Remote Configuration Over Ethernet with the Nios II Processor

© April 2010 AN-429-3.0

Introduction

Firmware in embedded hardware systems is frequently updated over the Ethernet. For embedded systems that comprise a discrete microprocessor and the devices it controls, the *firmware* is the software image run by the microprocessor. When the embedded system includes an FPGA, firmware updates include updates of the hardware image on the FPGA. If the FPGA includes a Nios® II soft processor, you can upgrade both the Nios II processor—as part of the FPGA image—and the software that the Nios II processor runs, in a single remote configuration session.

This application note presents a methodology for implementing remote configuration in Nios II-based systems. The Trivial File Transfer Protocol (TFTP) is used to upload images—software, hardware, or binary data—to the system and to trigger reconfiguration. The web server provided with the Nios II Embedded Development System (EDS) and the web server that accompanies the Nios Embedded Evaluation Kit (NEEK) provide additional remote configuration functionality. Providing yet another web-based method to upload images and trigger reconfiguration, the Embedded Systems Development Kit, Cyclone® III Edition (EDK) includes software that enables access to a Board Update Portal (BUP).

This application note contains the following sections:

Assumptions and Requirements

This document targets the systems engineer who wants to design a system for remote configuration. It provides a simple application that implements basic remote configuration features for the Nios II system.

The following sections describe the knowledge and experience assumed for the instructions in this application note, and the hardware and software required to run the software example.

Knowledge Requirements

This discussion of remote configuration assumes that you have the indicated level of knowledge or experience in the following areas:

- Knowledge of network programming, preferably sockets-based programming. Knowledge of the TFTP protocol is not required; this application note provides a brief description of the protocol as implemented in the remote configuration example.

- Working knowledge of Nios II systems and the tools to build them. These systems and tools include the Quartus® II software, SOPC Builder, and the Nios II EDS.
- Knowledge of Altera® configuration methodologies and tools, such as the Cyclone III Remote Update Controller component for SOPC Builder and the Max® II parallel flash loader (PFL) feature.
- Experience with the Nios II flash programmer using the GUI in the Nios II Software Build Tools (SBT) for Eclipse or using the command-line interface.
- Knowledge of the Motorola S–Record format, which is used for all flash programming and remote configuration images.

For information about remote configuration of an FPGA over an Ethernet connection, refer to "Upgrade Hardware Image via an Ethernet Link" in *AN 346: Using the Nios II Configuration Controller Reference Designs*. For general information about how to configure the FPGA on a development board, refer to *AN 346: Using the Nios II Configuration Controller Reference Designs* and the configuration chapter in the relevant device handbook.

For more information about the Flash Programmer, refer to the *Nios II Flash Programmer User Guide*.

Hardware and Software Requirements

To run the software example described in this application note, you must have the following hardware and software:

- One of the following two Altera development kits:
  - Nios II Embedded Evaluation Kit, Cyclone III Edition (NEEK)
  - Embedded Systems Development Kit, Cyclone III Edition (EDK)
- A computer with a TFTP client installed. You can find a TFTP client to download using an Internet search. For Windows, multiple TFTP clients are available. For convenience, one client is included with the design files that accompany this application note. For Linux or UNIX, multiple TFTP clients are available online at no charge.

  CAUTION The TFTP client command lines in this document are provided for example purposes only. Your client may require different command lines for proper operation.

- The **an429_remote_configuration.zip** file, which contains the directories and files listed in Table1. Click here to download this **.zip** file.

Table 1. Files and Directories in an429_remote_configuration.zip File (Part 1 of 2)

File or Directory Name	Description
walkthroughs/tftp.exe	TFTP client for Windows. You can save this executable file in your Windows directory.
walkthroughs/remote_config.sh	Bash shell script for remote configuration.

