



Remote System Upgrade (ALTREMOTE_UPDATE)

Megafunction User Guide



101 Innovation Drive
San Jose, CA 95134
www.altera.com

Software Version: 10.0
Document Version: 2.5
Document Date: August 2010

Copyright © 2010 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Chapter 1. About this Megafunction

Device Family Support	1-1
General Description	1-1
Remote System Configuration Modes	1-2
Remote Configuration Mode	1-2
Local Configuration Mode	1-3
Remote System Configuration Components	1-4
Page Mode Feature	1-5
Factory Configuration	1-5
Application Configuration	1-6
Watchdog Timer	1-6
Remote Update Sub-Block	1-6
Remote Configuration Registers	1-7
Resource Utilization and Performance	1-7

Chapter 2. Getting Started

Parameter Settings	2-1
Page 1 and Page 2a Settings	2-1
Page 3 Settings	2-1
Page 4 Settings	2-2
Page 5 Settings	2-2
Instantiating Megafunctions in HDL Code or Schematic Designs	2-3
Generating a Netlist for EDA Tool Use	2-3
Using the Port and Parameter Definitions	2-3
Identifying a Megafunction after Compilation	2-4
EDA Simulation	2-4
SignalTap II Embedded Logic Analyzer	2-4
Design Example 1: Parameter Writes and Reconfig	2-4
Design Files	2-4
Generating the Megafunction Variation	2-5
Compiling the Design Example	2-6
Simulating the Design Example	2-6
Design Example 2: Parameter Read	2-8
Design Files	2-8
Generating the Megafunction Variation	2-8
Compiling the Design Example	2-9
Simulating the Design Example	2-10


Chapter 3. Specifications

Verilog HDL Prototype	3-1
VHDL Component Declaration	3-1
VHDL LIBRARY-USE Declaration	3-2
Ports and Parameters	3-2

Additional Information


Document Revision History	Info-1
---------------------------------	--------

This user guide describes the features and behavior of the Remote System Upgrade (ALTREMOTE_UPDATE) megafunction that you can configure through the MegaWizard™ interface in the Quartus® II software.

-  This user guide assumes that you are familiar with megafunctions and how to create them. If you are unfamiliar with Altera megafunctions or the MegaWizard interface, refer to the [Megafunction Overview User Guide](#).

Device Family Support

The ALTREMOTE_UPDATE megafunction is available for all Altera® device families supported by the Quartus II software except the Max series, Cyclone, Cyclone II, and HardCopy II devices.

-  In this user guide, the term *Cyclone III series* includes the Cyclone III and Cyclone III LS devices; the term *Cyclone IV series* includes the Cyclone IV E and the Cyclone IV GX devices. When the term *Cyclone* is used alone, it includes Cyclone III and Cyclone IV series devices.

General Description

The ALTREMOTE_UPDATE megafunction implements a remote system upgrade (also known as *remote system update*) by taking advantage of the dedicated remote system upgrade circuitry available in supported devices.

A remote system upgrade helps you deliver feature enhancements and bug fixes without recalling the product, and reduces time-to-market and extends product life. By using the ALTREMOTE_UPDATE megafunction and the dedicated circuitry, your design can download a new configuration image from a remote location, store it in the configuration memory, and direct the dedicated remote system upgrade circuitry to start a reconfiguration cycle.

The dedicated circuitry performs error detection during and after the configuration process. If there are errors, the circuitry facilitates system recovery by reverting back to a safe, default factory, configuration image and then provides error status information.


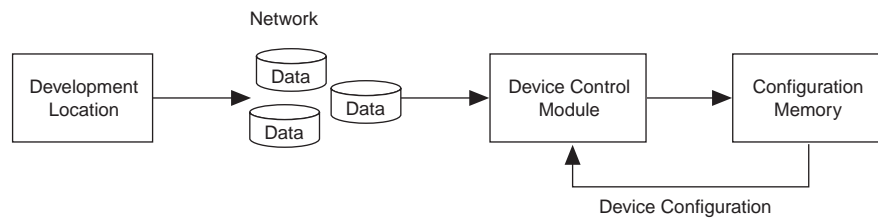
-  For detailed device-specific information on the functionality and implementation of the dedicated system upgrade circuitry, and design guidelines for implementing remote system upgrades with the supported configuration schemes, refer to the [Configuration Handbook](#). For details on configuration schemes and features, refer to the [Configuration Center](#) page of the Altera website.

Figure 1-1 shows a functional diagram for a typical remote system upgrade process.

