Application Note

TM4C129x 基于套接字的 TI-RTOS NDK 以太网应用

Charles Tsai

摘要

TM4C129x MCU 是高度连接的 32 位 Arm® Cortex®-M4F 微控制器，配有集成式以太网 MAC 和 PHY 以及各种有线通信外设。本应用报告展示了基于 TI-RTOS NDK（在 TI 嵌入式处理器上开发网络使能应用的平台）的各种以太网应用示例。这些示例面向 EK-TM4C1294XL LaunchPad™ 开发套件，该套件是适用于 TM4C129x 以太网 MCU 的评估平台。

本文档所讨论的工程配套资料和源代码可从以下 URL 下载：https://www.ti.com/cn/lit/zip/spma080。

内容

插图清单

表格清单

商标

LaunchPad™ and Code Composer Studio™ are trademarks of Texas Instruments.

1 引言

Network Developer’s Kit (NDK) 是一个平台，用于在 TI 嵌入式处理器上开发和演示支持网络的应用。NDK 中包含的代码是通用 C 代码，可在各种 TI 器件上运行。

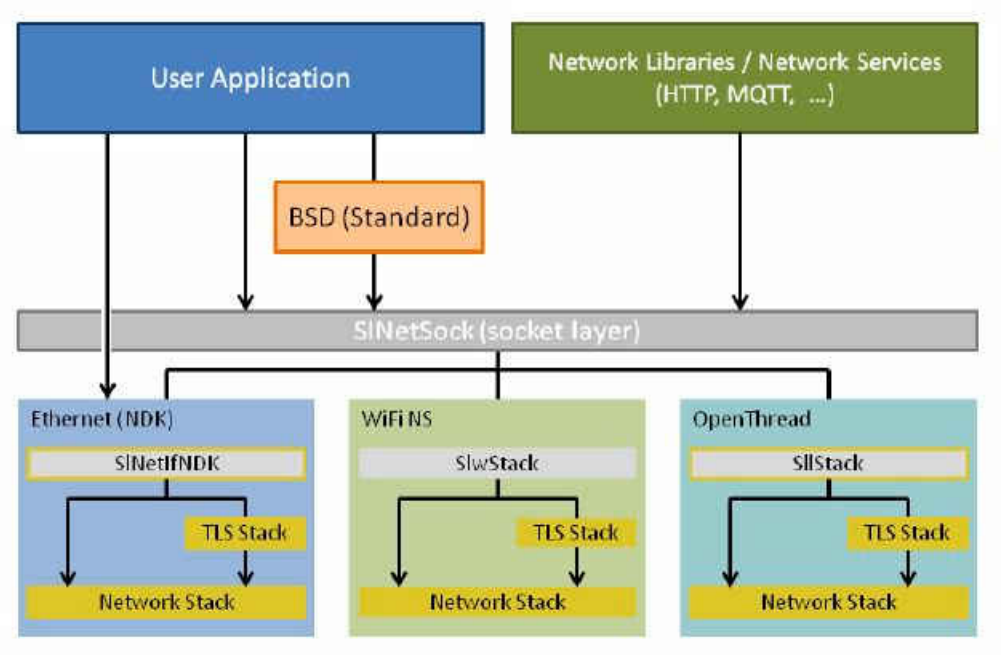


图 1-1. 启用 NDK 的应用的组件

在图 1-1 中，用户应用可以使用标准 BSD 套接字 API 进行调用，也可以直接调用 SlNetSock 层与网络连接进行通信。SlNetSock 层是用户应用和服务特定栈之间一个独立于栈的层。

NDK 可提供独立于平台、独立于器件和独立于 RTOS 的 API 接口以供应用使用。

NDK 是 TI-RTOS 的网络组件，TI-RTOS 是面向 TI 器件的可扩展嵌入式工具生态系统。TI-RTOS 从实时多任务内核 (SYS/BIOS) 扩展到包括中间件组件和器件驱动程序的完整 RTOS 解决方案。

本应用报告的主要重点是说明 TI-RTOS NDK 的各种以太网应用，假设用户对 TI-RTOS 和 NDK 有一些基本的了解，如果您不熟悉 TI-RTOS 和 NDK，请参阅下面的文档了解详细信息：

- *TI Network Developer’s Kit (NDK) 用户指南*
- *TI Network Developer’s Kit (NDK) API 参考指南*
- TI-RTOS-MCU 概述 – 产品页面
- *适用于 TivaC 的 TI-RTOS 入门指南*
- *TI-RTOS 用户指南*
- *SYS/BIOS（TI-RTOS 内核）用户指南*

1.1 TI-RTOS 下载

针对 Tiva C 的 TI-RTOS 的下载地址：http://downloads.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/tirtos/index.html。

1.2 许可

TI-RTOS 遵守 BSD 许可。下载 TI-RTOS 时，请查看许可协议。

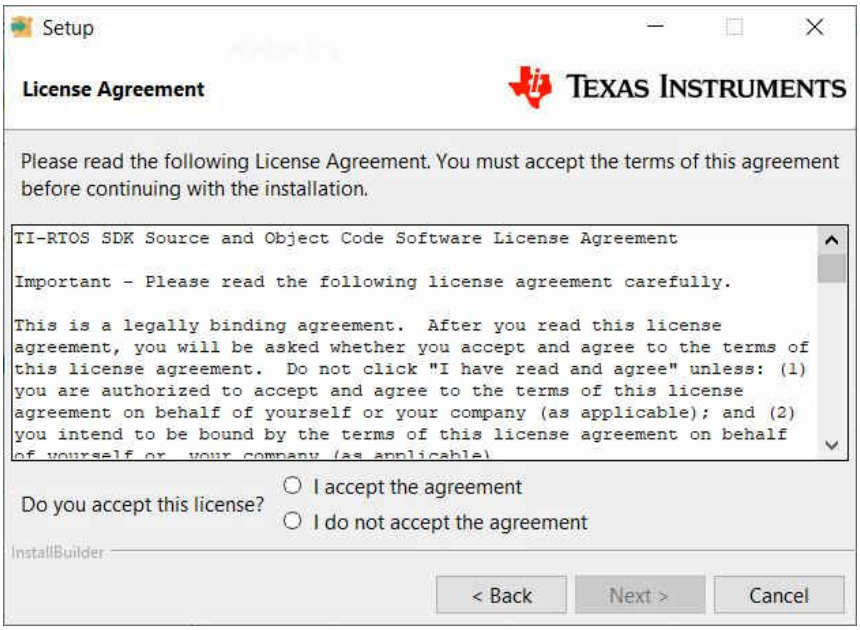


图 1-2. TI-RTOS 许可协议

1.3 XDCtools

XDCtools 是德州仪器 (TI) 提供的独立软件组件，它提供对 SYS/BIOS、NDK 和 UIA 进行配置和编译所需的底层工具。通常，XDCtools 会作为 Code Composer Studio™ (CCS) 程序的一部分自动安装。

在 TI-RTOS 程序中，仅当 TI-RTOS 所需版本尚未随 CCS 或 SYS/BIOS 一同安装时，才会安装 XDCtools。

1.4 版本

包含 NDK 版本 2.25.00.09 的 TI-RTOS 版本 2.16.xx.xx 是 Tiva C (TM4C) 开发最新版本。

为 Tiva C (TM4C) 推荐的 XDCtools 版本是 v3.32.0.06。

备注

对于 Tiva C (TM4C)，不要使用 2.16.xx.xx 以后的任何 TI-RTOS 版本，不要使用 3.32.0.06 以后的任何 XDCtools 版本。

1.5 配置 NDK 模块

要配置 NDK 及其组件，NDK 允许您使用 C 代码调用函数或使用 CCStudio 中的 XGCONF 配置工具。

- C 代码：通过编写 C 代码来配置应用，代码调用 *CfgNew()* 来创建配置数据库，并调用其他 *Cfg*()* 函数向配置数据库添加各种设置。详细信息请参阅 SPRU523 的 2.1 节。推荐使用此配置方法，因为它可以与 SDK 支持的任何 RTOS 一起使用。
- XGCONF：XGCONF 是 CCStudio 中的一个 GUI，用于启用和设置属性。使用 TI-RTOS 内核模板创建工程时，工程将包含一个配置文件 (*.cfg)，可以使用 CCSstudio 中的 XGCONF 图形编辑器对该文件进行编辑。配置文件是在编译期间处理的，从而生成配置应用的代码。仅当 RTOS 是 TI-RTOS 内核时，才能使用此配置方法。

本应用中提供的所有示例都使用 XGCONF 来配置 NDK 模块。

1.6 基于套接字的 API

本应用报告中提供的应用示例使用标准 BSD API 进行套接字操作，例如接受、发送和接收。如图 1-1 所示，标准 BSD API 是通过 SINetSock 提供的，也可以使用 NDK 套接字 API 函数。NDK 套接字 API 在 NDK_accept()、NDK_send() 和 NDK_receive() 等中的前缀为 NDK_*()。虽然 NDK_*() 套接字调用占用的空间更小，但性能略好且差异小。

1.6.1 BSD 套接字 API

表 1-1 列出了典型的 BSD 套接字 API。图 1-3 显示使用 BSD 套接字 API 的 TCP 客户端-服务器通信的简化流程图。图 1-4 显示了 UDP 通信的 API 使用情况。

表 1-1. BSD 套接字 API

功能类别	API	说明
套接字连接	socket	创建某个类型的新套接字，由整数标识，并为其分配系统资源。
	bind	通常用于服务器端，将套接字与套接字地址结构（即指定的本地 IP 地址和端口号）关联起来。
	listen	用于服务器端，使绑定的 TCP 套接字进入监听状态。
	accept	用于服务器端。它接受接收到的从远程客户端创建新 TCP 连接的传入尝试，并创建与此连接的套接字地址对相关联的新套接字。
	connect	用于客户端，为套接字分配空闲的本地端口号。对于 TCP 套接字，它会导致尝试建立新的 TCP 连接。
接收数据	recv	用于接收数据。通常仅用于已连接的套接字。
	recvfrom	用于接收数据。可能用于在套接字上接收数据，无论它是否面向连接。
发送数据	send	用于发送数据。通常用于 TCP SOCK_STREAM 已连接的套接字。
	sendto	用于发送数据。通常用于 UDP SOCK_DGRAM 未连接的数据报套接字。
I/O 多路复用	poll	用于检查一组套接字中某个套接字的状态。可以测试该集合以查看是否可以写入、读取任何套接字或是否发生错误。
	select	用于挂起，等待提供的套接字列表中一个或多个套接字准备好读取、准备好写入，或者是因为存在错误。
关闭连接	close	关闭 TCP 连接。
套接字选项	setsocketopt	用于为指定的套接字设置特定的套接字选项。
	getsockopt	用于检索指定套接字的特定套接字选项的当前值。

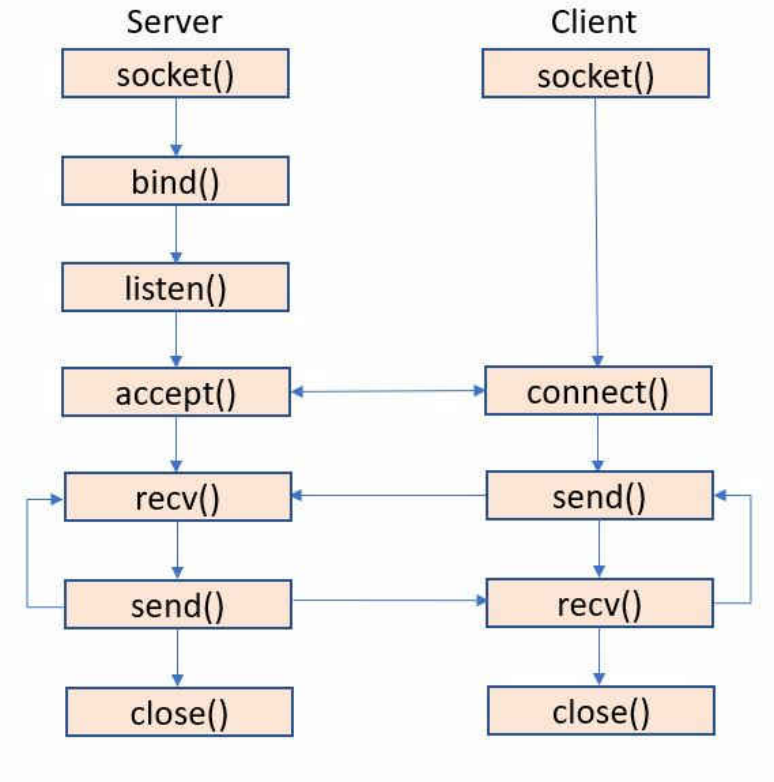


图 1-3. 使用 BSD 套接字 API 的 TCP 客户端服务器通信

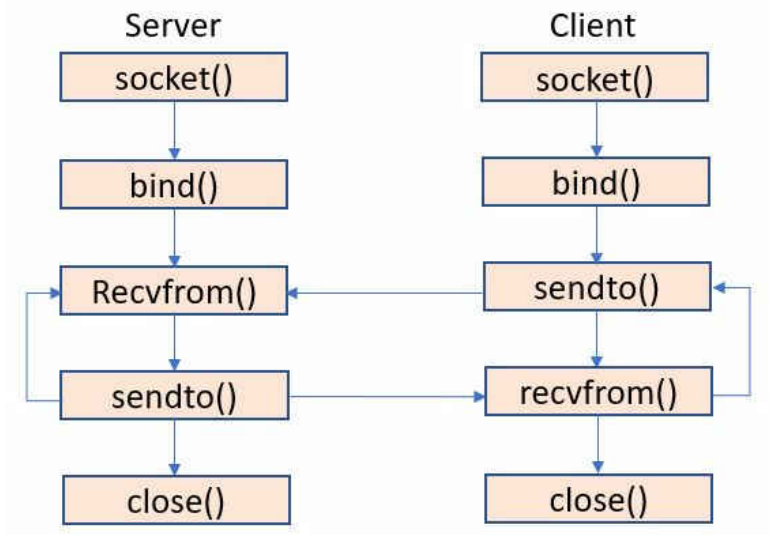


图 1-4. 使用 BSD 套接字 API 的 UDP 客户端服务器通信

2 应用示例

此处共提供了八个示例，用来展示使用 TI-RTOS NDK 运行以太网服务器应用或客户端应用的 TM4C129x MCU。

表 2-1. 应用示例

示例	类型	说明
enet_tcpecho_server_tirtos	服务器	使用 TCP 协议的回显服务器应用。服务器回传从客户端接收的数据包。动态 IP 地址是从 DHCP 服务器获取的。
enet_updecho_server_tirtos	服务器	使用 UDP 协议的回显服务器应用。服务器回传从客户端接收的数据报。

表 2-1. 应用示例 (continued)

示例	类型	说明
enet_httpServer_tirtos	服务器	托管网页的 HTTP 网络服务器应用。
enet_dns_tirtos	客户端	一个客户端应用，它请求 DNS（域名服务器）将域名转换为 IP 地址，从而使 DNS 客户端能够访问源服务器。
enet_sntp_tirtos	客户端	基于 SNTP（简单网络时间协议）报告当前网络时间的客户端应用。
enet_tcpecho_client_tirtos	客户端	使用 TCP 协议的回显客户端应用。客户端向服务器发送问候消息，并回传从服务器接收的数据包。
enet_udpecho_client_tirtos	客户端	使用 UDP 协议的回显客户端应用。客户端向服务器发送问候消息，并回传从服务器接收的数据报。
enet_httpget_tirtos	客户端	向互联网服务器发送 HTTP GET 请求的客户端应用。

3 应用设置

3.1 硬件设置

- 连接到互联网的网络路由器
- 将设备连接到 LAN 网络的以太网交换机。如果可以将设备直接连接到路由器，则交换机是可选项。大多数家用路由器都具有 LAN 端口，用于与设备进行有线连接。
- PC 用于：
  - 调试目标器件。
  - 作为服务器或客户端，来生成或响应进出目标器件的网络流量。
  - 监控网络流量。
- 运行本应用报告中提及应用的 EK-TM4C1294XL LaunchPad 评估套件。

