将更新后的消息信息传递给 USB 输出任务。

处理 USB 键盘消息输出的第二项任务是在被阻止状态等待，直到按钮给出信标。按下一个按钮将触发该按钮的 ISR，为输出任务提供信标。然后输出任务会检查按下了哪个按钮，并发送相应消息。如果按下的按钮对应着可以更新的消息，则输出任务首先会检查队列中是否已从 UART 外设接收到新消息。

4.3 以太网示例

大多数 TM4C129x 微控制器均附带一个以太网外设，其中包含集成的以太网硬件组件。根据具体器件的不同，以太网外设可以包括一个可连接 MII/RMII 的集成以太网 MAC，以支持外部以太网 PHY，将以太网 MAC 和 PHY 集成到微控制器中（信号只能访问 PHY），或者将 MAC 和 PHY 与信号访问选项相集成（用于 PHY 和 MI/RMII 连接）。

EK-TM4C1294XL LaunchPad 具有一个 TM4C 微控制器，该微控制器具有集成的以太网 MAC+PHY 组合，该电路板还附带一个以太网插孔以连接外设。针对该 LaunchPad 提供了两个示例，以演示如何将轻量级 IP (lwIP) 协议栈与 FreeRTOS 配合使用，创建支持以太网的应用。

4.3.1 enet_lwip

此示例应用程序演示了使用 lwIP TCP/IP 协议栈的 TM4C 以太网控制器的操作。在此示例中，EK-TM4C1294XL LaunchPad 充当网络服务器。作为网络服务器，EK-TM4C1294XL LaunchPad 接收传入的网络 HTTP 请求，通过 TCP/IP 连接将传出的 HTTP 响应与网络内容一起发送。DHCP 用于获取以太网地址。如果 DHCP 超时且未获取地址，将使用 AutoIP 来获取链路本地地址。然后通过 UART 接口输出所选地址。

UART0 连接到 ICDI 虚拟 COM 端口，并以 115,200、8-N-1 运行，用于显示来自此应用程序的消息。请使用以下命令重新构建有改动的任何文件系统文件。

../../../../tools/bin/makefsfile -i fs -o enet_fsdata.h -r -h -q

有关 lwIP 的更多详细信息，请参阅 lwIP 网页，网址为 http://savannah.nongnu.org/projects/lwip/

4.3.1.1 运行 enet_lwip 示例

加载并运行应用程序固件后，该示例将显示各种信息，例如获取的 DHCP IP 地址和终端仿真器上的通信进度。可以使用连接到串行 COM 端口的任何终端仿真器。终端仿真器应配置为 115200 波特和 8-N-1。