Figure 1-1. Typical Remote System Upgrade Process



 When using JTAG mode for configuration, remote system upgrade is unsupported.

Remote System Configuration Modes


Remote system upgrade has two modes of operation:


- “Remote Configuration Mode”
- “Local Configuration Mode”

The remote and local configuration modes offer different features and allow you to determine the functionality of your system upon power-up.

Remote Configuration Mode

Use the remote configuration mode to manage up to seven different application configurations and one factory configuration for the devices that use enhanced configuration devices. The seven configuration file limit is due to the number of pages that the PGM[] pins in the device can select.

 If the system receives more than seven configuration files from a remote source over a network or other data source, the previous configuration files are overwritten.

 For more information on the supported enhanced configuration devices for remote upgrades, refer to the *Configuration and Remote System Upgrades* section in the relevant device handbook.

When used with serial configuration devices, the remote update mode allows a configuration space to start at any flash sector boundary, allowing a maximum of 128 pages in the EPCS64 device and 32 pages in the EPCS16 device, in which the minimum size of each page is 512 Kbits. Additionally, the remote update mode features a user watchdog timer that can detect functional errors in an application configuration.

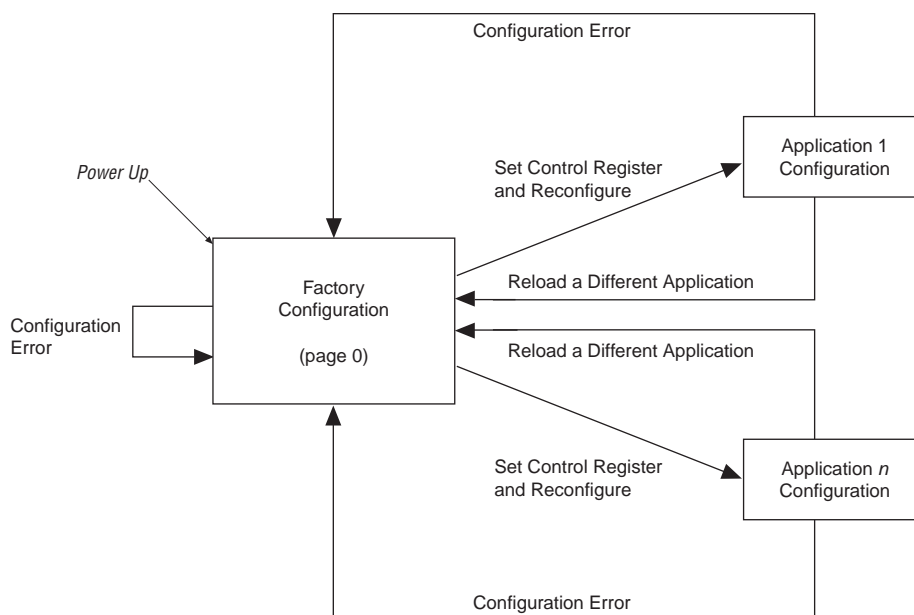
Cyclone III devices support the AP configuration scheme for Altera devices. In the AP configuration scheme, Cyclone III devices are configured using commodity 16-bit parallel flash memory. The flash memory provides a fast interface to access configuration data, because it allows the device to retrieve data on a 16-bit wide parallel data bus.

For more information, refer to the *Configuration Cyclone III Devices* chapter and the *Remote System Upgrade with Cyclone III Devices* chapter in the *Cyclone III Device Handbook*, and to *AN 521: Cyclone III Active Parallel Remote System Upgrade Reference Design*.

On power-up in remote configuration mode, the device loads the user-specified factory configuration file, located at the default page address of 000 in the enhanced configuration device. After the device is configured by the remote update dedicated circuitry, the remote configuration control register points to the page address of the application configuration that needs to be loaded into the device. If an error occurs during user mode of an application configuration, the device reloads the default factory configuration.

Figure 1-2 shows a diagram of remote configuration mode.

Figure 1-2. Remote Configuration Mode



Local Configuration Mode

Use local configuration mode for systems that load an application immediately upon power-up. In local configuration mode, you can only use one configuration, which you can update either remotely or locally.

Not all devices support local configuration mode; check the device handbook for details.

Upon power-up or nCONFIG assertion, the device loads the application configuration. If an error occurs during local configuration mode, the device loads the factory configuration. If you use an enhanced configuration device, page address 001 is the location for the application configuration data, and page address 000 is the location for the factory configuration data.