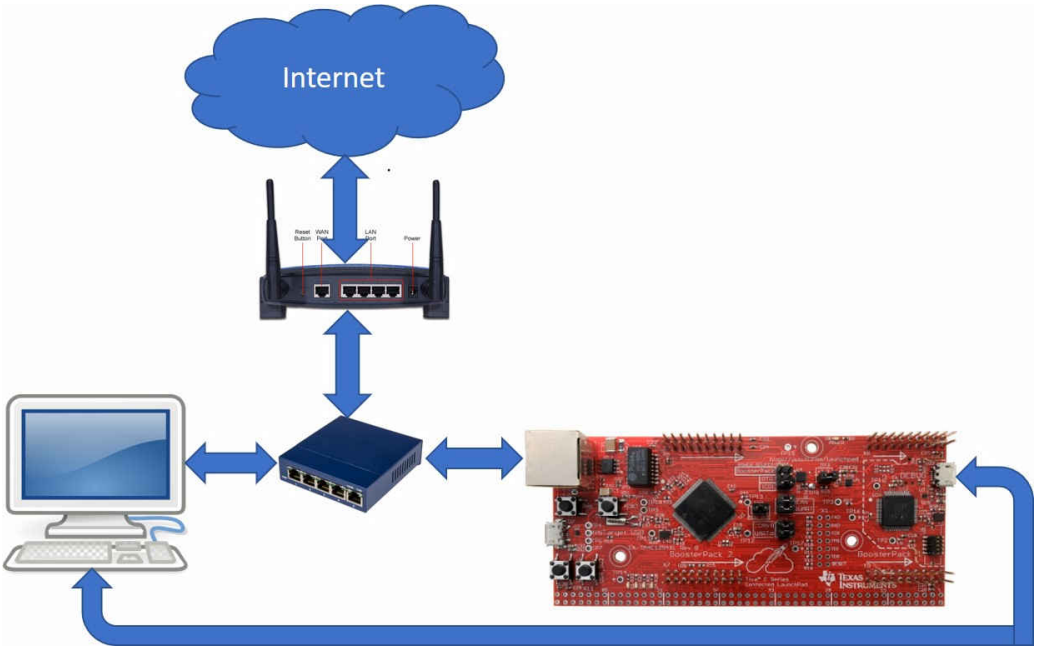


图 3-1. 应用示例的硬件设置

3.2 软件工具

有几种工具可以简化应用示例的调试和测试：

- CCS。该 IDE 工具用于调试目标器件和编译应用示例。本应用报告使用 CCS v10.1.1 和 TI 编译器 v20.2.4.LTS。
- SocketTest。一个免费的小型软件工具，用于测试使用 TCP 或 UDP 协议进行通信的任何服务器或客户端。
- Wireshark。此应用报告中用于网络故障排除和分析的免费数据包分析器。

4 下载并导入以太网示例

本文件附加了八个 CCS 工程示例作为配套资料。您可以解压缩该工程，或将其保留为 zip 格式。CCS 可以导入这两种格式。

1. 若要将工程导入 CCS，首先选择“File”（文件）-> “Import”（导入）。

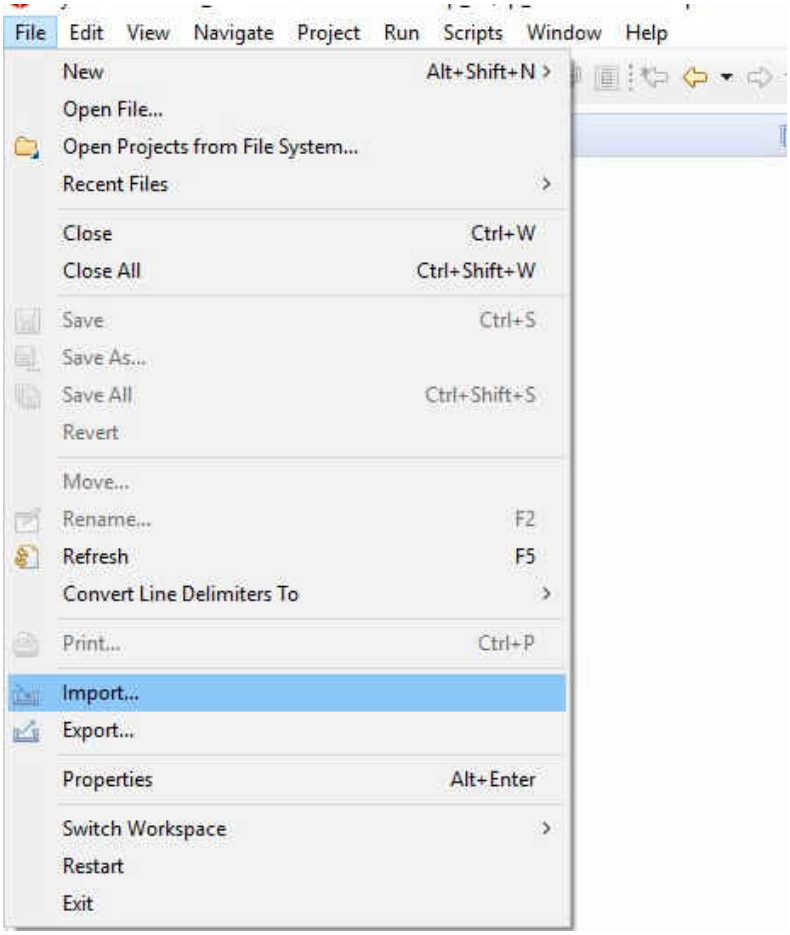


图 4-1. 导入 CCS 工程步骤 1

2. 选择“CCS Projects”以导入示例，然后点击“Next”（下一步）。

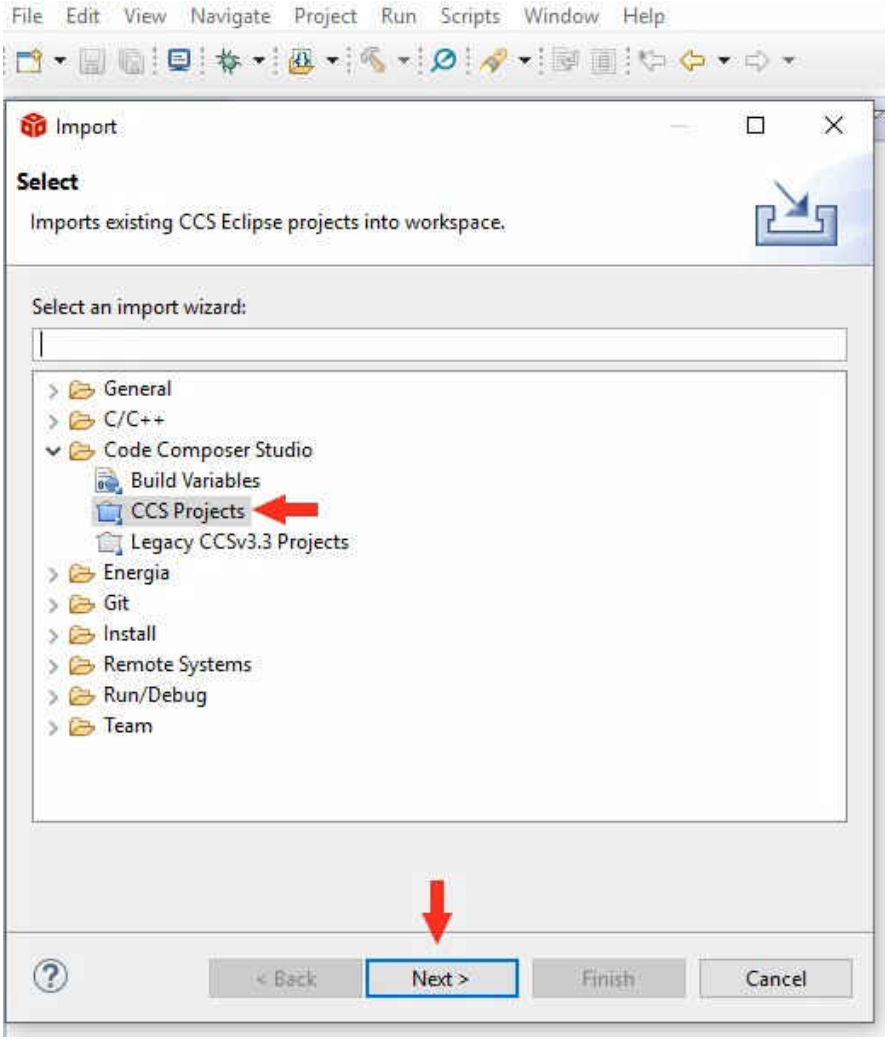


图 4-2. 导入 CCS 工程步骤 2

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”（将工程复制到工作区）。

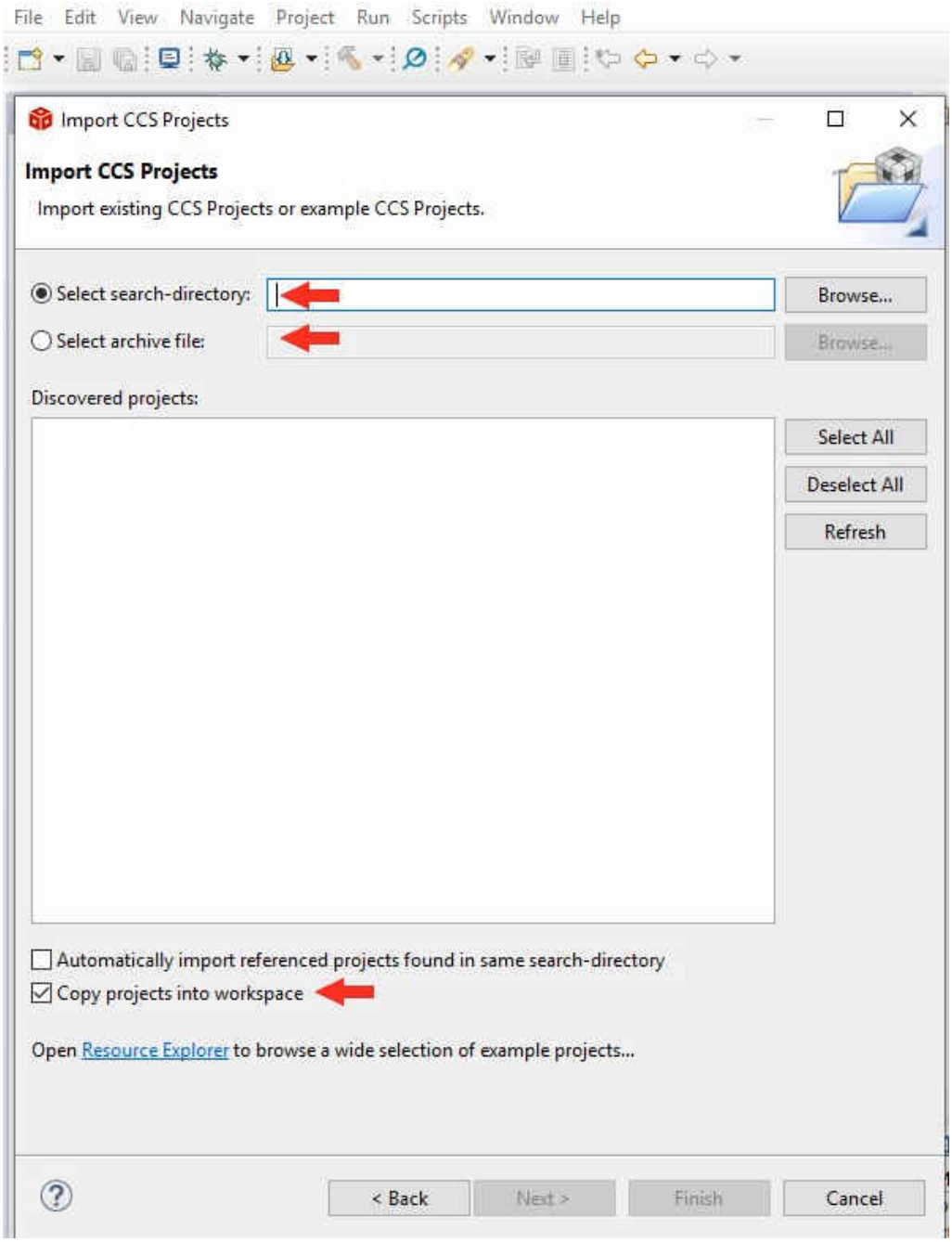


图 4-3. 导入 CCS 工程步骤 3

4. 提供工程路径后，将显示总共八个已发现的工程。首先点击“Select All”（全选）按钮，然后点击“Finish”（结束）按钮完成导入。

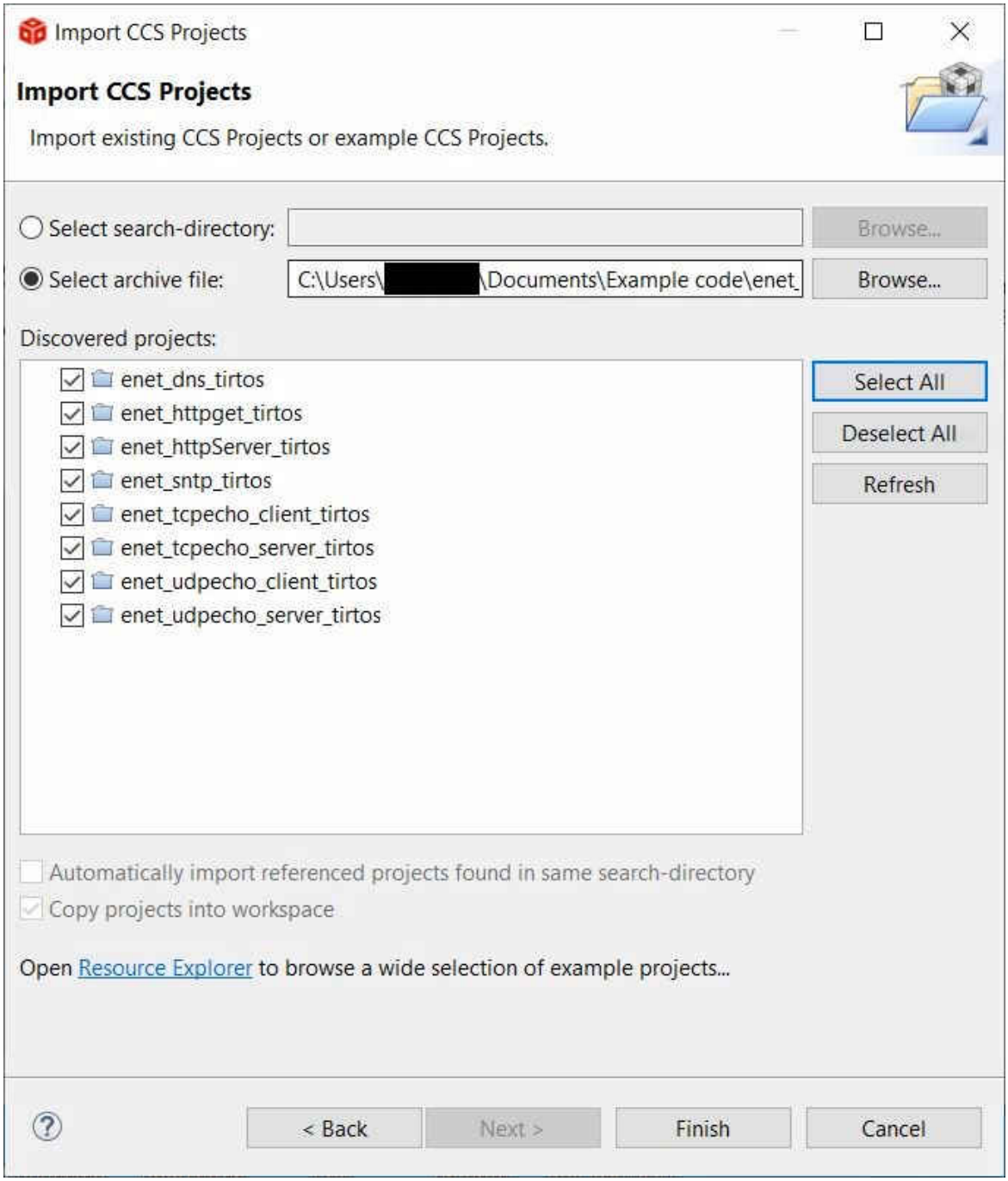


图 4-4. 导入 CCS 工程步骤 4

5 如何为 TI-RTOS NDK 创建以太网应用

开始开发 TI-RTOS NDK 应用的最佳方法是参考现有示例。按照以下步骤查看 enet_tcpecho_server_tirtos 示例的 NDK 配置。

1. 如果如节 1.5 所述，XGCONF 是用于配置的方法，则每个 TI-RTOS 工程都将包含一个 *.cfg 文件。*.cfg 用于添加和配置各种 TI-RTOS 和 NDK 模块和参数。首先，右键点击 *.cfg 文件，请参阅图 5-1。
2. 在“Open With”子菜单下选择“XGCONF”。完成后，您将首先看到 TI-RTOS 欢迎页面。
3. 点击“Outline”选项卡中的“Global”将进入 NDK 欢迎页面。接下来，点击框 5 中显示的“System Overview”选项卡。
4. 各种 NDK 模块将以图形方式显示，如图 5-1 中所示。例如，您将在“Transport Layer”中看到 TCP、UDP 和 NAT 模块。根据应用对模块/功能的需要，可以点击这些模块以启用它们。在此示例中，只启用了 TCP、UDP 和 IP 模块。本应用报告中的其他示例可能需要启用不同的 NDK 模块。启用后，可以看到一个绿色复选标记。例如，点击互联网协议 (IP) 模块将进入 IP 模块配置，如图 5-2 所示。在这里，可以启用该模块并选择自动获取动态 IP 地址，您也可以提供静态 IP 地址。
5. 在方框 5 中，各种全局 NDK 设置划分在不同的选项卡中。点击它们中的每一个可以了解相应的信息。请参阅图 5-3，其中“netOpenHook”定义为用户提供的挂钩函数。当栈准备好开始创建由应用提供的网络任务时，会调用“Network open hook”函数。在工程的 tcpEchoHooks.c 文件中，可以看到这个函数的定义。其他示例可能使用不同的挂钩函数。例如，enet_tcpecho_client_tirtos 示例使用“Network IP address hook”挂钩函数，这是因为客户端将在尝试连接到服务器之前等待获得 IP 地址。
6. 对于 TI-RTOS 和 NDK 配置，您也可以直接编辑类似脚本语言的 *.cfg 文件。