如图 4-5 所示，在给定的网络上运行此示例时，获取的 IP 地址为 192.168.254.201。当同一示例在另一网络上运行时，将分配不同的 IP 地址。

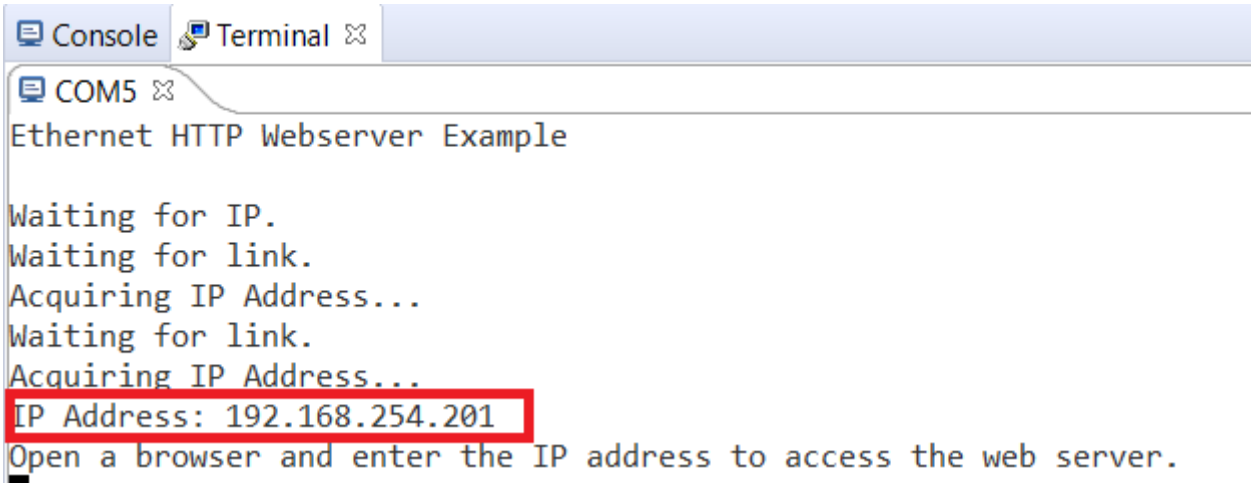

图 4-5. DHCP IP 地址

在浏览器的 URL 字段中输入终端窗口中显示的 IP 地址。网络服务器将在浏览器中显示 Web 内容。点击各菜单项，浏览不同的 Web 内容。

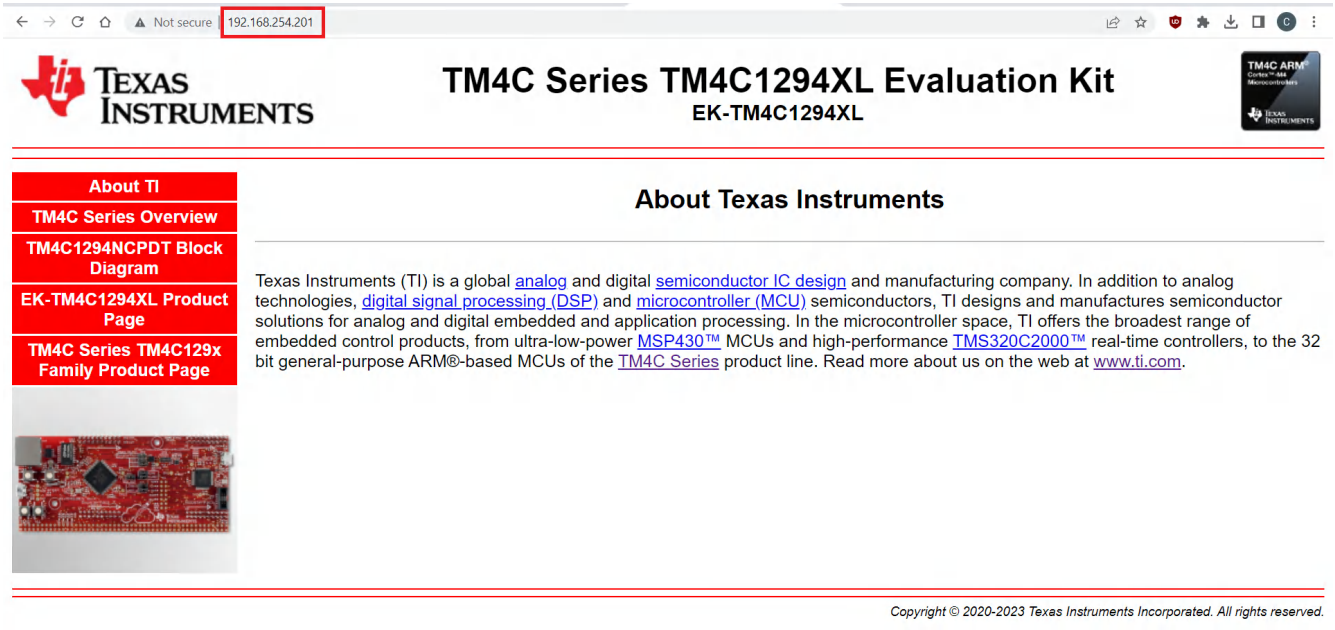

图 4-6. HTTP 网络服务器主页

4.3.2 enet_io

此示例应用程序演示了使用 TM4C 以太网控制器和 lwIP TCP/IP 协议栈的基于网络的 I/O 控制。DHCP 用于获取以太网地址。如果 DHCP 超时且未获取地址，将使用 AutoIP 来选择静态 IP 地址。所选地址将显示在 UART 上，允许用户通过普通 Web 浏览器访问该应用程序提供的内部网页。

