# Remote Configuration Over Ethernet with the Nios II Processor

November 2007, version 1.1 Application Note 429

## Introduction

Firmware in embedded hardware systems is frequently updated over the Ethernet. For embedded systems comprised of a discrete microprocessor and the devices it controls, the term *firmware* refers to the update of the software image run by the microprocessor. When the embedded system includes an FPGA, firmware updates also include 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 one remote configuration session.

This application note presents a methodology for implementing remote configuration in Nios II-based systems. 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 that the Nios II Embedded Development System (EDS) and the web server that accompanies the Nios II Embedded Evaluation Kit (NEEK) provide additional remote configuration functionality.

## Prerequisites

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

### Knowledge Requirements

This discussion of remote configuration assumes that you have the following:

- Knowledge of network programming, preferably sockets-based. Knowledge of the TFTP protocol is a plus, though a brief description of how the remote configuration example works is given.
- A working knowledge of the Nios II subsystems and the tools necessary to build them. These systems and tools include the Quartus® II software, SOPC Builder, and the Nios II EDS.
- Knowledge of Altera configuration methodologies.
- Experience using the Nios II flash programmer either through the GUI in the IDE or via the command line interface.

- Knowledge of the Motorola S–Record format which is used for all flash programming and remote configuration images.

Refer to *AN 346: Using the Nios Development Board Configuration Controller Reference Designs* for information relevant to this application note. In particular, pay attention to the section on remote Ethernet updating. Refer to the *Nios II Flash Programmer User Guide* for more information about the flash programmer.

## Requirements for the Software Example

To use the software example described in this application note, you must have the following:

- One of the Nios II development kits.

  The 1C12 Nios II Evaluation Kit is not supported.

- A computer with a TFTP client installed. The TFTP client can be found using an Internet search. For Windows, there are multiple TFTP clients available. For convenience, one client is included with this application note. For Linux or UNIX, there are multiple TFTP clients that are freely available.

CAUTION

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

- The **an429_rc_collateral.zip** file which contains all the necessary collateral:
  - **ReadMe.txt**—describes the files
  - **tftp.exe**—the TFTP client for Windows. You can save this executable in your Windows directory.
  - **remote_config.zip**—the software example referenced later in this document for use with the Nios II IDE
  - **rc_iniche.zip**—the software example for use as a command line flow. To use the command line flow, unzip **an429_rc_collateral.zip** in the directory for the board reference design (standard or full-featured) that you are using.
  - **remote_config.sh**—a bash shell script for remote configuration

- Sample flash images (hardware, software, or binary) to upload.

The remainder of this document discusses the following topics:

## High-Level Description

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

1. Preparing the images
2. Uploading the images
3. Programming the images to flash
4. Triggering reconfiguration

The following sections discuss the methods chosen, why they were chosen, and present possible alternatives.

### Preparing the Images

If you have programmed the flash in your design, you probably already have the files necessary to remotely update your design images. The software images are automatically created every time you build your project in the Nios II IDE.

Before programming the flash, the Nios II flash programmer creates flash files using one of three commands:

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

Here is a sample command to create 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 uses the Motorola S-Record format. The addressing is relative to the base address of the destination flash device. On the target side, you only need to know which flash to target for programming. You can also concatenate multiple flash images (software, binary, SOF/FPGA configuration) and then remotely update all of them at once, provided that they are destined for the same flash.

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

- The reset address, in SOPC Builder, must be set to the base address of the EPCS controller component.
- 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 with the provided EPCS boot scheme.

## 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 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. This operation is covered in more detail in "Updating Images Remotely" on page 5.

## Programming Images to Flash

The Nios II EDS includes the Hardware Abstraction Layer (HAL) which provides routines to program the flash. The programming algorithm becomes a four-step task, including one optional step:

1. Gather the data you want to program into a buffer.

2. (optional) Determine if the data that is already on the flash is identical. If it is, skip the programming.

3. Determine if the flash block needs to be erased.

- Programming or writing to flash can only change bits from 1 to 0.
- Erasing a flash sector/block sets all bits to 1, so that any value can then be programmed.

4. Write or program the flash.

   The low-level actions are different for CFI and EPCS flash memories, but the HAL consolidates this into a single command: `alt_write_flash_block()`.

### 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**, as the following example illustrates:

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

You can 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. See "Triggering a Reconfiguration" on page 6, for more information.