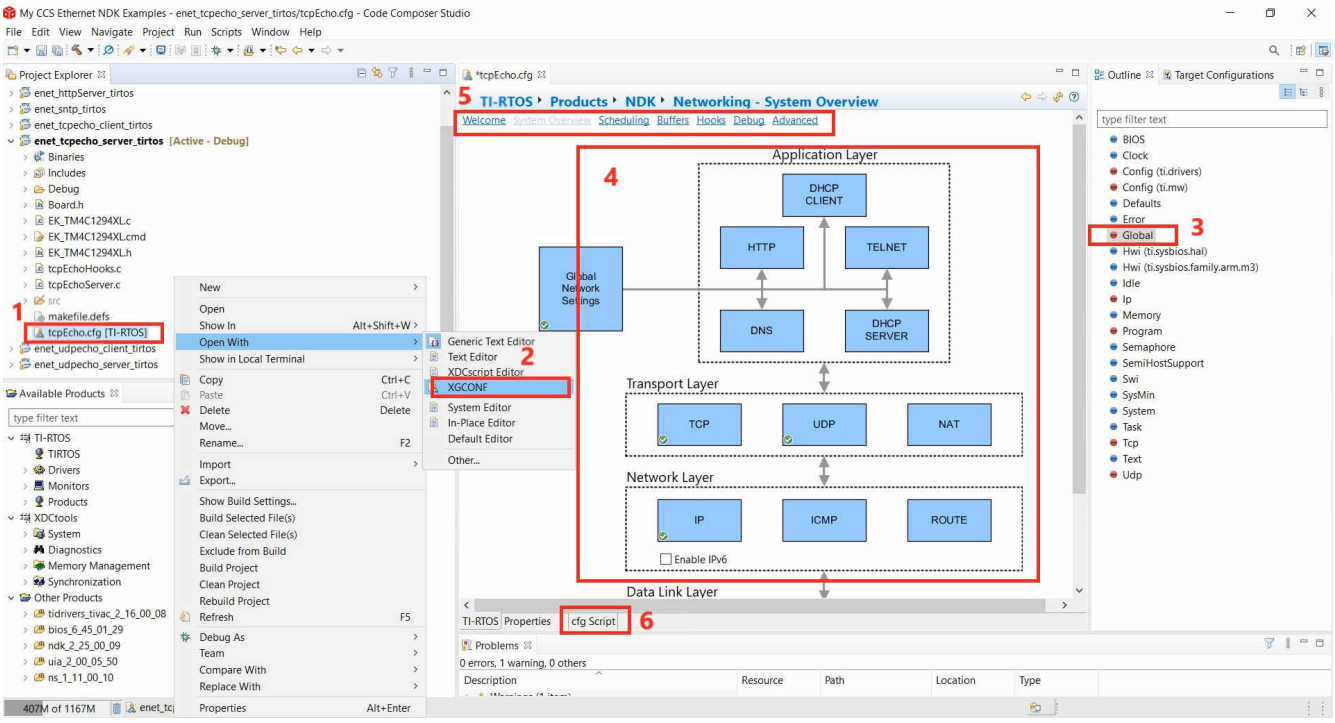


图 5-1. 使用 XGCONF 的 NDK 配置

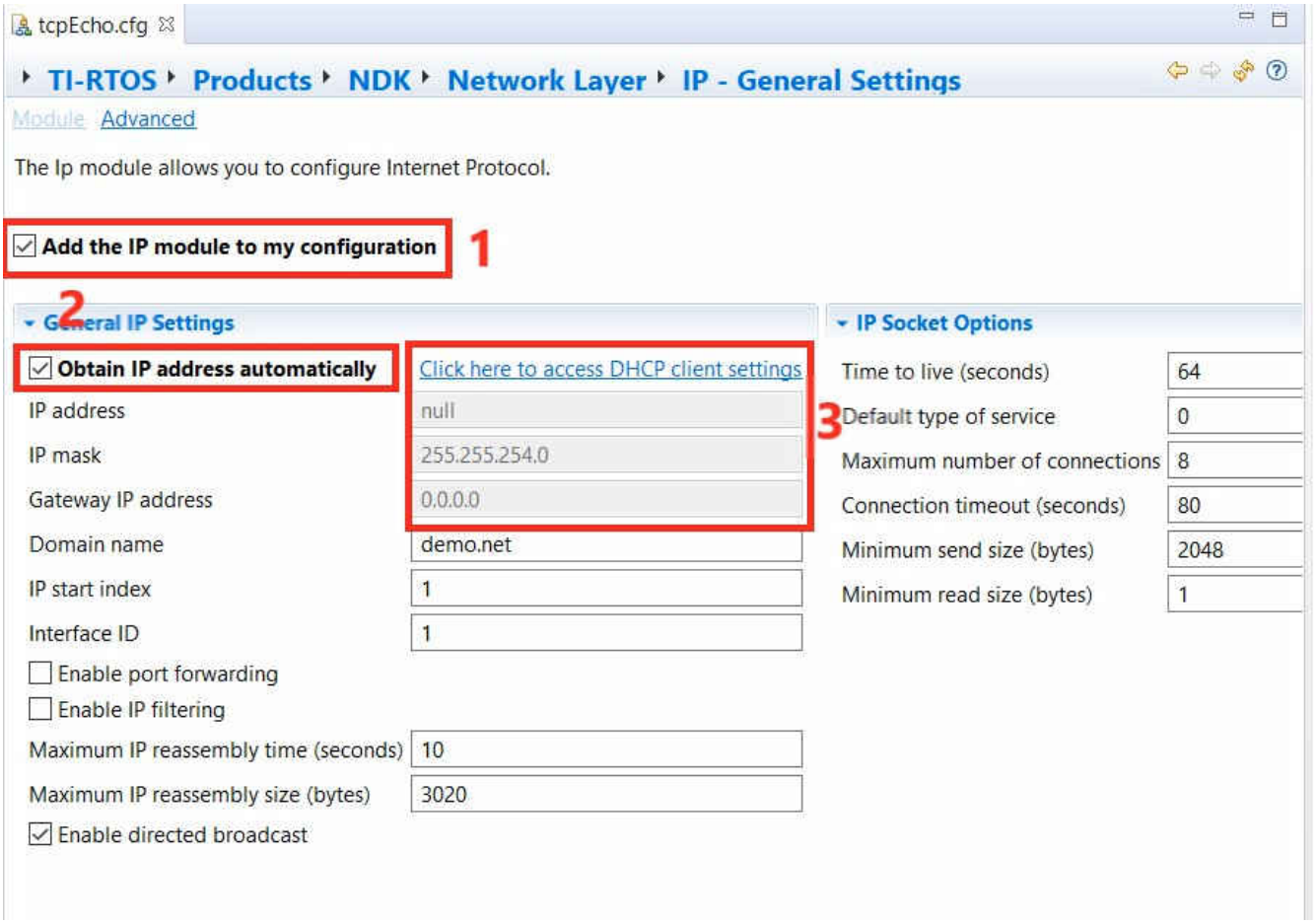


图 5-2. IP 模块配置

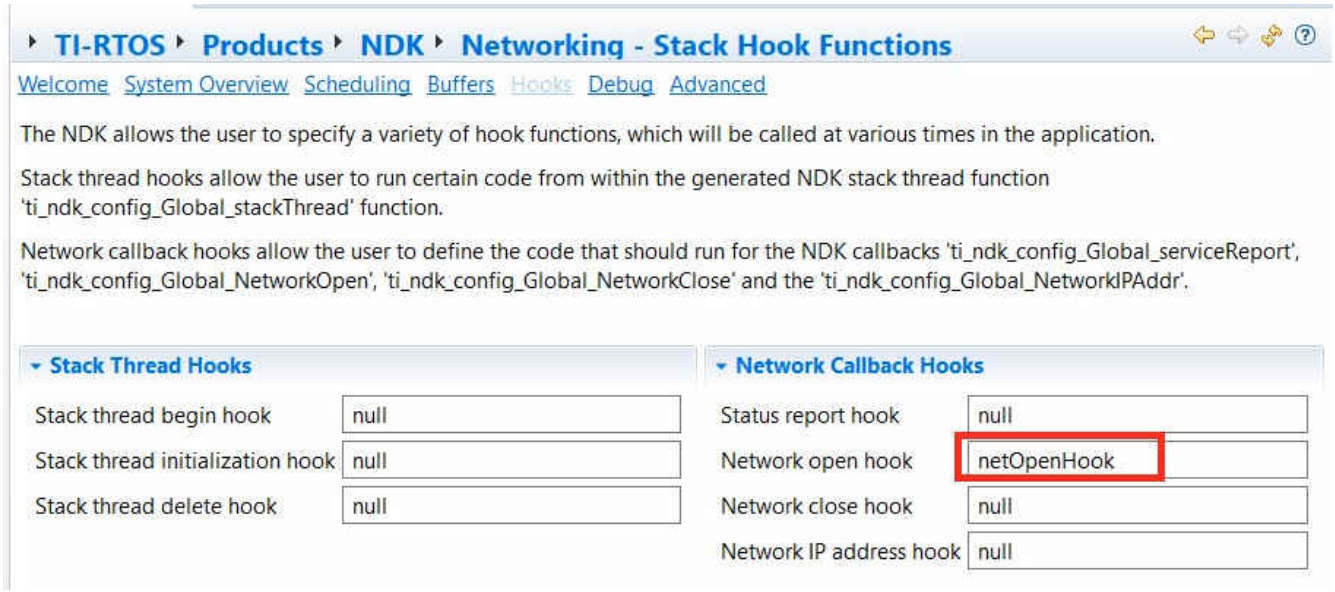


图 5-3. 挂钩函数配置

6 Enet_tcpecho_server_tirtos 示例概述

enet_tcpecho_server_tirtos 示例演示了一个在 TM4C129x MCU 上运行、使用传输控制协议 (TCP) 作为底层传输层协议的回显服务器应用。TCP 是一种面向连接的协议，具有内置的错误恢复和重新传输功能。这种连接协议类似于电话连接。拨打方和接听方都需要握手连接（例如，拨打方拨号，接听方拿起电话）才能交流。连接一直存在，直到一方挂断连接。当需要保证无差错的消息传递时，应用使用 TCP。

在本例中，TM4C129x MCU 用作服务器。NDK 栈配置为 DHCP 以自动获取 IP 地址。获取后，IP 地址将显示在 CCS“Console”窗口中。此时，回显服务器准备就绪。服务器将监听来自客户端的连接。一旦客户端建立起连接，服务器和客户端之间的通信就可以开始。本例中实现的服务器将读取接收到的字符，然后将这些字符回传给客户端。

6.1 构建和刷写程序

首先选择 enet_tcpecho_server_tirtos 作为当前工程。将 LaunchPad 的 ICDI USB 端口通过 USB 电缆连接到 PC，通过点击“Debug”图标来编译工程并加载程序。

如果您是 CCS 新手，请点击 CCS 用户指南。

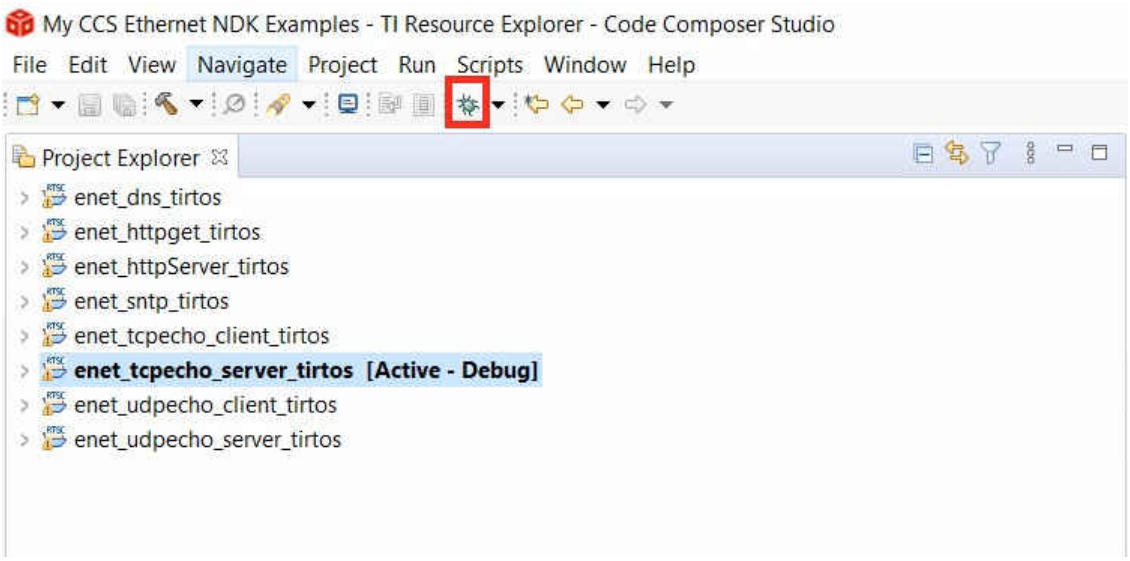


图 6-1. 调试 CCS 工程

6.2 对 MAC 地址进行检查和编程

网络上的每个 NIC（网络接口控制器）都必须由一个 MAC 地址唯一标识，以便在网段内进行通信。MAC 地址是一个 48 位值，表示为两个十六进制数字的六个八位字节。MAC 地址主要由设备制造商来分配。前三个八位字节是 OUI（组织唯一标识符）。MAC 地址通常在 EK-TM4C1294XL LaunchPad 板上预先编程。LaunchPad 的背面还有一个贴纸，上面写着 MAC 地址。预编程的 MAC 地址的前三个八位字节等于 00:1A:B6，用来唯一标识德州仪器 (TI)。如果您拥有原始状态的设备，则 MAC 地址未预先编程。您必须自行使用分配给贵组织的地址对 MAC 地址进行编程。

提供三种工具可用于对 MAC 地址进行读取和编程。

6.2.1 使用 LM Flash Programmer（闪存编程器）

如果您在使用 EK-TM4C1294XL LaunchPad，此工具最合适。LM Flash Programmer 仅支持 LaunchPad 上内置的 ICDI 调试探针。若要对 MAC 地址进行检查和编程，须遵循以下步骤。

1. 打开 LM Flash Programmer 并转到“Other Utilities”（其它实用程序）选项卡。
2. 选择“MAC Address Mode”（MAC 地址模式）单选按钮。
3. 若要读取 MAC 地址，请按下“Get Current MAC Address”（获取当前 MAC 地址）按钮。
4. 若要对 MAC 地址进行编程：
   a. 首先在“MAC Address”（MAC 地址）字段中键入六个八位字节的 MAC 地址。
   b. 接下来，点击“Commit MAC Address”（提交 MAC 地址）复选框。提交后，MAC 地址将永久存储在内部 EEprom 上。如果未选中该提交选项，则刚刚输入的 MAC 地址将在下一次下电上电前临时保存。
   c. 最后，按下“Program MAC Address”（编程 MAC 地址）按钮完成编程。

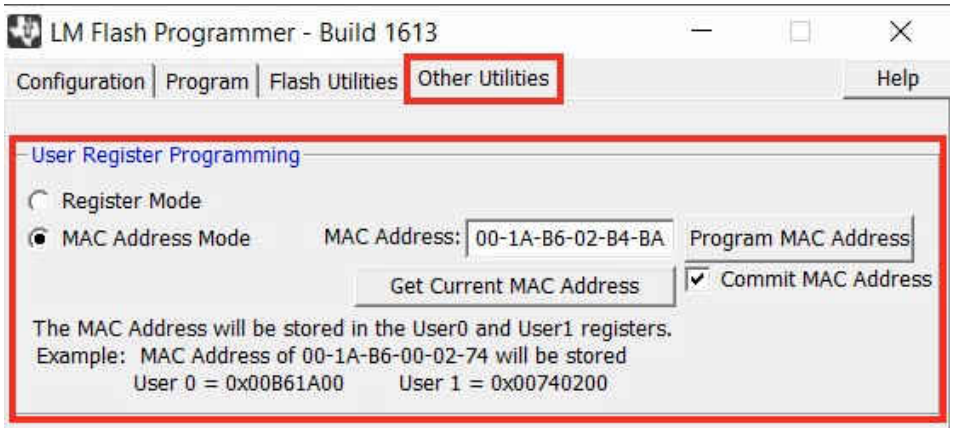


图 6-2. 使用 LM Flash Programmer 进行 MAC 地址编程

6.2.2 使用 CCS

CCS 还具有一个内置实用程序，可用于对 MAC 地址进行编程。如果您选择使用 CCS IDE 进行软件开发，这会非常合适。

若要对 MAC 地址进行读取和编程，请先转到“Tools”->“On-Chip Flash”。对 MAC 地址进行读取和编程的步骤与节 6.2.1 中所述的步骤相同。