Table 1. Files and Directories in an429_remote_configuration.zip File (Part 2 of 2)

File or Directory Name	Description
src/SBT_flow/remote_config	Software example files for use with the Nios II SBT for Eclipse. To use the Nios II SBT for Eclipse, copy the contents of **remote_config** to **<***Nios II EDS installation directory***>/examples/software**. These files support the remote_update component method of remote configuration ("Method 2—With remote_update Component" on page 7).
src/CLI_flow/rc_iniche	Software example files for use with the command-line flow. To use the command-line flow, copy the contents of **rc_iniche** to the directory for the board reference design (standard or full-featured) that you are using. These files support the remote_update component method of remote configuration ("Method 2—With remote_update Component" on page 7).
walkthroughs/NEEK **walkthroughs/NiosII_CycloneIII_3C120**	These directories contain preprogrammed files to run each example on its targeted development board, including versions of the files in **src/SBT_flow/remote_config** and **src/CLI_flow/rc_iniche**. For the NEEK walkthrough, the files support the remote_update component method of remote configuration ("Method 2—With remote_update Component" on page 7). For the EDK walkthrough, the files are customized to support the Max II parallel flash loader method of remote configuration ("Method 3—With Max II Parallel Flash Loader Feature" on page 7). Instead of the preprogrammed flash memory files, you can substitute your own **.flash** files that contain hardware, software, or binary images, if you set the flash file offsets appropriately in the **remote_config.h** file.
Readme.txt	Description of the accompanying software examples.

If you create your own hardware design, you must ensure that the remote_update component or the Max II parallel flash loader component, as appropriate, is included in your design, and that you have incorporated a working network stack. In the designs for the two **.sof** files provided in the **.zip** file, the Altera Triple Speed Ethernet MegaCore function provides the network function.

High-Level Description

In a Nios II subsystem, remote configuration consists of the following four steps:

1. Preparing the Images
2. Uploading the Images
3. Programming Images to Flash Memory
4. Triggering a Reconfiguration

The following sections describe how to implement these steps.

Preparing the Images

To program the flash memory in your design, you must convert the images you wish to program to the **.flash** format. The software images are automatically created every time you build your project in the Nios II SBT for Eclipse.

Before programming the flash memory device, you create the **.flash** files using one of the following three commands:

- `elf2flash` (software images)
- `bin2flash` (binary images)
- `sof2flash` (FPGA configuration images)

Example 1 shows a sample command line that creates a software flash image.

Example 1. Creating a Software Flash Image

```
elf2flash  --base=<base flash address>
           --end=<end flash address>
           --reset=<Nios II reset address>
           --input=<project>.elf
           --output=<flash name>.flash
           --boot=<path to boot copier srec>
```

The flash file is in the Motorola S-Record format. Addressing is relative to the base address of the destination flash device. To program the flash memory device, you need only identify the target flash memory device. You can also concatenate multiple flash images (software, binary, SOF/FPGA configuration) and then remotely update all of them at once, if they target the same flash memory device.

If your system is configured to boot from an EPCS device, the following requirements apply:

- The reset address must be set to the base address of the EPCS controller component in the SOPC Builder system.
- The hardware and software flash images must be concatenated prior to remotely updating the images.

Do not update the hardware and software separately, because hardware and software images share the same sector when you use the default EPCS boot method.

Uploading the Images

TFTP is the accepted standard for remotely updating embedded systems over Ethernet. TFTP is a lightweight protocol that is commonly implemented on top of the User Datagram Protocol (UDP). Extending this common methodology to provide remote configuration and update capabilities is straightforward. A sample command line from the host computer follows:

`tftp` <*target hostname* or *ip address*> `PUT ext_flash.flash`

This command causes the target Nios II system and software to receive the file and program it to flash memory. This operation is described in more detail in "Updating Images That are Provided Remotely" on page 5.

Programming Images to Flash Memory

The Nios II EDS includes the Hardware Abstraction Layer (HAL), which provides routines to program a flash memory device. In the Nios II EDS, to program a flash memory device, perform the following four steps:

1. Gather the data you want to program in a buffer.
2. Optionally, determine if the data that is already on the flash memory device is identical to the data you wish to program. If the data are identical, skip the programming.
3. Determine if the flash memory block must be erased, based on the following facts:
   - Programming or writing to flash memory can only change bits from 1 to 0.
   - Erasing a flash memory sector or block sets all bits to 1, after which any value can be programmed.
4. Write or program the flash memory device.

   The low-level actions to program the flash memory device differ for CFI and EPCS flash memories, but the HAL `alt_write_flash_block()` command identifies and implements the correct actions for the relevant type of flash memory.