应用程序主页左侧导航菜单中标记为 `IO Control Demo 1` 和 `IO Control Demo 2` 的页面介绍了通过网页控制电路板外设的两种不同方法。在这两种情况下，该示例都允许用户切换板载 LED 的状态，并设置 LED 的闪烁速率。

`IO Control Demo 1` 使用在 Web 浏览器中运行的 JavaScript，为特定 URL 发送 HTTP 请求。这些特定 URL 在文件系统支持层 (`io_fs.c`) 中被截获，并控制 LED 及动画效果。电路板生成的响应将返回浏览器，并通过更多 JavaScript 代码动态地插入页面 HTML 中。

IO Control Demo 2 使用标准 HTML 表单将参数传递至在电路板上运行的 CGI（通用网关接口）处理程序。在将响应页（在本例中为原始表单）发送回浏览器之前，这些处理程序会处理表单数据并根据请求控制动画效果和 LED。该应用程序会在初始化期间向 HTTPD 服务器注册每个 CGI 的名称和处理程序，并且在每次请求一个 CGI URL 时，服务器都会在解析 URL 参数后调用这些处理程序。

有关第二个演示中各种控制状态的信息，将使用 SSI（服务器端包含）标签插入提供的 HTML 中，这些标签由应用程序中的 HTTPD 服务器解析。与 CGI 处理程序一样，该应用程序在初始化期间向网络服务器注册其 SSI 标签列表和处理程序函数，只要在提供给浏览器的 .shtml、.ssi、.shtm 或 .xml 文件中找到任何已注册的标签，就会调用此处理程序。

除了 LED 和动画速度控制，第二个示例还允许将一行文本发送到电路板以输出到 UART。其目的是演示 HTTP 文本字符串的解码。

此示例中的网络服务器已根据基本 lwIP 包中提供的示例进行了修改。增加的功能包括 SSI 和 CGI 支持，还能够使服务器自动插入 HTTP 标头，而不是将这些标头内置到文件系统映像中的文件中。

内部文件系统映像的源文件可在 fs 目录中找到。如果更改了其中的任何文件，必须从 enet_io 目录运行以下命令，来重建文件系统映像 io_fsdata.h：

../../../../tools/bin/makefsfile -i fs -o io_fsdata.h -r -h -q

有关 lwIP 的更多详细信息，请参阅 lwIP 网页，网址为 http://savannah.nongnu.org/projects/lwip/

4.3.2.1 运行 enet_io 示例

与 enet_lwip 示例类似，此示例将在终端窗口中显示获取的 DHCP IP 地址。在浏览器 URL 中输入 IP 地址。网页显示的 Web 内容与 enet_lwip 类似，但有另外三个菜单项。点击“IO Control Demo 1”和“IO Control Demo 2”菜单项，控制板载 LED 和动画速度。