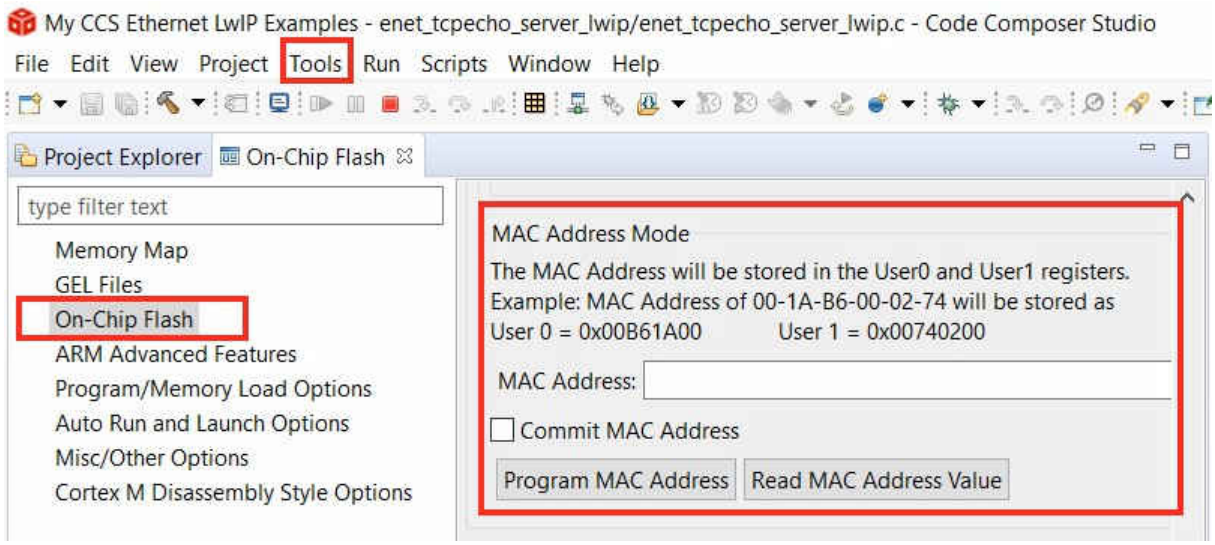


图 6-3. 使用 CCS 对 MAC 地址进行编程

6.2.3 使用 UniFlash

UniFlash 是一个 TI 独立工具，支持对各种 TI 器件进行编程，包括 TM4C129x MCU 的 MAC 地址。在定制电路板上对 MAC 地址进行编程而调试探针不是 Stellaris ICDI 时，最适合使用 UniFlash，尽管 Stellaris ICDI 也支持 ICDI。

若要对 MAC 地址进行读取和编程，请先转到“Settings and Utilities”选项卡。对 MAC 地址进行读取和编程的步骤与节 6.2.1 中所述的步骤相同。

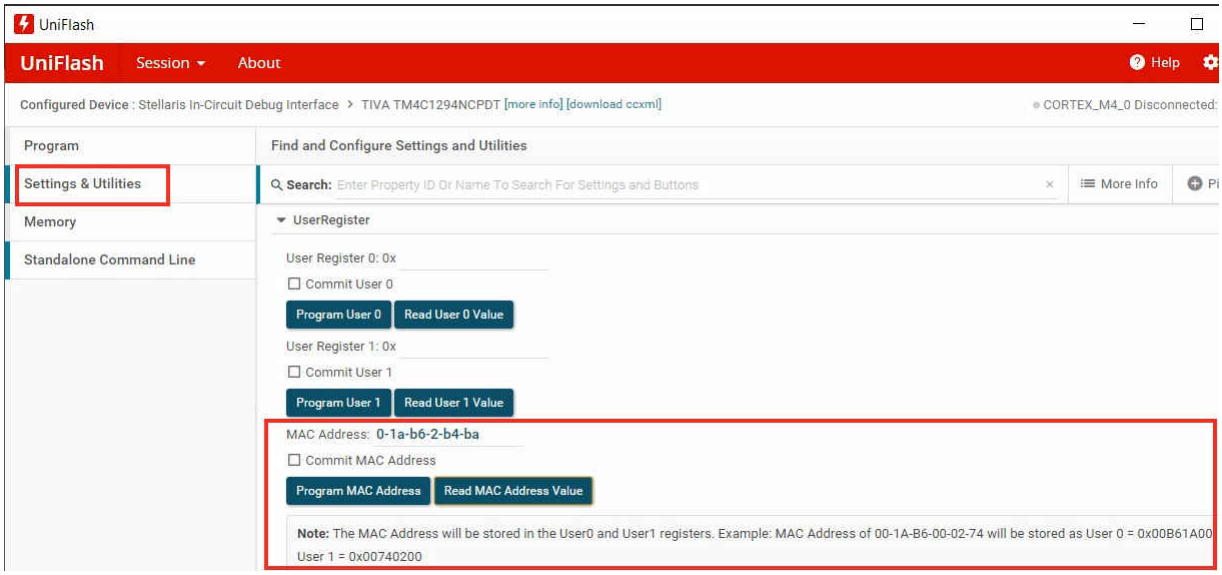


图 6-4. 使用 Uniflash 对 MAC 地址进行编程

6.3 运行 enet_tcpecho_server_tirtos 示例

使用以太网电缆将 EK-TM4C1294XL LaunchPad 连接到以太网交换机或路由器，如图 3-1 所示。运行示例。打开 CCS“Console”窗口后，您应该会看到显示的 IP 地址（箭头所指）并且服务器已准备就绪，如图 6-5 所示。记录此 IP 地址，因为客户端上需要此信息。最初，服务器将处于侦听状态，等待客户端与其连接。因此，若要继续运行示例的其余部分，需要设置远程客户端。

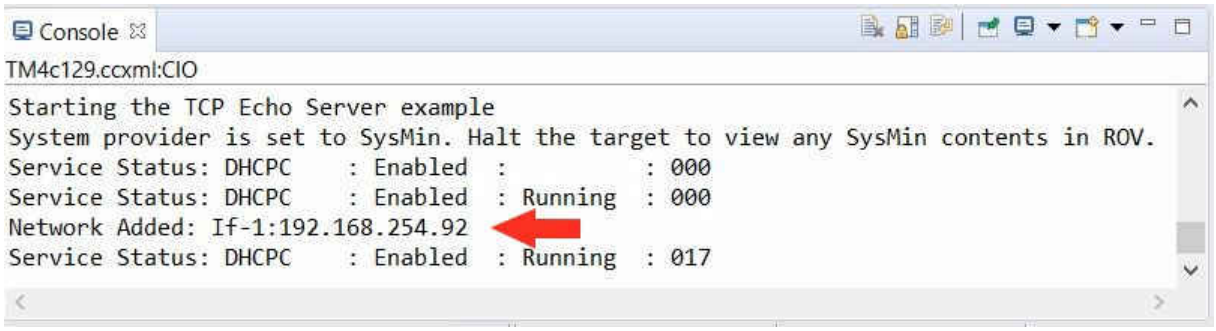


图 6-5. Enet_tcpecho_server_tirtos 输出

使用将在 PC 上作为客户端运行的 SocketTest 工具。确保 PC 连接到与 EK-TM4C1294XL 具有相同子网掩码的网络。

按照图 6-6 中所示的步骤设置客户端：

1. 打开 SocketTest 并输入服务器 IP 地址以及端口号 23。端口 23 是 TCP 和 UDP 协议中的默认 Telnet 端口号。最后，按下“Connect”（连接）按钮。与服务器的连接很快就会建立，然后您便可以与服务器进行对话了。
2. 转到“Message”（消息）字段并输入一些消息，然后点击“Send”（发送）按钮。
3. 您输入的消息将显示在对话字段中。服务器收到该消息后，会将该消息回传给客户端。

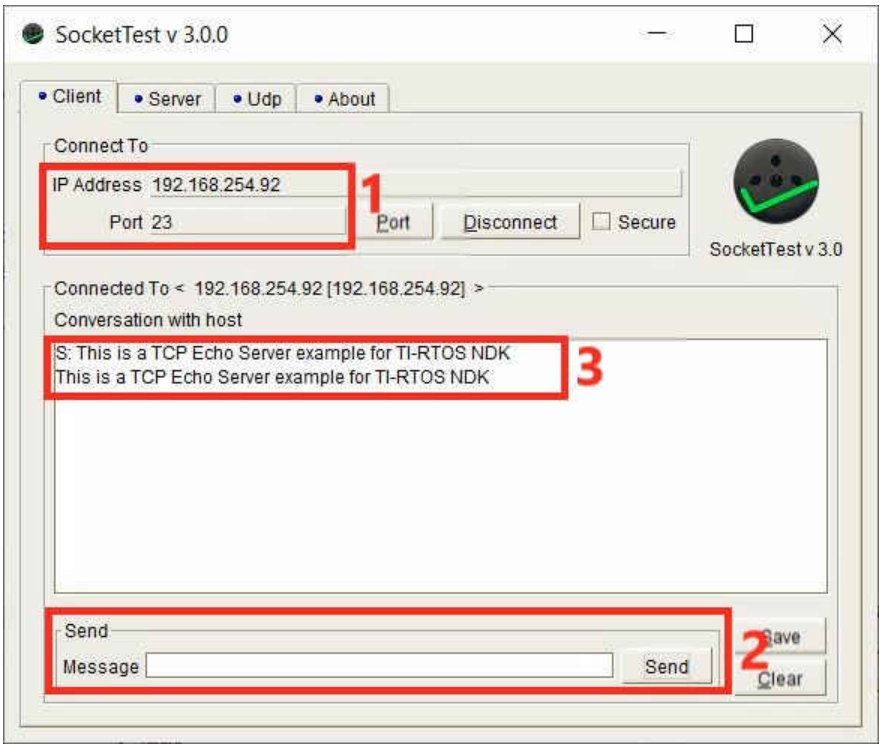


图 6-6. Enet_tcpecho_server_tirtos 的 SocketTest 客户端配置

检查客户端发送到服务器的消息“This is a TCP Echo Server example for TI-RTOS NDK”。如果您选择手动计数，可以计算出总长度为 51 个字节，包括两个 \n\r 转义字符。\n 是换行符，\r 是 ASCII 表中的回车符。

从会话字段中可以看出，服务器回传了完全相同的消息。

此外，检查图 6-7 中 Wireshark 捕获的以太网流量。

1. 客户端（IP 地址 192.168.254.69）将该消息发送到服务器（IP 地址 192.168.254.92）。如需更多信息，请参阅图 6-5 了解获取的服务器 IP 地址。
2. 服务器（IP 地址 192.168.254.92）将该消息回传给客户端（IP 地址 192.168.254.69）。
3. 在 Wireshark 中，点击任何一行，都会展开事务详细信息。点击帧编号 10938 会显示回传消息的详情。
4. 可以看到，服务器回传消息的长度确实是 51，这与客户端发送的消息长度相同。
5. 显示的服务器回传消息内容包括两个 \r\n 转义字符。

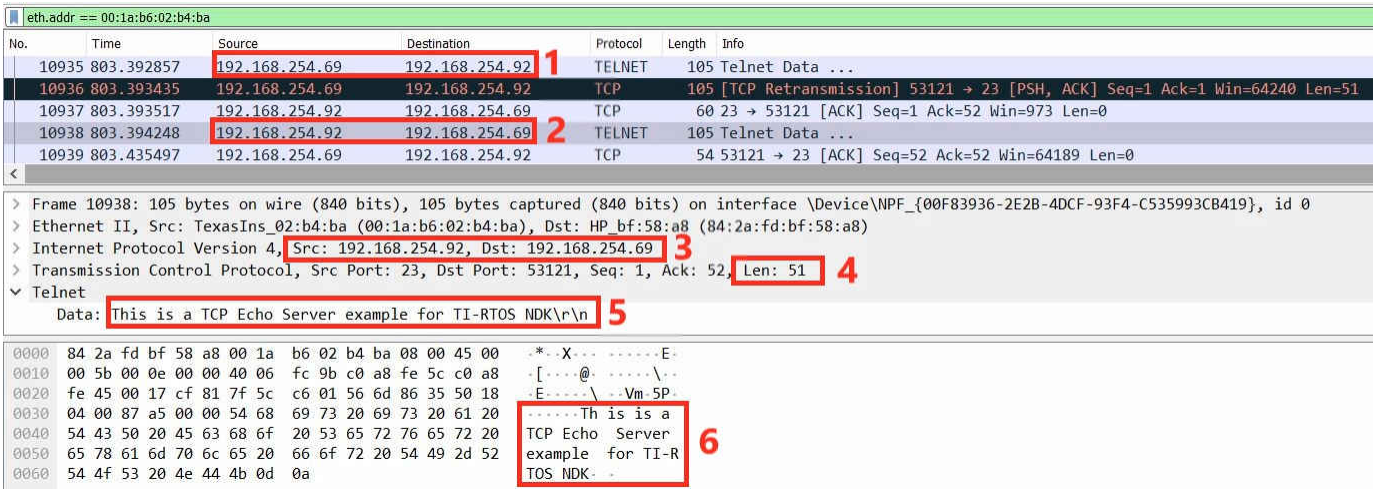


图 6-7. Enet_tcpecho_server_tirtos 的服务器到客户端 Wireshark 捕获

7 Enet_udpecho_server_tirtos 示例概述

用户数据报协议 (UDP) 是另一个众所周知的传输层协议。enet_udpecho_server_tirtos 示例演示了使用 UDP 协议在 TM4C129x MCU 上运行的回显服务器应用。UDP 是一种不涉及错误恢复和重新传输的无连接协议。可以将无连接协议比作邮寄信件。您将信件投入邮局信箱中，这封信件最终能否送到收件人手里，以及是否完好无损（下雨或处理不当可能导致信件损坏）都无法保证。

7.1 运行 enet_udpecho_server_tirtos 示例

SocketTest 工具将在 PC 上作为客户端运行。确保 PC 连接到与 EK-TM4C1294XL 具有相同子网掩码的网络。

按照图 7-1 中所示的步骤设置 SocketTest：

1. 转到“Udp”选项卡。
2. 输入 PC 的 IP 地址和端口号 23，然后按下“Start Listening”按钮。服务器 IP 地址应为运行 SocketTest 的 PC 的地址。若要查找 PC 在网络中的 IP 地址，可以使用 Windows 的 ipconfig 命令。打开一个 Windows 命令窗口，在提示符下键入“ipconfig”，然后将能看到分配给 PC 的 IP 地址。请注意，在 SocketTest 中，无论 PC 是实际的服务器还是客户端，服务器地址字段都将是 PC 的地址。SocketTest 只监听指定地址的任何传入数据。
3. 在“IP Address for Client”中输入 MCU 的 IP 地址。分配给 MCU 的 IP 地址显示在 CCS“Console”窗口中。
4. 转到“Message”（消息）字段并输入一些消息，然后点击“Send”（发送）按钮。
5. 查看 SocketTest 中的对话字段。

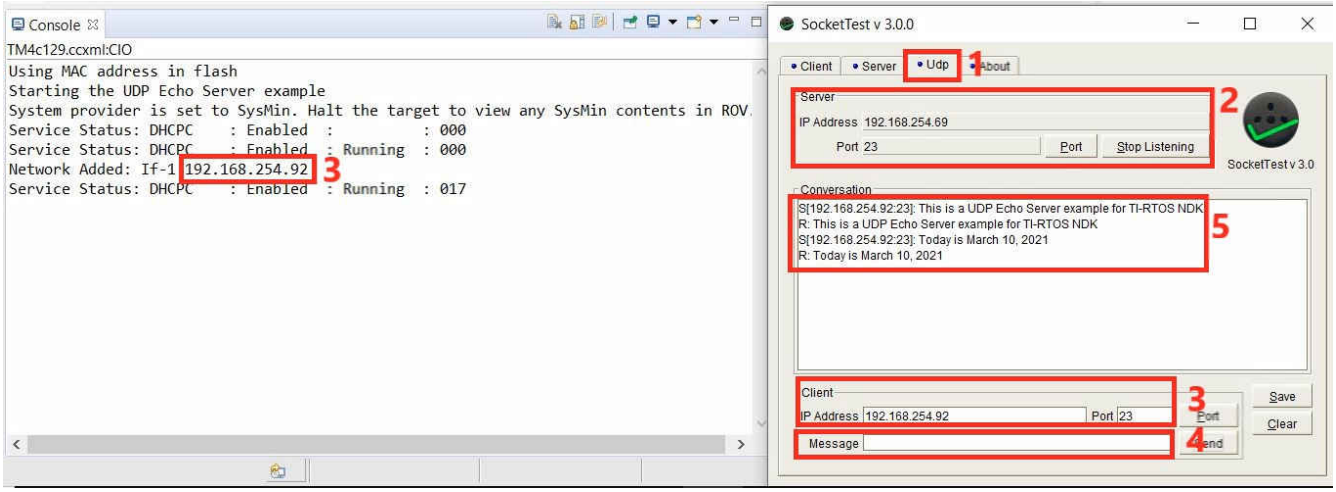


图 7-1. Enet_udpecho_server_tirtos 输出

检查图 7-2 中 Wireshark 捕获的 UDP 流量。

1. 帧号 15351 是从服务器（IP 地址 192.168.254.92）到客户端（IP 地址 192.168.254.69）的第一个回传消息。
2. 长度为 49 字节，如果我们数一下消息“This is a UDP Echo Server example for TI-RTOS NDK\n\r”的长度，就不会感到奇怪了。另外，看一下端口号。PC（客户端）的端口号是 60766，而 MCU（服务器）的端口号是 23。60766 与图 7-2 中输入的内容不同，原因是客户端从未显式选择要绑定到的 UDP 端口。客户端的栈只是选择一个当前可用的 UDP 端口来隐式绑定发送的 UDP 套接字，这个端口每次都可能不同。但是，在 MCU 上运行的 NDK 栈配置为绑定到端口 23。