Triggering a Reconfiguration

To force the system to reconfigure and reboot after successfully uploading hardware, software, or binary data images, you must trigger reconfiguration. You can use TFTP to request a file with the name **reconfig**, by typing the following command:

`tftp <`*target hostname or ip address*`> GET reconfig ↵`

Issue this request to start a sequence of events that initiates a reconfiguration. Most Nios II development boards contain a PLD that acts as a configuration controller. Both the standard and full-featured reference designs, shipped as part of the Nios II EDS, contain a PIO component named `reconfig_request_pio` that is connected to a pin on the PLD. Drive this pin low to force a reset. For more information, refer to "Implementing Reconfiguration That is Triggered Remotely".

Software Example Explanation

This section describes how the target Nios II-based design responds to the host TFTP client commands.

Updating Images That are Provided Remotely

After you run the TFTP `PUT` command to upload a flash file, the following sequence of events occurs:

1. The target software receives a TFTP `WRQ` (write request) packet.
2. The target software checks that the argument filename has a **.flash** extension.

   If the filename has an extension other than **.flash**, a file not found (FNF) error message is sent and the process terminates.

3. The target sends an ACK packet back to the host.
4. The host responds to this ACK by sending the first data packet (512-byte payload).
5. On receipt of this data packet, the target performs the following steps:
   a. Copies the payload to a buffer.
   b. Parses this S-Record buffer and programs the data line-by-line to the flash memory device.
   c. Returns an ACK on completion.
6. The host and target cycle through steps 4 and 5 until the host sends a data packet that contains fewer than 512 bytes, indicating that data transmission is complete.
7. The target completes parsing and programming of this final buffer and prints a message stating the programming is complete.

Implementing Reconfiguration That is Triggered Remotely

The correct method to remotely trigger an FPGA reconfiguration event depends on whether your design contains a remote_update SOPC Builder component, the Max II PFL feature, or neither. This section describes the three methods of remote configuration.

Method 1—Without remote_update Component

After you issue the GET command to get a file named **reconfig**, the following sequence of events occurs:

1. The target software receives a RRQ (read request) packet.
2. The target software checks that the filename argument to the GET command is `reconfig`. Any other filename causes a File not found (FNF) error.
3. The target begins the reconfiguration process by performing the following two write operations:
   a. Write 0 to the `reconfig_request_pio` base address. This is the `data` register.
   b. Write 1 to the `reconfig_request_pio` component's `direction` register.

   These writes tell the bidirectional PIO component `reconfig_request_pio` to drive a 0 on its output signal. This PIO output is connected to the PLD's reconfig-request line. When the reconfig-request line is driven low, the PLD starts the reconfiguration process.

For more information about using the MAX II configuration controller, refer to *AN346: Using the Nios II Configuration Controller Reference Designs*.

Method 2—With remote_update Component

This method is the correct way to trigger reconfiguration if your SOPC Builder system includes a Cyclone III Remote Update Controller component but not the Max II PFL feature. If you have both the component and the feature, you can also use this method. The Cyclone III Remote Update Controller component is an Avalon-ready instance of the ALTREMOTE_UPDATE megafunction, and therefore requires a CFI-compliant flash memory device to temporarily store the new configuration image locally. The design example demonstrates this method on the NEEK.

When you issue the `GET` command to get a file named **reconfig**, the following events take place:

1. The target software receives a `RRQ` packet.
2. The target software checks that the filename argument to the `GET` command is `reconfig`. Any other filename causes an FNF error.
3. The target begins the reconfiguration with the following three write operations:
   a. Write 0 to register 3 of the `remote_update` component to disable the component's watchdog timer.
   b. Write the offset address—the offset address in the flash memory device—to register 4.
   c. Write 1 to address 0x20.

   These writes direct the `remote_update` component of the SOPC Builder system currently configured on the FPGA to reconfigure the FPGA with the hardware image located at the address written in step b.

For more information about the ALTREMOTE_UPDATE megafunction, refer to the *Remote Update Circuitry (ALTREMOTE_UPDATE) Megafunction User Guide*.

Method 3—With Max II Parallel Flash Loader Feature

If your design includes the Max II PFL feature, you can use a third method to trigger a reconfiguration. The design example demonstrates this method on the EDK.

In this method, the Max II CPLD generates the address and control signals to read configuration data from flash memory and convert it to serial data for the Cyclone III device. The Max II device can also generate the configuration clock for the Cyclone III device.

The EDK includes eight flash memory locations for hardware configuration images. Having multiple predefined locations for these images provides you the flexibility to store as many as eight different configuration images in flash memory. The design example loads the hardware image from the eighth location (number 7).