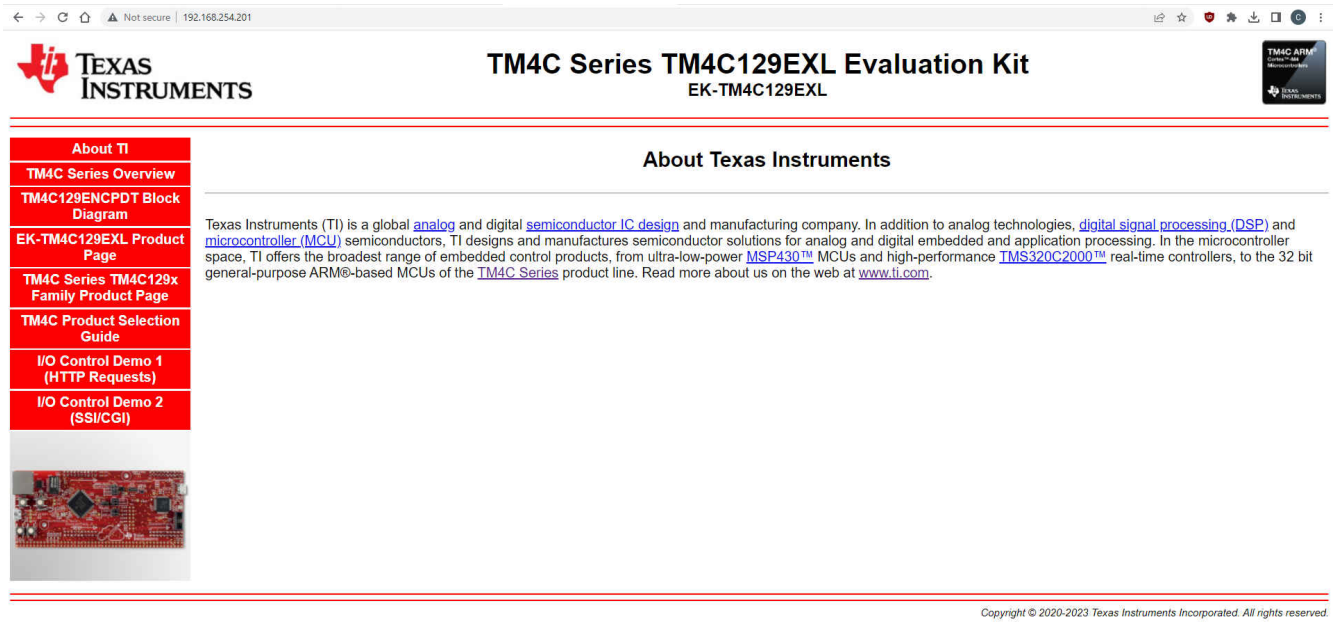

图 4-7. 以太网 IO 主页

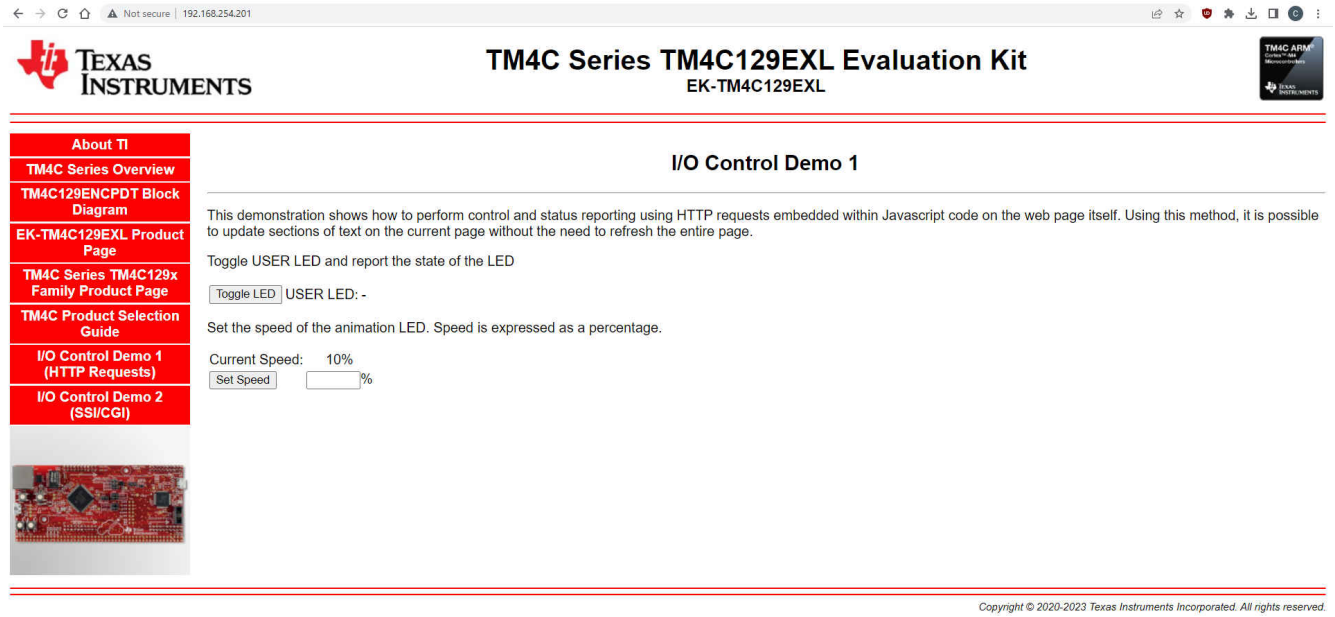

图 4-8. 使用 HTTP 请求的 IO 控制演示 1 网页

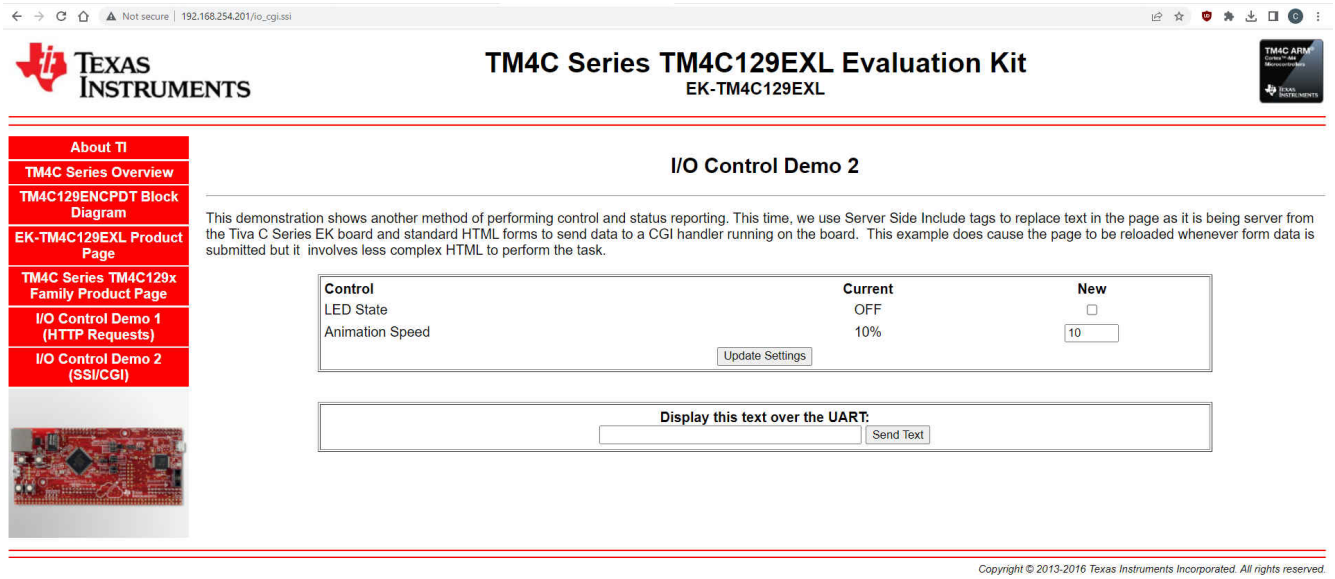

图 4-9. 使用 SSI/CGI 的 IO 控制演示 2 网页

5 参考文献

- 德州仪器 (TI)：*在 TM4C MCU 上使用 FreeRTOS 开发基本应用*
- 德州仪器 (TI)：*在 TM4C MCU 上使用 FreeRTOS 开发常见应用*
- 德州仪器 (TI)：*TivaWare™ 用户入门指南*
- 德州仪器 (TI)：*在 EK-TM4C123GXL LaunchPad 上使用 USB 主机模式*

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265