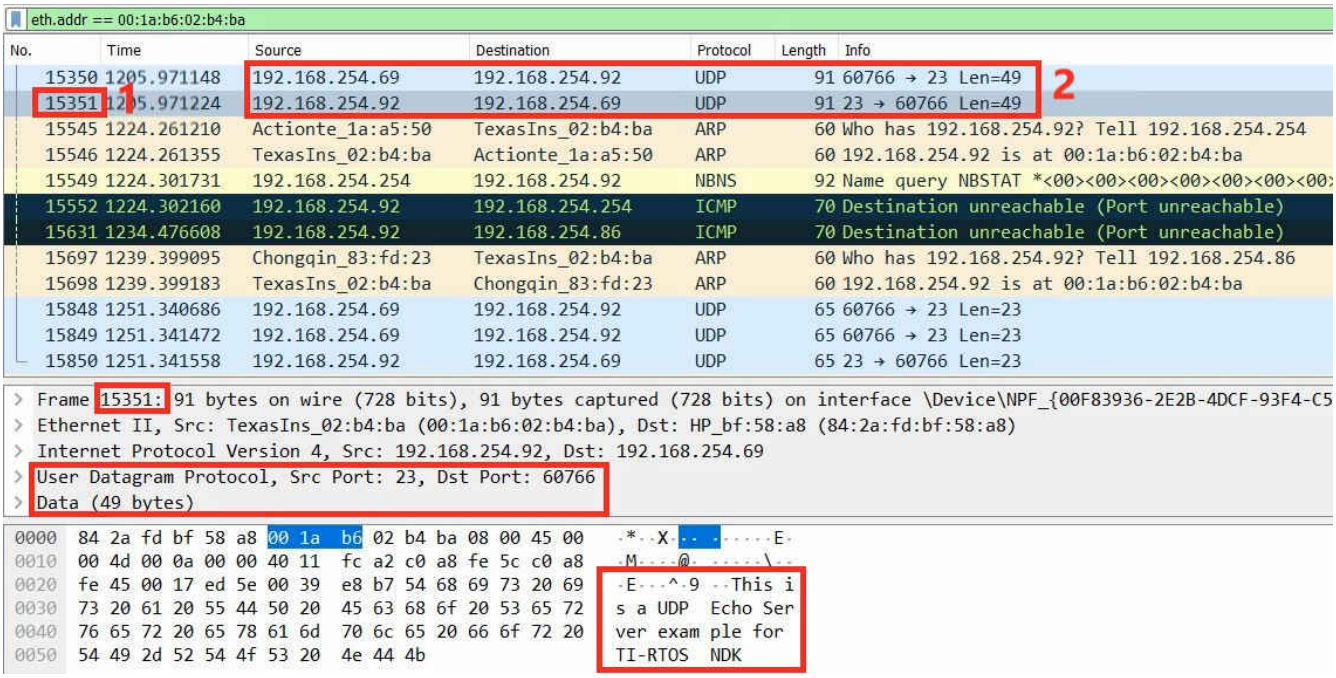


图 7-2. Enet_udpecho_server_tirtos 的服务器到客户端 Wireshark 捕获

8 Enet_httpServer_tirtos 示例概述

此示例将 EK-TM4C1294XL LaunchPad 作为网络服务器进行演示。EK-TM4C1294XL 作为网络服务器接收传入的网络 HTTP 请求，通过 TCP/IP 连接将传出的 HTTP 响应与网络内容一同发送。

HTTP 服务器从 NDK 软件包操作系统适配层内包含的嵌入式文件系统 (EFS) 中提取文件。在此示例中，这些文件被编译到位于基于内存的文件系统上的应用中。

此示例还演示了如何使用通用网关接口 (CGI) 生成动态内容。CGI 是一种接口规范，它使网络服务器能够通过执行程序来处理用户请求。CGI 程序在网络服务器上执行并处理来自用户的输入。CGI 程序是从单个 C 可调用入口点构建的。每个 CGI 函数都在自有独立任务线程上调用。

更多详细信息，请参阅使用 HTTP 服务器进行网络编程。

8.1 为 HTTP 应用配置 NDK

有关如何使用 XGCONF 调出 *.cfg 文件进行配置的更多信息，请参阅如何为 TI-RTOS NDK 创建以太网应用。此示例已包含为 HTTP 应用预配置的 httpServer.cfg。如果从空的 *.cfg 开始，请按照下面的说明为 HTTP 应用配置 NDK。

1. 将 HTTP 服务器模块添加到配置。

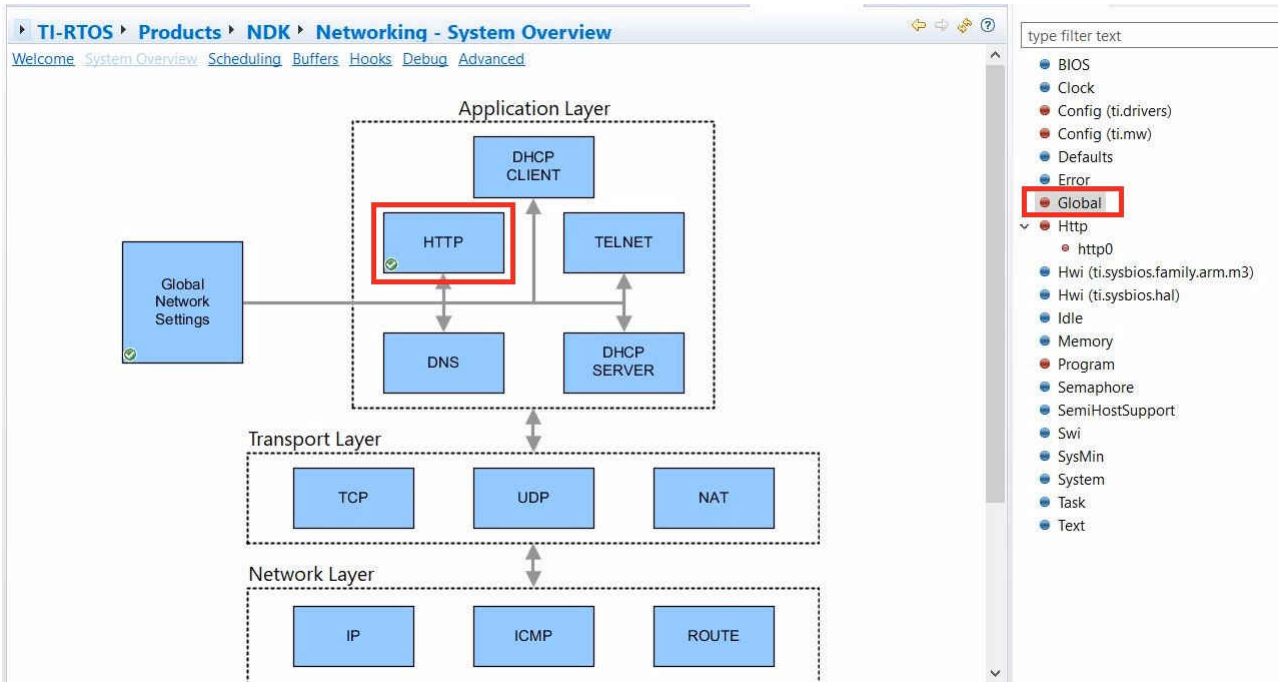


图 8-1. HTTP 应用的 NDK 配置

2. 添加 HTTP 实例。

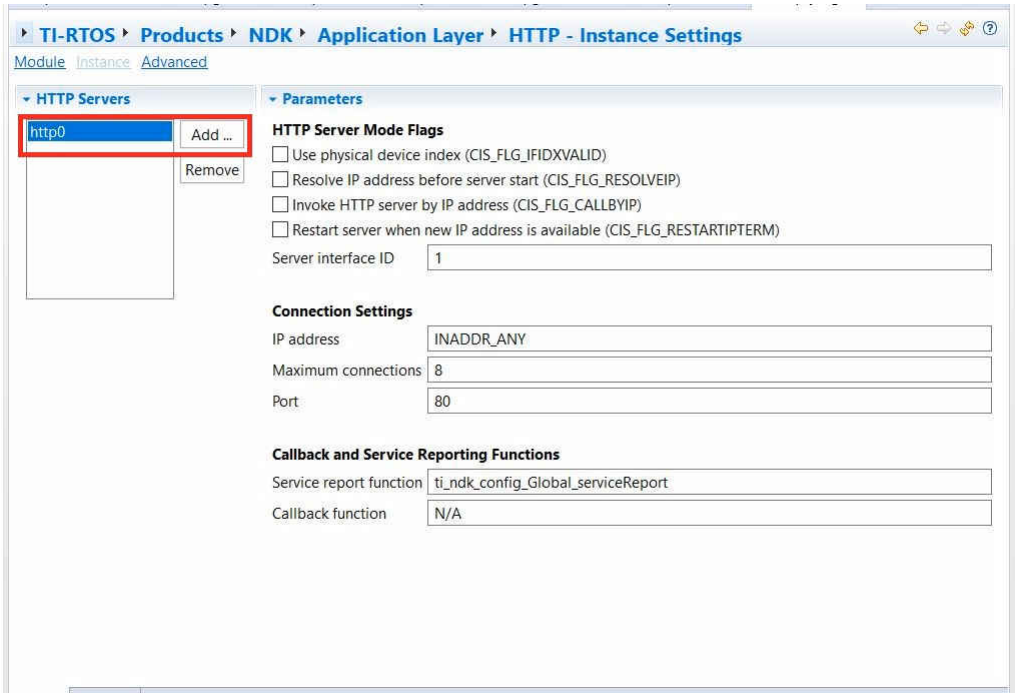


图 8-2. 添加一个 HTTP 实例

3. 默认堆太小。选择 BIOS 页面并导航到“Runtime”页面，将堆大小增加到 22528。HTTP 服务器在其自有任务中运行。它需要一个栈（默认为 2048 字节）。每个连接的客户端都将有一个套接字，因此默认情况下还有 4096 个字节（Rx 和 Tx 缓冲区都有 2048 个字节）。有关 Networking Stack 的内存使用和自定义方式的更多详细信息，请参阅 https://e2e.ti.com/support/processors/f/processors-forum/947313/faq-how-is-memory-managed-in-the-ndk。

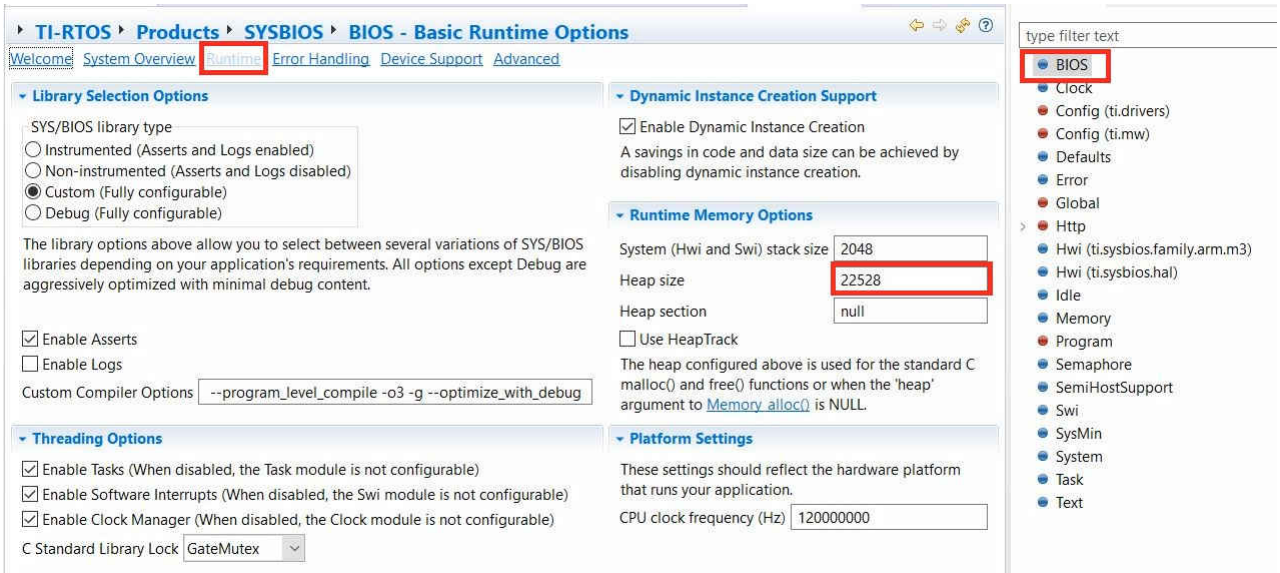


图 8-3. 配置堆大小

8.2 嵌入式文件系统 (EFS) 操作

HTTP Server 服务提供了一种向远程 HTTP 客户端应用提供 HTTP 内容的机制。它使用操作系统适配层中包含的嵌入式文件系统 (EFS) API。EFS 编程 API 中的这些函数包含前缀 efs_。该 API 默认安装基于 RAM 的文件系统，该系统使 HTTP 服务器能够在系统中包含的任何文件存储器件上运行。

8.3 添加 HTTP 服务器内容

本示例的所有 html 页面都存放在工程的“fs”目录下，请参阅图 8-4。这些 html 文件构成了网络服务器要提供的网页。

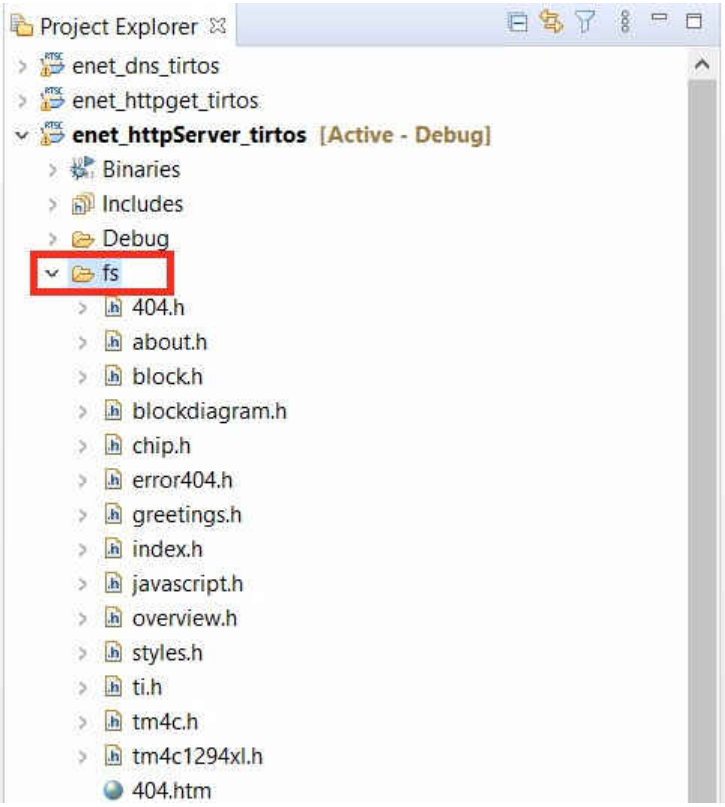


图 8-4. 文件系统目录

html 网页必须先从其二进制 HTML 文件转换为用 C 语言声明的数据数组。NDK 包中提供了一个 MS-DOS 实用程序“binsrc”，用于支持将文件转换为 C 数组。binsrc 实用程序可以在 <TI-RTOS Installation>\products\ndk_2_25_00_09\packages\ti\ndk\tools\binsrc\binsrc.exe 中找到。

binsrc 的调用格式为：

```
>binsrc <input file name> <output file name> <identifier>
```

参数：

- 输入文件名：要转换的文件
- 输出文件名：包含输入文件名 C 数据表示形式的文件的名称
- 标识符：数据的 C 名称

例如，若要转换 HTML 文件 index.html 以供 EFS 使用，可以从 Windows 命令窗口执行以下命令：

```
>binsrc index.html index.h INDEX
```

转换为 index.h 后的 index.html 将包含以下内容，如图 8-5 所示。

1. index.h 文件将文件的大小定义为 1932 字节。稍后我们将查看 Wireshark，以确认网络服务器确实会将其 HTTP 响应中的相同字节数传输给客户端。
2. INDEX 字符数组以 <!DOCTYPE HTML> 开头。打开文本编辑器读取 index.html 的第一行，可以确认它与存储在 INDEX 数组中的内容相同。
3. 查看 INDEX 数组中接下来几个字节的内容，它对 <!-Copyright © 2013-2021 Texas Instruments Incorporated.All rights…..> 进行编码。再次将其与 index.html 进行比较，并确认 index.h 是 index.html 的二进制表示形式。

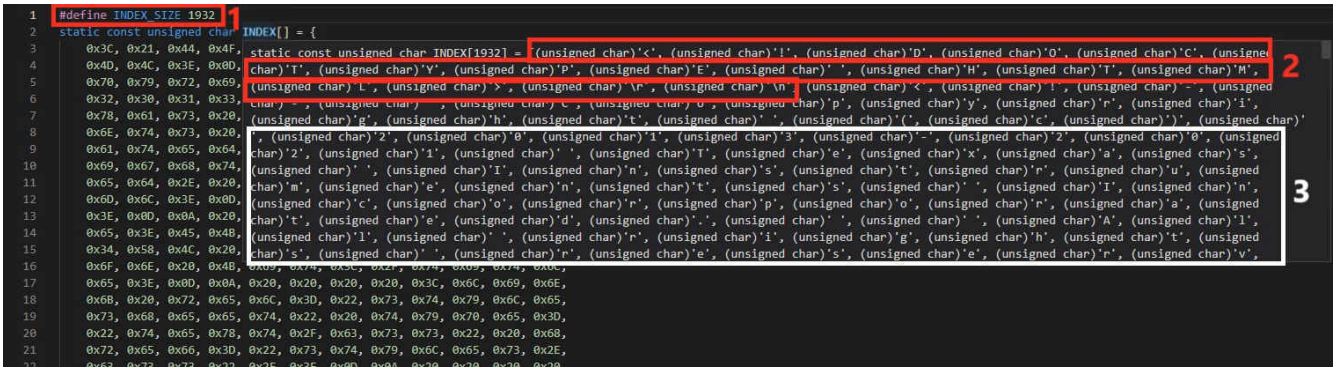


图 8-5. Index.html 二进制文件内容

8.4 向 EFS 声明 HTML 文件

只要 HTML 文件转换为内存映像，就会通过调用函数 *efs_createfile()* 将该文件声明到 EFS 文件系统。在初始化期间，在实际调用 HTTP 服务器之前，通常会同时创建所有 HTML 文件。在 enet_httpServer_tirtos 示例代码中，使用了 *AddWebFiles()* 和 *RemoveWebFiles()* 两个函数，这两个函数包含初始化和清理 EFS 文件环境所需的所有代码。