To trigger reconfiguration, write to the System Reset (SRST) field (bit 2) of the Regfile 2 register at offset 0x8 in the Max II register map.

When you issue the `GET` command to get a file named **reconfig**, the following events take place:

1. The target software receives a `RRQ` packet.
2. The target software checks that the filename argument to the `GET` command is `reconfig`. Any other filename causes an FNF error.

3. The target begins the reconfiguration with the following three operations:
   a. Clear and set the Page Select Register (PSR) and Page Select Software Override (PSO) fields in the Regfile 2 register—set the PSR field to 3'b111 and set the PSO bit to 0—to read from the eighth hardware image location.
   b. Write 1 to the SRST bit (bit 2) of the Regfile 2 register at offset 0x8 in the Max II register map, to trigger reconfiguration.

   These writes direct the Max II PFL component of the SOPC Builder system currently configured on the FPGA to reconfigure the FPGA with the hardware image located in the eighth hardware image location on the flash memory device.

For more information about using the MAX II configuration controller, refer to *AN346: Using the Nios II Configuration Controller Reference Designs*.

For more information about the MAX II parallel flash loader feature, refer to *AN386: Using the Parallel Flash Loader with the Quartus II Software*.

Understanding the Software Example Files

The **an429_remote_configuration.zip** file contains the two folders **src/SBT_flow/remote_config** and **src/CLI_flow/rc_iniche**. You can use the files in **remote_config** to create a remote configuration application in the Nios II SBT for Eclipse, download program files, and remotely configure your FPGA device with the Flash Programmer GUI. You can use the files in **rc_iniche** to accomplish the same tasks using a Nios II command-line interface.

Table 2 describes the software example files that are included in the two folders.

Table 2. Source Files in rc_iniche and remote_config (Part 1 of 2)

Filename	Description
create-this-app **create-this-bsp**	Scripts to build the software and hardware images in the command-line interface. These files are provided in **rc_iniche** but not in **remote_config**.
alt_error_handler.c **alt_error_handler.h**	Contains error handling for the networking, MicroC/OS-II real-time operating system (UCOSII), and remote configuration application portions of this software example.
flash_utilities.c **flash_utilities.h**	Contains flash programming functions. The header file, **flash_utilities.h**, defines the variable `DEFAULT_FLASH_TYPE`. Edit this variable as necessary for your system. The default value is `CFI`. You may also have to modify the `GetFlashName()` function to match the flash name(s) in your system.
iniche_init.c **network_utilities.c** **network_utilities.h**	Contains Interniche NicheStack TCP/IP stack and Ethernet device initialization, and MAC address routines. These functions are similar to the routines provided by the Simple Socket Server design example, which is included in the Nios II EDS.
remote_config.c **remote_config.h**	Contains the TFTP server and handler. All TFTP commands are controlled by the `tftp_fsm()` function.
srec_utilities.c	Contains the S-Record parsing routines, `ParseSRECBuf()` and `ParseAndProgramLine()`.
alt_2_wire.c **alt_2_wire.h**	Contains utilities that provide a low-level interface to the EEPROM.

Table 2. Source Files in rc_iniche and remote_config (Part 2 of 2)

Filename	Description
alt_eeprom.c **alt_eeprom.h**	Contains utilities that read, write, dump, and fill the contents of EEPROM devices.
flash_intel_p30.c **flash_intel_p30.h**	Contains flash write and erase routines that are optimized specifically for Intel P30 StrataFlash memory devices.

The files in these two folders support Method 1 ("Method 1—Without remote_update Component" on page 6) and Method 2 ("Method 2—With remote_update Component" on page 7). Versions of these files are also available in the **walkthroughs/NEEK** and **walkthroughs/NiosII_CycloneIII_3C120** folders. The files in **walkthroughs/NEEK** support Method 1 or Method 2 to trigger remote configuration. The versions in **walkthroughs/NiosII_CycloneIII_3C120** support Method 1 or Method 3 to trigger remote configuration. For implementation details, refer to the `reconfig_fpga()` function in the **remote_config.c** source file.

Software Example Walkthrough

This section provides step-by-step instructions to configure your Nios II development board remotely from the command line or from the Nios II SBT for Eclipse with an example provided with this application note. The example is available for several Altera development boards. It provides executable files that implement remote configuration, a hardware image with which to configure your FPGA, in the file **hw.flash**, and a software image to run on the newly configured system, in the file **sw.flash**.