## Software Example Explanation

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

### Updating Images Remotely

When you execute the TFTP `PUT` command to upload a flash file, the following process takes place:

1. The target software receives a TFTP `WRQ` (write request) packet.

2. The target software validates that the file has a **.flash** extension.

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

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 into a buffer

   b. Parses this S-Record buffer and programs the data line-by-line to the flash

   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 is less than 512-bytes, indicating that data transmission is complete.

7. The target finishes parsing and programming of this last buffer and prints a message stating the programming is complete.

## Triggering a Reconfiguration

There are two ways to trigger an FPGA reconfiguration event, based upon whether or not your design contains an **altera_avalon_remote_update_cycloneiii** SOPC Builder component named `remote_update`. Each method defines an `fpga_config()` function that triggers an FPGA configuration.

### *Method 1—without remote_update Component*

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

1. The target software receives a `RRQ` (read request) packet.

2. The target software looks for the file, **reconfig**. Any other filename causes an FNF error.

3. The target begins the reconfiguration process with two writes:

   a. A write of 0 to the `reconfig_request_pio's` base address. This is the `data` register.

   b. A write of 1 to the `reconfig_request_pio's direction` register.

These writes tell the bi-directional PIO to drive a 0 on the `reconfig_request_pio's` output. This PIO's output is connected to the PLD's reconfig request line, which starts the reconfiguration process when drive low.

### *Method 2—with remote_update Component*

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 looks for the file **reconfig**. Any other filename causes an FNF error.
3. The target begins the reconfiguration with three writes:
   a. A write of 0 to register 3 of the remote update component to disable the component's watchdog timer.
   b. A write of the offset address in a CFI compliant flash to register 4.
   c. A write of 1 to address 0x20.

   These writes tells the `altremote_update` feature of Cyclone® III devices to reconfigure the FPGA with the hardware image located at the address supplied in step *(b)*.

For more information on the altremote_update feature refer to the *altremote_update Megafunction User Guide*.

## Understanding the Software Example Files

The file **an429_rc_collateral.zip** contains two **.zip** files, **remote_config.zip** and **rc_iniche.zip**. You can use the files in **remote_config.zip** to create a remote configuration application, download program files and remotely configure your FPGA device with the Flash Programmer GUI. You can use the files in **rc_iniche.zip** to accomplish the same tasks using a Nios II command line interface. The contents of the two **.zip** files is similar; however, **rc_iniche.zip** also includes utilities that are needed for the command line flow. Table 1 describes the software example files are included in **remote_config.zip**.

*Table 1. Software Example Files*

| Filename | Description |
|---|---|
| **ReadMe.txt** | Description of the accompanying software example |
| **alt_error_handler.c, alt_error_handler.h** | Contains error handling for the networking, uC-OSII, and remote configuration application portions of this software example. |
| **flash_utilities.c flash_utilities.h** | Flash programming functions. The header file, **flash_utilities.h**, contains the variable, `DEFAULT_FLASH_TYPE`. You should 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 TCP/IP stack and Ethernet device initialization, and MAC address routines. These functions are similar to the routines provided by the Simple Socket Server 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.c, srec.h, srec_utilities.c.** | Contains the S-Record parsing routines, `ParseSRECBuf()` and `ParseAndProgramLine(), srec_decode()`, `srec_encode()` and `srec_strerror`. |
| **alt_2_wire.c, alt_2_wire.h** | Contains utilities to provide a low-level interface to the EEPROM. |
| **alt_eeprom.c, alt_eeprom.h** | Contains utilities to 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. |

## Software Example Walkthrough

This section provides step-by-step instructions to configure your Nios II development board.

### 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. In general, if your network stack is functional, you should not have any problems getting this application to work.

The following configuration was tested with this example:

- Hardware => SMSC LAN91C111 MAC/PHY
- TCP/IP Stack => Interniche NicheStack™ integrated with uC-OSII

## Programming Your Device and Running the Code

The following steps allow you to remotely configure your Nios II development board.

☞ The board's Ethernet port must be connected to your computer's network in order for this application to work.

1. Unzip the file **remote_config.zip** to your software examples directory, <*Nios II EDS install dir*>\**examples**\**software**. This action adds Remote Configuration as another project template in the Nios II IDE.

2. Open the standard or full_featured reference design for your board. Program your board with the SOF for the design you selected.

   Refer to the Quartus II Help for details about programming your board.