下面是向 EFS 声明 HTML 文件的示例代码片段。

```
/* 文件系统头文件 */
#include "fs/index.h"
#include "fs/about.h"
#include "fs/overview.h"
Snip…

Int getTime(SOCKET s, int length)
{
    Char buf[200];
    static UInt scalar = 0;
Snip…

Void AddWebFiles(Void)
{
    efs_createfile("index.html", INDEX_SIZE, (UINT8 *)INDEX);
    efs_createfile("overview.htm", OVERVIEW_SIZE, (UINT8 *)OVERVIEW);
    efs_createfile("about.htm", ABOUT_SIZE, (UINT8 *)ABOUT);
Snip…
    efs_createfile("getTime.cgi", 0, (UINT8 *)&getTime);
Snip…Void RemoveWebFiles(Void)
{
    efs_destroyfile("index.html");
    efs_destroyfile("overview.htm");
    efs_destroyfile("about.htm");
Snip…
    efs_destroyfile("getTime.cgi");
Snip…
```

图 8-6 显示了在 NDK 配置中对两个挂钩函数 *AddWebFiles()* 和 *RemoveWebFiles()* 进行声明的位置。

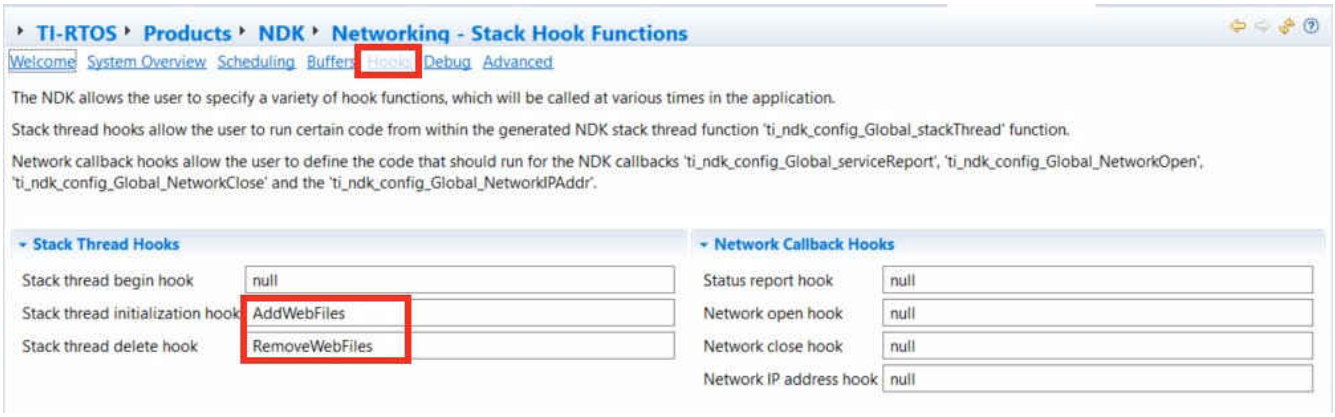


图 8-6. HTTP 挂钩声明

只要运行上述代码，EFS 系统就准备好让 HTTP 服务器提供内容。

8.5 编写 CGI 函数

CGI 程序必须在 EFS 中，HTTP 服务器才能看到它们。请参阅上面的示例代码，其中文件 getTime.cgi 的入口转换为 C 函数 *getTime()*。只要对文件 getTime.cgi 进行 HTTP POST 操作，就会调用 *getTime()* 函数。

如需编写 CFG 函数的详细信息，请参阅编写 CGI 函数。

8.6 运行 enet_httpServer_tirtos 示例

如果您要创建自有网络内容或加载并运行现有示例，请重新编译示例。在 CCS“Console”窗口中，应该会看到显示的网络服务器 IP 地址。如需更多信息，请参阅图 6-5。

网络服务器运行后，在浏览器的 URL 字段中输入 IP 地址，如图 8-7 所示，示例网页将呈现在浏览器中。如果没有指定其他页面，index.html 通常是网页上显示的默认页面。换句话说，index.html 是网站主页所用的名称。

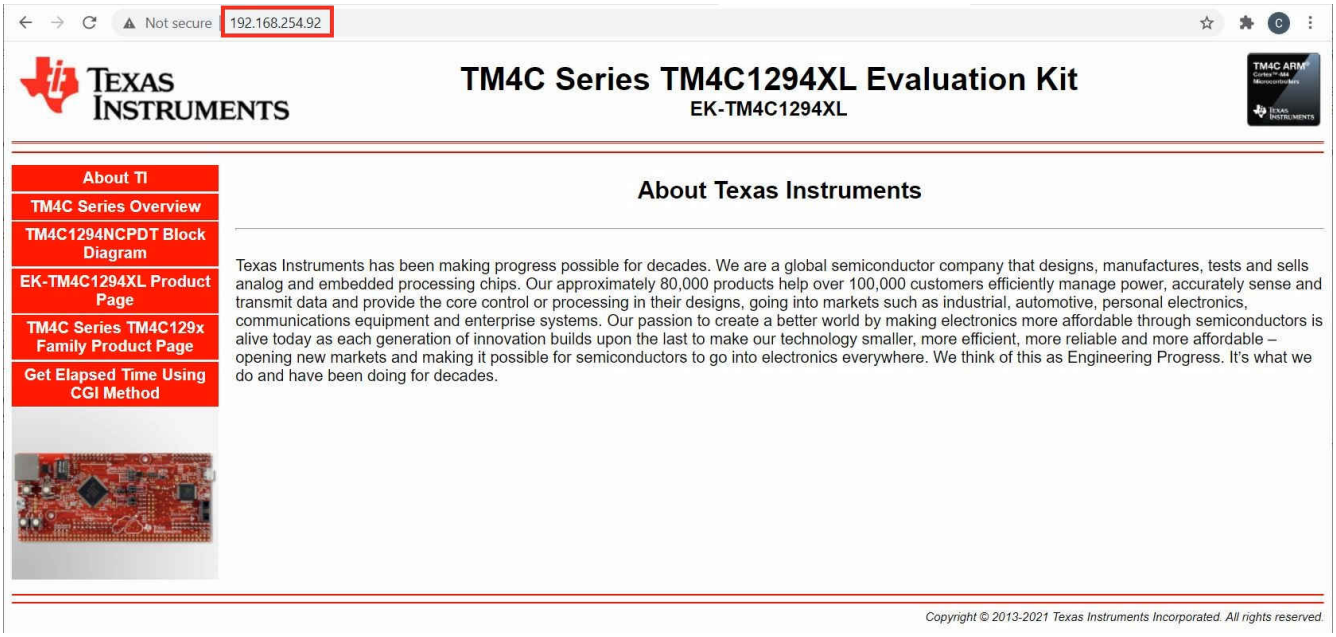


图 8-7. Enet_httpServer_tirtos 网络服务器主页

主页上显示若干超链接和图像，点击其中任何一个即可查看出现的内容。例如，点击“TM4C1294NCPDT Block Diagram”，将看到如图 8-8 所示的网页。请注意，通过在 URL 上直接输入“192.168.254.92/block.htm”也可以访问该网页。

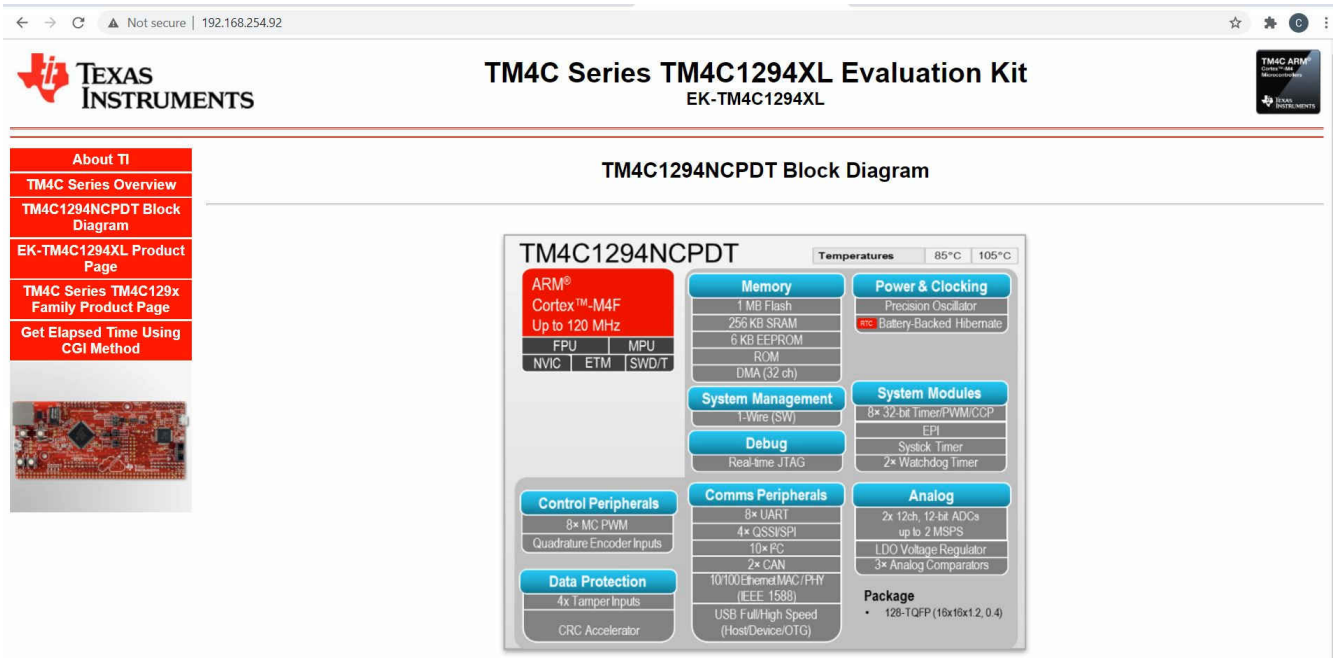


图 8-8. 方框图页面

检查图 8-9 中的 Wireshark 捕获。

1. 来自客户端的第一个 HTTP GET 请求发生在第 4460 帧，相应的服务器响应发生在第 4468 帧。服务器以 OK 响应以及默认的 index.html 网页内容进行响应。
2. 文件大小为 1932 字节。这与 index.h 文件中指定的 INDEX 数组完全相同，如图 8-5 所示。
3. 共有 54 行 html 数据，第一行为 <!DOCTYPE HTML>\r\n。这与图 8-5 相符。
4. 同样，index.html 的第二行是 <!—Copyright (c) ……> ，这与 INDEX 字符数组中的编码相同。

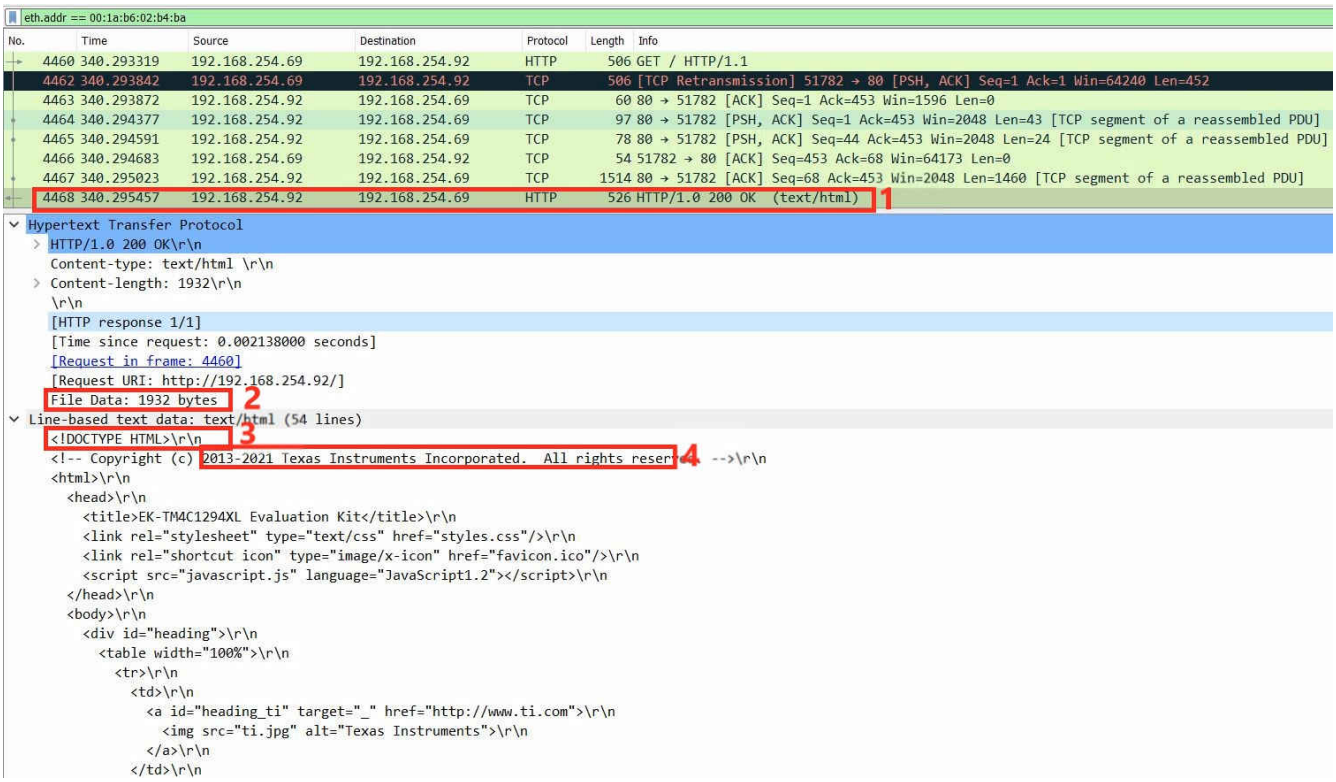


图 8-9. enet_httpServer_tirtos 网络服务器的 Wireshark 捕获

最后，在主页上，点击“Get Elapsed Time Using CGI Method”。这一操作会触发 *getTime()* 函数来执行并输出网络服务器的运行时长，请参阅图 8-10。按下浏览器上的“Refresh”按钮可刷新屏幕以查看更新的时长。请注意，也可以通过在 URL 中直接指定 CGI 函数来将其触发，如“192.168.254.92/getTime.cgi”中所示。

图 8-10. 网络服务器运行时长

9 Enet_dns_tirtos 示例概述

连接到网络的每个设备都将具有唯一的 IP 地址，该地址依赖于 IP（互联网协议）进行通信。但是，IP 地址要么是 IPv4 32 位地址，要么是 IPv6 128 位地址，很难让人记住。DNS（域名系统）是一种应用层服务，可将“人性化”的域名转换为 IP 地址。人类很容易记住 www.ti.com 而不是其数字 IP 地址。请注意，DNS 是依赖于 UDP 协议的应用服务，它本身并非是一种协议。