Ethernet Considerations

Because this software example is based on the industry standard sockets interface style of network programming, the code is portable between Ethernet hardware and TCP/IP network stacks. If your network stack is functional, the application should work.

This example is tested using the Interniche NicheStack TCP/IP Stack integrated with the UCOSII.

Programming Your Device and Running the Code

To remotely configure your Nios II development board, perform the following steps:

1. Ensure that your board's JTAG port is connected to your computer.
2. Ensure that your board's Ethernet port is connected to your computer network.
3. Copy the **walkthroughs** folder contents for your development board to a new working directory.
4. Copy the **tftp.exe** file to the **flash_content** subdirectory of your working directory.

5. If you are using the Nios II command-line flow, perform the following steps:
   a. Open two Nios II command shells, shell A and shell B.
   b. In both Nios II command shells, change directory to your working directory.
   c. In shell A, download the SRAM Object File (**.sof**) **rc.sof** and the Executable and Linking File (**.elf**) **rc.elf** from the relevant **walkthroughs** folder to your board, and open a nios2-terminal session, by typing the following command:

```
nios2-configure-sof && nios2-download -g rc.elf && nios2-terminal ↵
```

   The application runs and acquires an IP address for the board. Example 2 shows example output on the nios2-terminal from a successful session.

Example 2. Output of a Successful Remote Configuration Session

```
TFTP-based Remote Configuration starting up...
Your Ethernet MAC address is 00:07:ed:ff:8f:10
Using DHCP to find an IP Address
Assigned IP Address is 192.168.1.41
```

7. If you are using the Nios II SBT for Eclipse, perform the following steps:
   a. To start the Nios II SBT for Eclipse, on the Programs menu, click **Altera** > **Nios II EDS** <*version*> > **Nios II** <*version*> **Software Build Tools for Eclipse**.
   b. When you are prompted to specify a workspace, create a new workspace for this project. The Nios II SBT for Eclipse opens.
   c. Open a Nios II command shell, shell B.
   d. In the Nios II SBT for Eclipse, right-click in the **Project Explorer** tab.
   e. Click **New** > **Nios II Application and BSP from Template**. The **Nios II Application and BSP from Template** dialog box appears.
   f. For SOPC Information File name, browse to **rc.sopcinfo** in your working directory.
   g. For **Project name**, create a new project name, <*project*>.
   h. Under **Templates**, highlight **Blank Project**.
   i. Click **Next**.
   j. Select **Select an existing BSP project from your workspace**.
   k. Click the Create button. The Nios II Board Support Package dialog box appears.
   l. For **BSP name**, type <*project*>_bsp.
   m. For **Operating system**, select **Micrium MicroC/OS-II**.
   n. Click **Finish**. A new project folder <*project*> appears in the Project Explorer tab.
   o. Change directory to the **software** directory and select all of the files and folders.
   p. Drag the files and subfolders to the <*project*> folder in the **Project Explorer** tab.
   q. In the **Project Explorer** tab, right-click <*project*> and click **Nios II > BSP Editor**.

r. In the Nios II BSP Editor, in the **Software Packages** tab, enable **altera_iniche**. This step enables the network component to function.

s. On the **File** menu, click **Save**.

t. Click **Generate**.

u. Close the BSP Editor.

v. In the **Project Explorer** tab, right-click <*project*> and click **Build Project**. This step creates the Executable and Linking Format File (**.elf**) which contains the software image for your project.

w. In the Quartus II Programmer, download the **.sof** file, **rc.sof**, to the board.

x. In the **Project Explorer** tab, right-click <*project*> and click **Run As > Nios II Hardware**. The Run Configurations dialog box appears.

y. Click **Run**.

The application runs and acquires an IP address for the board. Example 2 shows example output in the Nios II SBT for Eclipse console from a successful session.