3. On the Tools menu, click **SOPC Builder**.

4. Click the **System Generation** tab.

5. Click **Run Nios II IDE**.

6. Create a new Nios II C/C++ Application based on the Remote Configuration Project Template from within the IDE.

   Refer to the Nios II Help for information about creating new applications.

7. Build the project. An **ext_flash.flash** file is created and stored in your **Debug** or **Release** directory, depending upon your configuration settings. It will be used later to update the flash image remotely.

8. To program the project to flash, complete the following steps:

   a. Highlight **remote_config_0** directory in the **Nios II C/C++ Projects** tab.

   b. On the Tools menu, click **Flash Programmer**. If you have never used the flash programmer before, you need to set up a flash configuration.

Refer to the *Nios II Flash Programmer User Guide* for detailed instructions.

c. Click the **Program Flash** button.

9. Observe your running application.

Here is the output from a successful 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
```

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

## Updating Images Remotely

Using either the provided TFTP client or your own TFTP client, you can upload a flash image file to the target. The target receives the image and programs the flash.

For this example, upload the **ext_flash.flash** file that you created in step 7. Before running the tftp command, start a Nios II command shell and change directories to the **Release** or **Debug** directory of your remote configuration project in the IDE.

A sample session follows:

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.
```

Follow the same steps for any other images you want to update. You can also update a hardware image, **standard.flash** or **full_featured.flash**, with this procedure.

## Triggering a Reconfiguration

After you have updated your images, run the following TFTP command line to reconfigure the FPGA device. The command resets the CPU and restarts your software.

Host-side input and response:

```
$ tftp 192.168.1.41 GET reconfig
Error on server : Nios II Based System will now reconfigure!!!
```

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

Target-side response:

```
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿ
```

This stream of characters is a result of resetting and reconfiguring the system while connected via **nios2-terminal**. If you see this stream of characters from within the IDE, click **Terminate**. If you see this from a command shell, type a `CTRL-C` to terminate the connection.

Reconnecting to your board with **nios2-terminal** displays the following output:

```
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
```

This message indicates that you have successfully updated and reconfigured your system remotely.

### Remote Update for Linux Systems

If you are running the linux operating system, you can use the script, **remote_config.sh,** 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.

A sample session follows:

```
$ ./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 on several related topics.

### Flash Image Locations

You must define the flash's base address for the remote configuration application to know where your flash is located. Perform the following steps to define this base address:

1. Modify `DEFAULT_FLASH_TYPE` in **flash_utilities.h**.

2. Modify the `GetFlashName()` function to match your flash 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 only need to update the base address once for your custom system.

## Simple Configuration Options

Though the Nios II development boards use an additional PLD to act as a configuration controller for the board, this is not a necessity for reconfiguration. You should be able to design a board with the following elements for configuration/reconfiguration purposes:

- 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` such as PIO in your SOPC Builder system. Pulsing this pin low initiates a reconfiguration. The code in the accompanying software example can be used 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.

There are three main components to the code in the software example:

- 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 provided checks that the data being received is correct by calculating and verifying an S-Record checksum dynamically. However, it does not verify the image after it has been written to flash. The TFTP server functionality can be extended to support image verification or, perhaps, checksum calculation. This code is very similar to to the code to trigger reconfiguration.

## Web Based Remote Configuration

If you have the web server installed on your board, you can use a combination of HTML forms to upload data and individual web pages to control the flow of the application. Using either the Nios II EDS or the NEEK you can use HTML forms to configure your system remotely by completing the following steps:

1. Fill in a file upload form and press the submit/upload button.

2. The web server stores the uploaded file/data in intermediate memory and/or immediately starts programming it to flash.

3. (Optional) Once the file upload has finished, a flash programming web page appears.

4. Press the program flash button and the flash is programmed with the data you uploaded in steps one and two.

5. The server programs the flash and, upon completion, displays a system reset web page.

6. Press the system reset button to reset the system.

All remote configuration applications take advantage of the flash file format, which contains information on where to store the data and provides a line-by-line data checksum.

More information on the web server design example is available on the altera website **www.altera.com.**

## Summary

This application note has provided you with enough information to implement your own Nios II-based remote configuration system. Because the Nios II processor is a softcore processor, updating the processor, its entire subsystem, and any custom logic is just a matter of updating the FPGA hardware image and re-configuring. 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 2 shows the revision history for this document.

*Table 2. Document Revision History*

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| November 2007 v1.1 | ● Added “Remote Update for Linux Systems”. Added files to implement remote update using a command line 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
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
literature@altera.com