这个简单的示例演示如何使用 DNS 功能来检索四个不同网站的 IP 地址。

9.1 如何为 DNS 配置 NDK

若要使用 DNS，必须在 NDK 中启用 DNS 模块。因为 DNS 以 UDP 作为传输协议，所以也需要启用 UDP 模块。

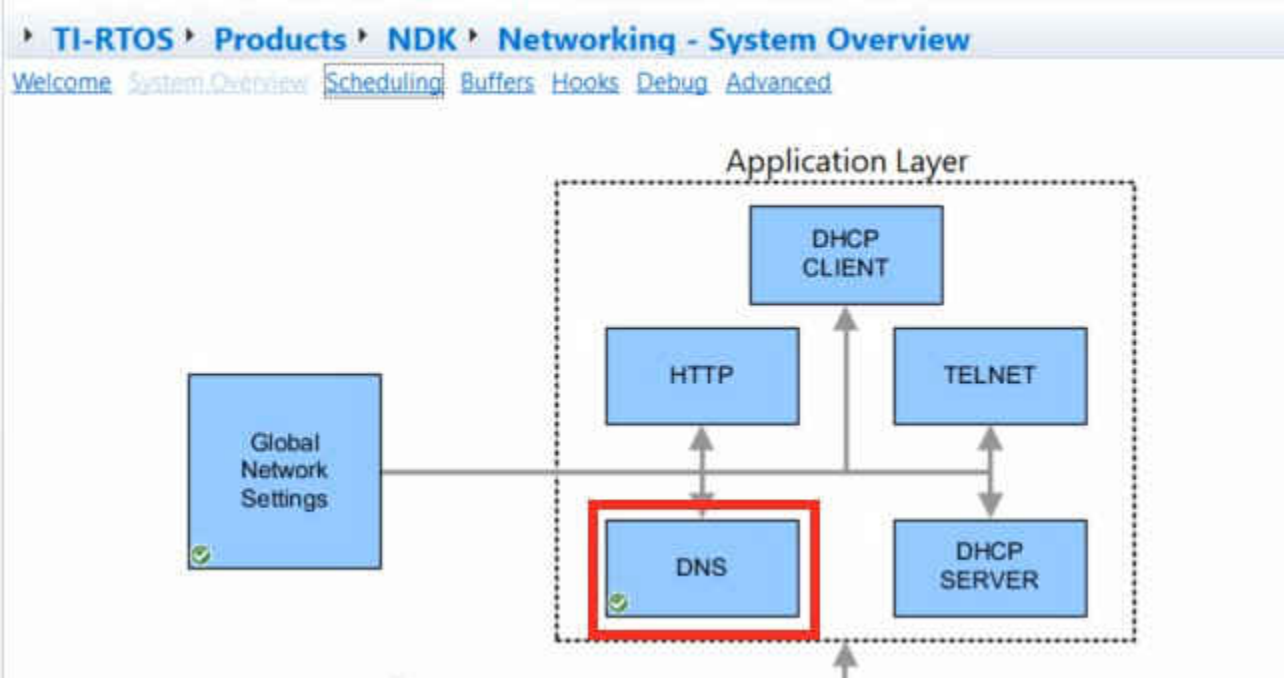


图 9-1. 为 DNS 配置 NDK

9.2 如何在 Wireshark 上查看 DNS 流量

如果将 MCU 器件和 PC 连接到以太网交换机，则将无法在 Wireshark 上查看 DNS 流量。原因是 DNS 流量位于 DNS 服务器和器件之间。交换机不会将流量路由到运行 Wireshark 的 PC 上。若想查看 DNS 流量，您要么必须有一个以太网集线器，以将网络上的所有流量传播到连接它的所有端口，要么需要将以太网交换机配置为“端口镜像”。如今，很难找到以太网集线器这种设备。容易找到的是智能交换机，可以将其配置为“端口镜像”。图 9-2 显示了在何处将连接交换机端口 2 的 EK-TM4C1294XL 镜像到连接 PC 的端口 1。利用端口镜像，进出端口 2 的所有流量都将被端口 1 上的 Wireshark 监视。请查阅交换机或路由器的数据表，了解如何配置端口镜像。

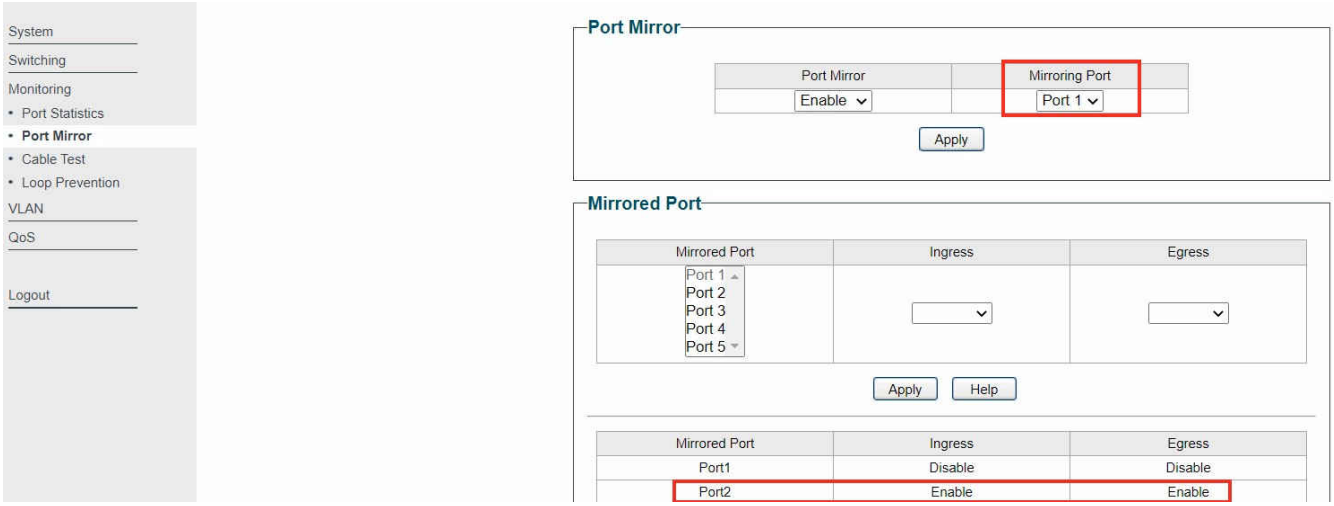


图 9-2. 端口镜像

9.3 运行 enet_dns_tirtos 示例

如预期的那样，运行本示例会在 CCS“Console”窗口上输出相应网站的四个 IP 地址，请参阅图 9-3。更多详细信息，请查看 Wireshark 捕获，特别是关于 www.google.com 的部分。

- 在图 9-4 的 Wireshark 捕获中，从突出显示的框 1 和框 2，可以看到与四个网站对应的四个 DNS 数据包。
- 事务的源地址来自 IP 地址 192.168.254.254。这恰好是此应用所连接的路由器的 IP 地址。智能路由器通常用作 DNS 服务器，将过去访问过的域名存储在其缓存中以便快速检索。
- 在框 3 中，www.google.com 返回的 IP 地址是 172.217.1.132。您可以通过在浏览器的 URL 字段中输入此地址进行确认，它应该会将您引导至 Google 网站。
- 在框 4 中，IP 地址表示为等于 0xACD90684 的 32 位二进制值。0xAC 等于十进制数 172，0xD9 等于十进制数 217，其余部分也是如此。

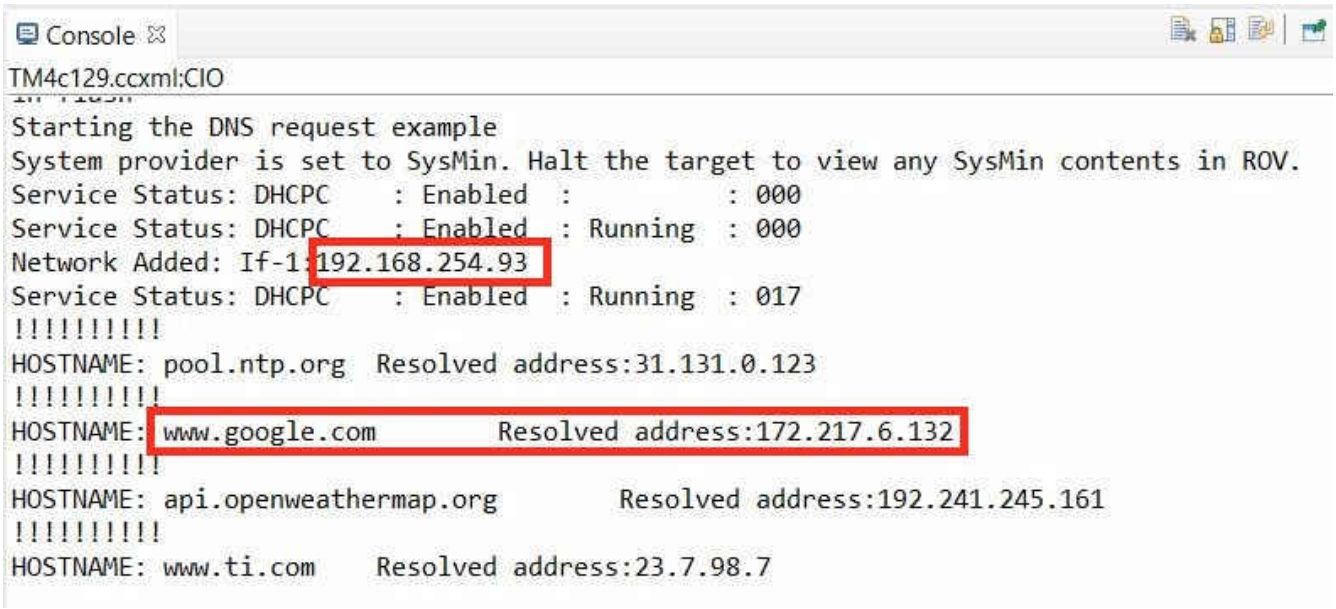


图 9-3. Enet_dns_tirtos 输出

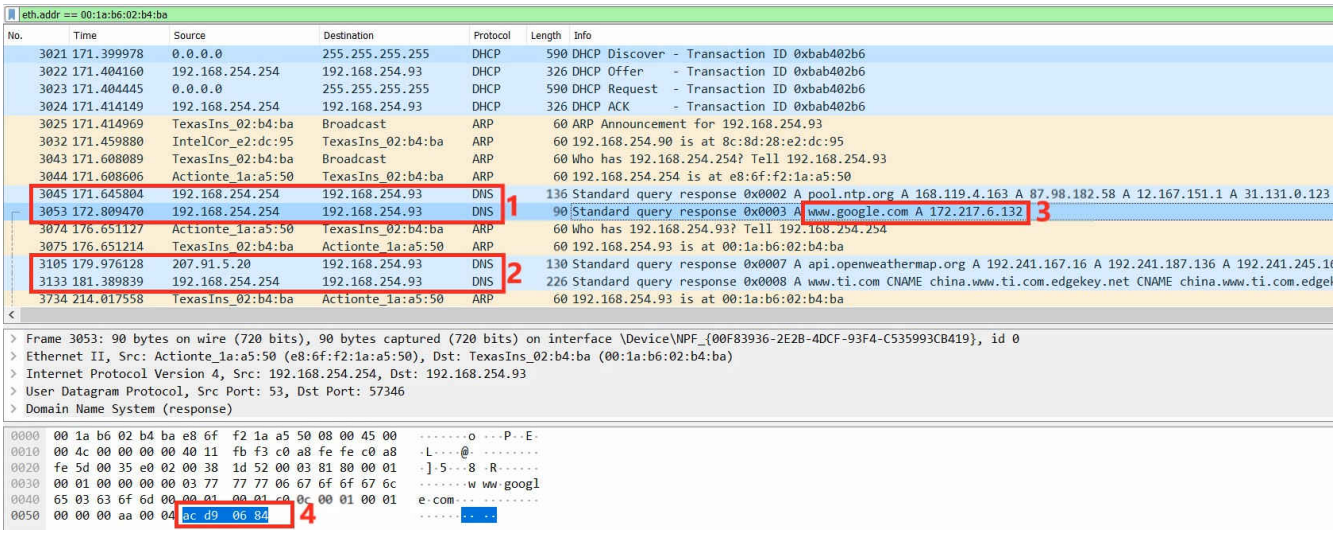


图 9-4. Enet_dns_tirtos 的 Wireshark 捕获

10 Enet_sntp_tirtos 示例概述

简单网络时间协议 (SNTP) 是 NTP 的简化版本。SNTP 通常提供精确到 100ms 以内的时间，无需复杂的 NTP 过滤机制。SNTP 是一种应用层协议，基于 UDP 作为传输层。

本示例使用 SNTP 协议获取网络时间，并在 CCS“Console”窗口上显示针对北美中部时间 (CT) 区调整的当前时间。

10.1 运行 enet_dns_tirtos 示例

客户端将请求发送到 pool.ntp.org 网站，从中随机选择一个公共时间服务器来提供网络时间，如图 10-2 中 Wireshark 的捕获信息所示。获取的网络时间先针对中央时区进行调整，然后显示在 CCS“Console”窗口中，如图 10-1 所示。

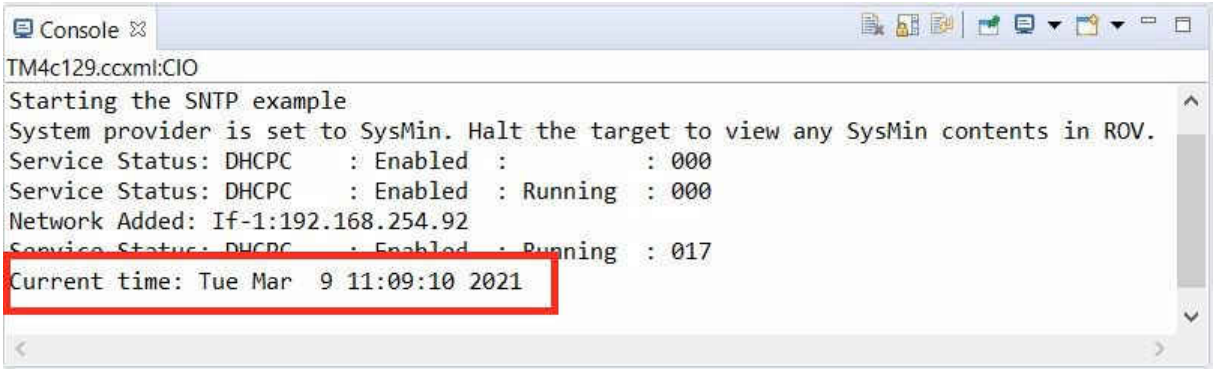


图 10-1. Enet_sntp_tirtos 输出

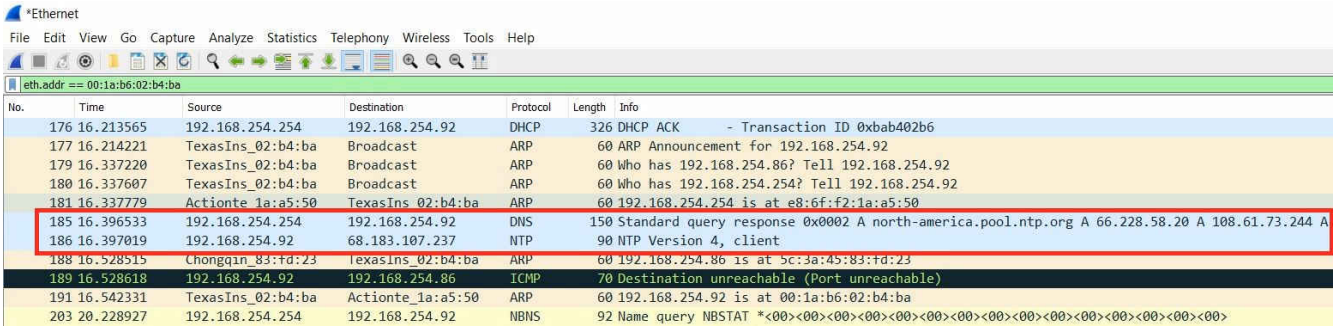


图 10-2. Enet_sntp_tirtos 的 Wireshark 捕获

11 Enet_tcpecho_client_tirtos 示例概述

enet_tcpecho_client_tirtos 示例演示了一个客户端应用，该应用首先连接到服务器并发出问候消息“Hello from TM4C1294XL Connected LaunchPad\n”，然后回显它从服务器接收到的任何内容。

如图 1-3 中 TCP 的 BSD 套接字流程图所示，客户端将使用 connect() 连接到指定的服务器地址和端口。建立连接后，客户端将使用 recv() 从服务器接收数据，然后回显数据。

流程图中客户端和服务器之间的另一个区别是客户端不需要调用 bind()。TCP 的客户端通常不需要绑定。某些情况下可能需要绑定客户端，这时可使用 bind() 来绑定客户端。一个例子是客户端上的防火墙只允许实现某个端口上的传出连接。

11.1 配置服务器 IP 地址

若要运行此示例，必须在编译期间知道服务器 IP 地址和端口号，以便客户端与服务器建立连接。

- 在 tcpEchoClient.c 文件中定义 SERVER_IPADDR 和 SERVER_PORT。请注意，下面的地址只是一个示例。必须将服务器 IP 地址更改为您在网络中连接到的服务器 IP 地址，否则该示例将不起作用。

```
#define SERVER_IPADDR "192.168.254.69"
#define SERVER_PORT 23
```

重新编译工程，然后您就可以运行示例了。

11.2 配置 SocketTest 服务器

因为这是一个客户端应用，所以必须首先设置服务器，且服务器必须处于监听状态，这样才能连接客户端。

按照图 11-1 中所示的步骤设置 SocketTest 服务器：

1. 打开 SocketTest 中的“Server”选项卡。
2. 输入服务器 IP 地址以及端口号。IP 地址是运行 SocketTest 的 PC 之一。IP 和端口号需要与 tcpEchoClient.c 中定义的设置匹配，如节 11.1 所示。最后，按下“Start Listening”按钮。客户端连接后，等待问候消息出现在对话窗口中。