8. Note the DHCP-assigned IP address, <*IP address*>, that displays in the nios2-terminal or in the Nios II SBT for Eclipse console. In Example 2, <*IP address*> is `192.168.1.41`.
9. In shell B, change directory to the **flash_content** subdirectory of your working directory.
10. To load the **hw.flash** file to your board's flash memory device through the Ethernet connection, in shell B, type one of the following commands:
    - In the Linux operating system, type the following command:

      ```
      tftp -m ascii <IP address> -c put hw.flash ↵
      ```

    - In a Windows operating system, type the following command:

      ```
      ./tftp.exe <IP address> PUT hw.flash ↵
      ```

11. To load the **sw.flash** file to your board's flash memory device through the Ethernet connection, in shell B, type one of the following commands:
    - In the Linux operating system, type the following command:

      ```
      tftp -m ascii <IP address> -c put sw.flash ↵
      ```

    - In a Windows operating system, type the following command:

      ```
      ./tftp.exe <IP address> PUT sw.flash ↵
      ```

    For more information about steps 10 and 11, refer to "Updating Images Remotely".
12. If you are using the Nios II command-line flow, to end the nios2-terminal session, in shell A, press Ctrl-C.
13. If you are using the SBT for Eclipse, to end the Nios II Console session, click the Toggle Nios II Console On/Off icon to the right of the red square Terminate and Remove Launch icon.

14. To trigger reconfiguration of your FPGA with the flash memory device content through the Ethernet connection, and run the new software, in either shell in the Nios II command-line flow or in shell B in the SBT for Eclipse flow, type one of the following commands:

    - In the Linux operating system, type the following command:

      ```
      tftp -m ascii <IP address> -c get reconfig ↵
      ```

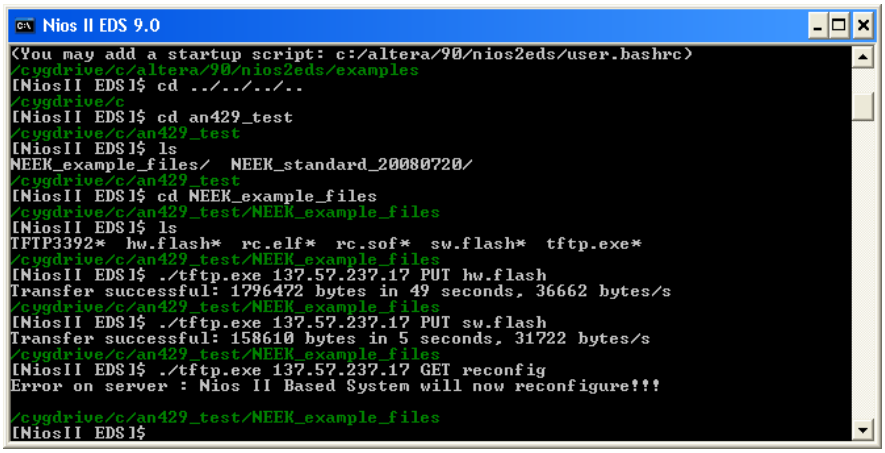
    - In a Windows operating system, type the following command:

      ```
      ./tftp.exe <IP address> GET reconfig ↵
      ```

    ```
    The following message appears:
    "Error on server : Nios II Based System will now reconfigure!!!"
    ```

    Figure 1 shows an example Nios II command shell B session in which the `tftp GET reconfig` command is typed.

Figure 1. tftp Commands to Load Flash Memory and Trigger Reconfiguration



For more information about step 14, refer to "Reconfiguring your Device".

15. Perform one of the following two actions:

    a. In the Nios II command-line flow, in either shell, open a nios2-terminal session, by typing the following command:

       ```
       nios2-terminal ↵
       ```

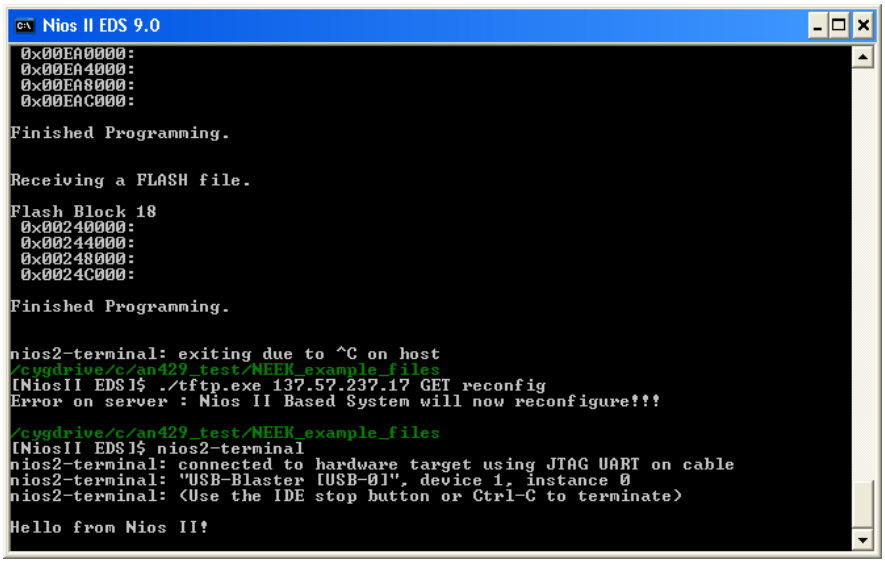
    b. In the SBT for Eclipse, toggle the Nios II Console on or type the following command in shell B:

       ```
       nios2-terminal ↵
       ```

    The message `"Hello from Nios II!"` appears on the nios2-terminal.

Figure 2 shows an example Nios II command shell A session in which the `tftp GET reconfig` command is typed. The figure shows the nios2-terminal output from completion of programming the flash memory device with the **hw.flash** file, and the nios2-terminal output from programming the flash memory device with the **sw.flash** file, followed by the `tftp GET reconfig` command and the `nios2-terminal` command.

Figure 2. Flash Programming Output and Commands to Trigger Reconfiguration and Run the nios2-terminal



If necessary, you can restore the original factory image from the **examples\factory_recovery** directory for your board.

Updating Images Remotely

Using either the provided TFTP client or your own TFTP client, you can write a flash image file to the target board. The target Nios II processor receives the image and programs the flash memory device.

In this example, you write the hardware image to your flash memory device in step 10 and the software image in step 11.

Example 3 shows a sample session.

Example 3. Loading a Flash Image File to a Flash Memory Device Using the TFTP PUT Command

Host-side input:

```
tftp 192.168.1.41 PUT ext_flash.flash
```

Target-side response:

```
Receiving a flash file.
Flash Name is /dev/ext_flash.
Block size is 65536 bytes.
Programming Flash...
Flash Block 0
 0x00000000:
 0x00002000:
 0x00004000:
 0x00006000:
 0x00008000:
 0x0000A000:
 0x0000C000:
 0x0000E000:
Flash Block 1
 0x00010000:
 0x00012000:
 0x00014000:
 0x00016000:
 0x00018000:
 0x0001A000:
 0x0001C000:
 0x0001E000:
Flash Block 2
 0x00020000:
 0x00022000:
 0x00024000:
 0x00026000:
 0x00028000:
 0x0002A000:
 0x0002C000:
Finished Programming.
```

Reconfiguring your Device

After you update your images, type the following TFTP command line to reconfigure the FPGA device:

`tftp <`*DHCP-assigned IP address of the board Ethernet interface*`> GET reconfig ↵`

In the example, you perform this action in step 14. The command resets the CPU and restarts your software. You should see the following response:

```
Error on server : Nios II Based System will now reconfigure!!!
```

The error is intentional. The application uses the TFTP error packet to send the reconfiguration message back to the host.

You may see the following response from the target device:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~
```

This stream of characters displays if you reset and reconfigure the system while connected to it in a nios2-terminal session. If you see this stream of characters in the Nios II SBT for Eclipse, click the red box labeled **Terminate** at the cursor. If you see this stream of characters in a Nios II command shell, press Ctrl-C to terminate the connection.

When you reconnect to your board by starting a new nios2-terminal session, you should see the expected output from the software you reset the system to run. This output indicates that you have successfully updated and reconfigured your system remotely.

Updating Remotely From a Linux System

If you are running the Linux operating system, you can use the **remote_config.sh** script to create the tftp-hpa commands to upload and reconfigure your FPGA device. This script takes three arguments for the `upload` command and two arguments for the `reconfig` command.

Example 4 shows a sample session.

Example 4. Using the remote_config.sh Script for Remote Update on a Linux System

```
$ ./remote_config.sh upload 137.57.237.140 ext_flash.flash
Transfer successful: 698874 bytes in 39 seconds, 17919 bytes/s
$ ./remote_config.sh reconfig 137.57.237.140
Error on server : Nios II Based System will now reconfigure!!!
```

Additional Information

The following sections contain information about several related topics.

Flash Image Locations

You must define the flash memory device's base address for the remote configuration application to know where your flash memory is located.

To define this base address, perform the following steps:

1. Modify `DEFAULT_FLASH_TYPE` in **flash_utilities.h**.
2. Modify the `GetFlashName()` function to match your flash memory component name in SOPC Builder.

For example, if your flash device is CFI compliant and is named **my_flash** in your SOPC Builder system, you could set the `DEFAULT_FLASH_TYPE` variable in **flash_utilities.h** as follows:

```
#define DEFAULT_FLASH_TYPE CFI
```

Your `GetFlashName()` function would contain:

```
static int GetFlashName(char line[30], int flash_type)
{
  if (flash_type == CFI)
  {
    strcpy(line, "/dev/my_flash\0");
  }
  else if (flash_type == EPCS)
```

```
  {
    strcpy(line, "/dev/epcs_controller\0");
  }
  return 0;
}
```

You need update the base address only once for your custom system.

Simple Configuration Options

Although most Nios II development boards use an additional PLD to act as a configuration controller for the board, the PLD is not necessary for reconfiguration.

If your design includes the ALTREMOTE_UPDATE megafunction, whether as the Remote Update Controller component or as a megafunction instantiated in your design directly, the megafunction manages remote configuration of your board.

However, even without the ALTREMOTE_UPDATE megafunction, you should be able to configure a board using only the following elements for configuration and reconfiguration:

- Ethernet PHY or MAC/PHY for Ethernet access
- EPCS device for configuration
- Your FPGA
- Other design-specific components

On your FPGA, wire the `nCONFIG` pin to a `reconfig_request` signal from a component such as PIO in your SOPC Builder system. Pulsing this pin low initiates a reconfiguration. You can use the code in the accompanying software example as a starting point for your remote configuration software.

After reading this application note and studying the accompanying software example, you should be able to modify your existing code to support this feature.

The software example code performs the following three main functions:

- TFTP Server
- S-Record Parsing
- Flash Programming

The TFTP Server is tightly integrated with the TCP/IP stack. The S-Record Parsing and flash programming are accessed through the following functions:

- S-Record Parsing:

```
int ParseSRECBuf( char* buf, int buf_size )
```

- Flash Programming:

```
int ProgFlash( int flash_type, int target_addr,
               char* data, int data_len )
```

Data Consistency and Verification

The software example checks that the received data is correct by dynamically calculating and verifying an S-Record checksum. However, this software does not verify the image after it is written to flash. The TFTP server functionality can be extended to support image verification or, in some cases, checksum calculation. This code is very similar to the code that triggers reconfiguration.

Web-Based Remote Configuration

The Altera Embedded Systems Development Kit, Cyclone III Edition includes the Application Selector. This example software design includes the Board Update Portal (BUP), which provides an intuitive web-based user interface for remote configuration over an Ethernet connection. The BUP enables you to update hardware and software flash content easily. This example also demonstrates the use and capabilities of Asynchronous Javascript and XML (AJAX) in an embedded web server.

For more information about the Embedded Systems Development Kit, Cyclone III Edition (EDK), refer to the Altera Embedded Systems Development Kit, Cyclone III Edition web page.

For more information about AJAX, refer to any online source such as the AJAX tutorial at http://www.w3schools.com/xml/ajax_intro.asp

Summary

This application note provides the information that enables you to implement your own Nios II-based remote configuration system. Because the Nios II processor is a softcore processor, you can update the processor, its entire subsystem, and any custom logic by simply updating the FPGA hardware image and reconfiguring. Remote configuration enables you to perform network-based updates on a massive scale and extends your product's life cycle by allowing you to change both hardware and software in the field.

Document Revision History

Table 3 shows the revision history for this document.

Table 3. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
April 2010 v3.0	■ Modified the example in "Software Example Walkthrough" to include instructions for using the Software Build Tools for Eclipse. ■ Added the Max II Parallel Flash Loader method for remotely configuring the FPGA. ■ Modified the design files to support current development boards.	Updated to add SBT for Eclipse flow and to add PFL method of remote configuration.

Table 3. Document Revision History (Part 2 of 2)

March 2009 v2.0	■ Modified the example in "Software Example Walkthrough" to match new design files, which provide examples for several different Altera development boards. ■ Revised "Web-Based Remote Configuration".	Updated to describe new design files. Added the Board Update Portal.
November 2007 v1.1	■ Added "Updating Remotely From a Linux System". ■ Added files to implement remote update using a flow instead of the IDE. ■ Added note providing factory_recovery location for board.	Added command-line flow functionality for expert users.
October 2003 v1.0	Initial release	—

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support
www.altera.com/support