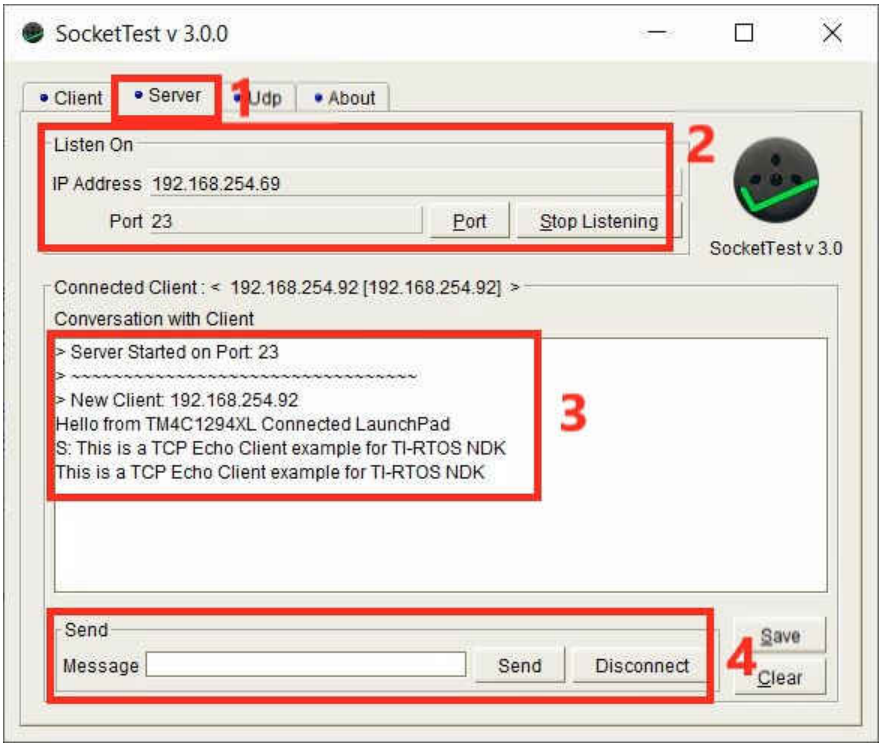


图 11-1. Enet_tcpecho_client_tirtos 的 SocketTest 服务器配置

11.3 运行 enet_tcpecho_client_tirtos 示例

示例运行后，它将在 CCS“Console”窗口上显示从 DHCP 服务器获取的客户端 IP 地址。客户端连接到 SocketTest 服务器后，它向服务器发送问候消息“Hello from TM4C1294XL Connected LaunchPad\n”。

查看图 11-1 框 3 中的对话字段，服务器接受连接后，该字段立即显示来自客户端 IP 地址 192.168.254.92 的消息“Hello from TM4C1294XL Connected LaunchPad\n”。转到框 4 中所示的 SocketTest 中的 Message 字段并键入一些消息。无论输入何种消息，客户端都会回传。在本例中，输入的消息是“This is a TCP Echo Client example for TI-RTOS NDK\n\r”，总长度为 53 个字符。

检查图 11-2 中的 Wireshark 捕获。

1. 如前所述，TCP 是基于连接的协议。在这里，您会看到客户端 192.168.254.92 发送到服务器 192.168.254.69 以建立连接的 SYN 段，以及来自服务器的用来接受连接的 ACK 段。
2. 连接被接受后，客户端发送消息“Hello from TM4C1294XL Connected LaunchPad\n”。这在第 129 帧中显示为 Telnet Data。
3. 请注意，消息的长度为 42 字节，与问候消息中的总字符数相匹配。
4. 捕获的消息内容与客户端发送的问候语相匹配。

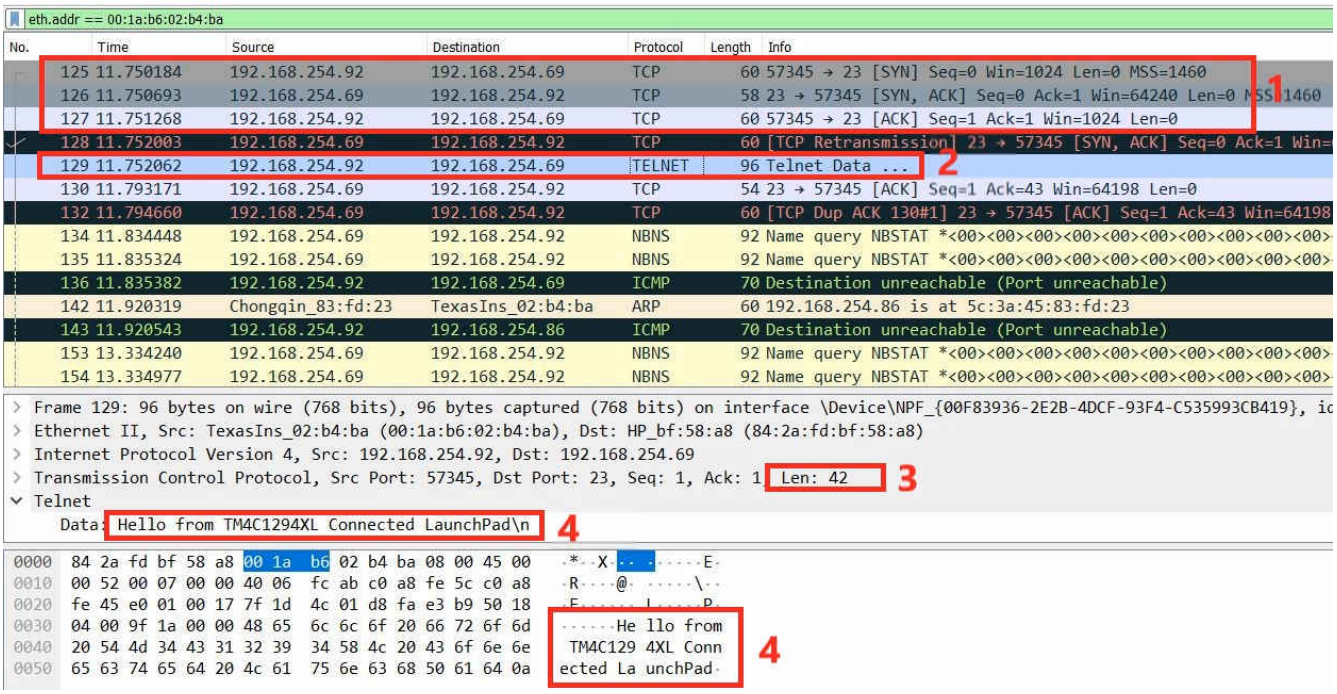


图 11-2. Enet_tcpecho_client_tirtos 的客户端服务器 Wireshark 捕获

12 Enet_udpecho_client_tirtos 示例概述

enet_udpecho_client_tirtos 示例是使用 UDP 协议的客户端应用。就 API 而言，它与 UDP 流程图中描述的服务器类似。此示例使用 sendto() 向远程服务器发送问候消息“Hello from TM4C1294XL Connected LaunchPad\n\r”。然后，它将回传从服务器接收的任何内容。

12.1 运行 enet_udpecho_client_tirtos 示例

有关如何设置 SocketTest 以进行 UDP 测试的信息，请参阅节 7.1。图 12-1 中的 SocketTest 首先显示客户端发送的问候消息“Hello from TM4C1294XL Connected LaunchPad\n\r”。然后服务器发送消息“This is a UDP Echo Client example for TI-RTOS NDK\n\r”。客户端接收该消息，然后回传给服务器。图 12-2 中 Wireshark 捕获的消息可以交互参考。

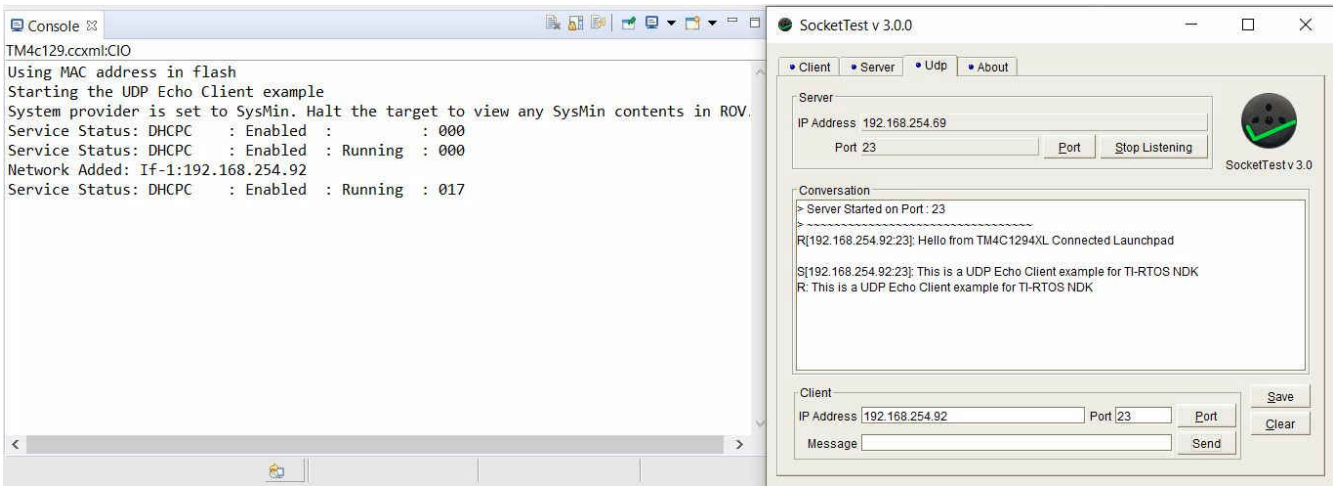


图 12-1. Enet_udpecho_client_tirtos 输出

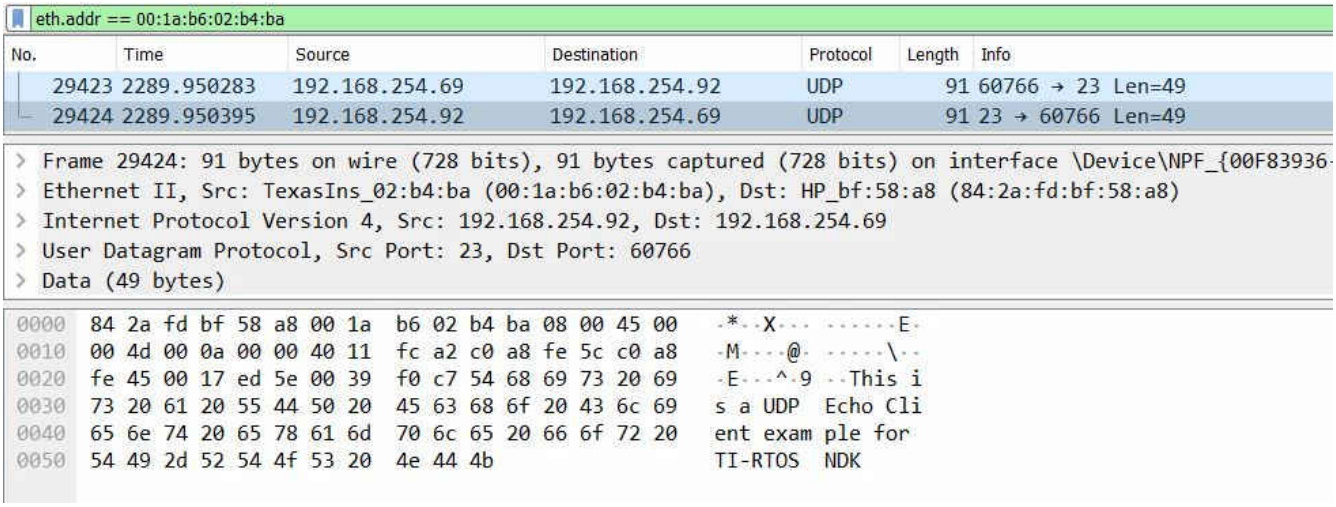


图 12-2. Enet_udpecho_client_tirtos 的客户端服务器 Wireshark 捕获

13 Enet_httpget_tirtos 示例概述

enet_httpget_tirtos 示例使用请求 URI 对 www.example.com 进行 HTTP GET 调用。通过处理来自 HTTP 服务器的响应状态代码、标头字段和正文，获取响应状态和数据。HTTP 响应状态和接收到的数据字节数列印在 CCS “Console”窗口上。

13.1 如何为 HTTP GET 配置 NDK 示例

HTTP 应用以 TCP 作为底层传输协议。此示例唯一需要的 NDK 模块就是 TCP 模块。如果要从空的 .cfg 工程开始，请启用 TCP 模块。

还应注意，客户端在建立连接和向服务器发出 HTTP 请求之前，必须首先获取其 IP 地址。如果客户端在没有 IP 地址的情况下尝试连接服务器，则地址解析协议 (ARP) 将失败。ARP 是一种通信协议，用于发现与给定 IP 地址相关联的链路层地址，例如 MAC 地址。本例中的 NDK 配置确保只有在获取 IP 地址后才会调用回调挂钩函数。*netIPAddrHook()* 关联为 NDK 网络 IP 地址回调函数 *NetworkIPAddr()* 的用户定义挂钩函数，该函数在从系统添加 IP 地址时调用，请参阅图 13-1。当 *netIPAddrHook()* 函数被调用时，它启动 HTTP 请求任务。

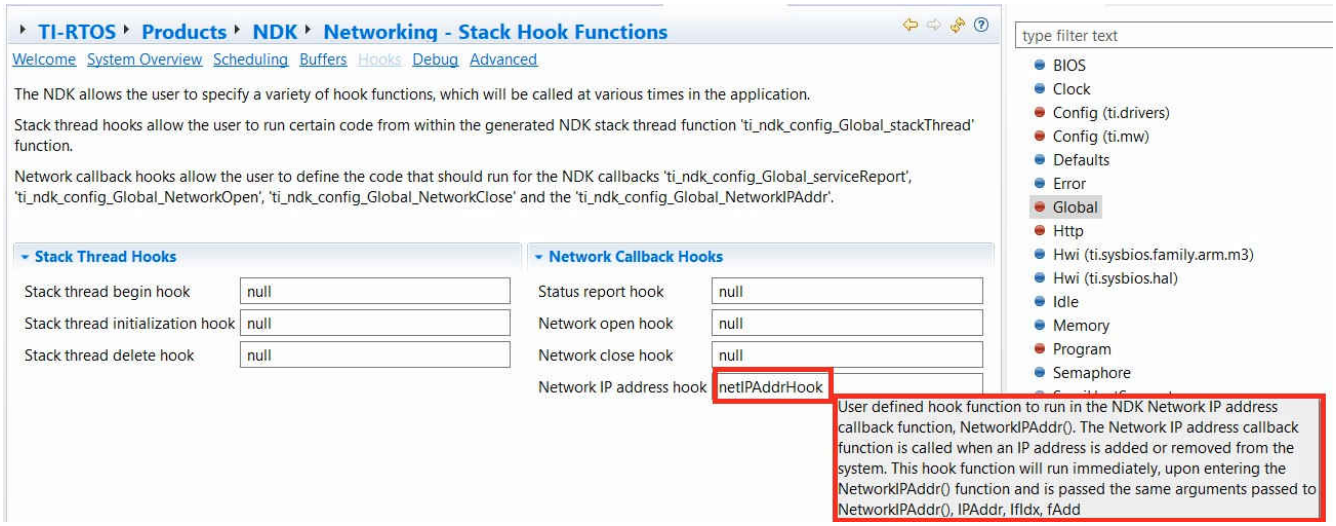
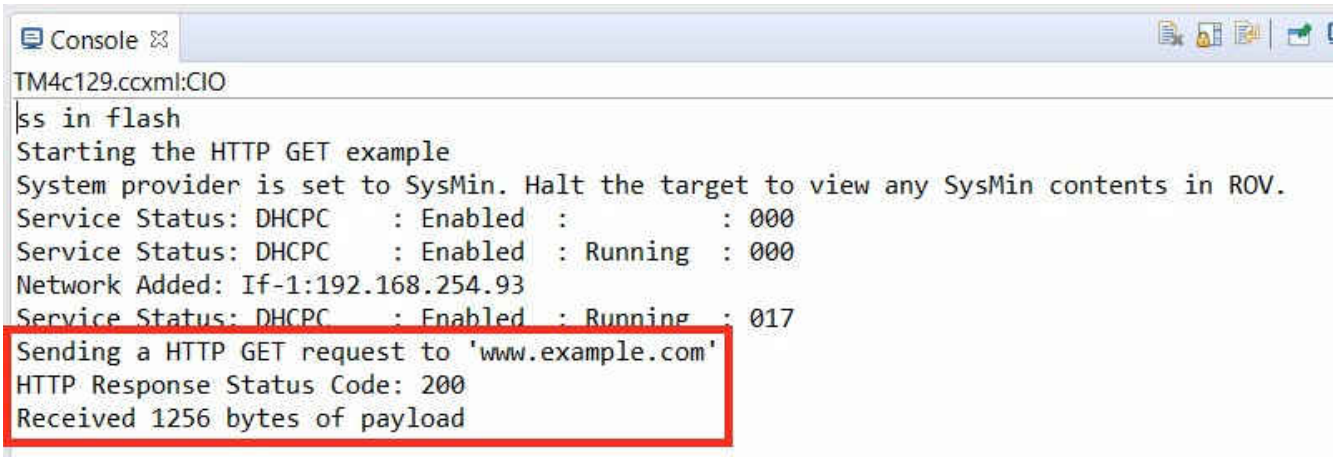


图 13-1. 网络 IP 地址挂钩函数

13.2 运行 enet_httpget_tirtos 示例

图 13-2. Enet_httpget_tirtos 输出



14 参考文献

- 德州仪器 (TI)：*TI Network Developer’s Kit (NDK) 用户指南*
- 德州仪器 (TI)：*TI Network Developer’s Kit (NDK) API 参考*
- TI-RTOS-MCU 概述
- 德州仪器 (TI)：*适用于 TivaC 的 TI-RTOS 入门指南*
- 德州仪器 (TI)：*TI-RTOS 用户指南*
- 德州仪器 (TI)：*SYS/BIOS（TI-RTOS 内核）用户指南*

重要声明和免责声明

TI 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安全、安保或其他要求。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 (https:www.ti.com/legal/termsofsale.html) 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。

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

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