If the configuration data at page address 001 does not load correctly due to cyclical redundancy check (CRC) failure, if it times out of the enhanced configuration device, or if the external processor times out, the factory configuration located at the default page (page address 000) loads into the device. The user watchdog timer feature is not supported in the local configuration mode. The remote system upgrade control register write access is disabled. However, the device supports read access to obtain error status information.


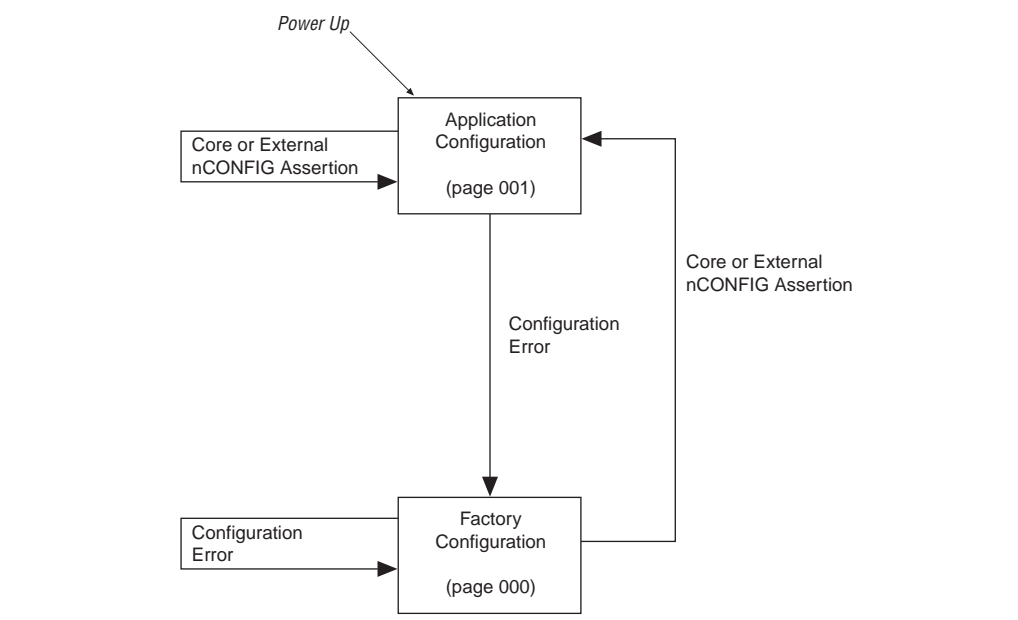
 Local update mode is not supported in the AS configuration scheme (with a serial configuration device).

Figure 1-3 shows the transition between configurations in local configuration mode.

Figure 1-3. Local Configuration Mode




Remote System Configuration Components

The following components are available in all supported devices to support remote and local configuration modes:

- Page mode feature
- Factory configuration
- Application configuration
- Watchdog timer
- Remote update sub-block
- Remote configuration registers

The following sections describe each remote system configuration component in detail; however, these descriptions do not provide full details of all device-specific features.


 For more information on device-specific features, refer to the *Configuration Handbook*.

Page Mode Feature

The page mode feature enables your design to select a location in which it accesses configuration data. The enhanced configuration device can store up to eight different configuration files (one factory and seven application files). Page selection is performed with the PGM[2 . . 0] pins on the devices. On the FPGA, these pins can be designated user I/O pins during standard configuration mode, but in remote system configuration mode, they are used as dedicated output pins.

Upon power-up in remote configuration mode, the factory configuration selects the user-specified page address via the Stratix series device PGM[2 . . 0] output pins. These pins drive the PGM[2 . . 0] input pins of the enhanced configuration device and select the requested page in the memory.

If an intelligent host is used instead of an enhanced configuration device, you should create logic in the intelligent host to support page mode settings similar to that in enhanced configuration devices.

 There are no pgmout pins for the AS configuration scheme, because the erasable programmable configurable serial (EPCS) device does not have inputs for pgm pins in the AS interface making it incapable to read the pgmout pins from the remote update block. The active serial memory interface (ASMI) controller handles this configuration process instead.

When implementing remote system upgrade with AS configuration, a dedicated 7-bit page start address register inside the Stratix II FPGA determines the start addresses for configuration pages within the serial configuration device. The PGM[6 . . 0] registers form bits [22 . . 16] of the 24-bit start address. The other 17 bits are set to zero: $StAdd[23 . . 0] = \{ 1'b0, PGM[6 . . 0], 16'b0 \}$.

In Stratix III devices and later, the dedicated 24-bit start address register PGM[23 . . 0] holds the start address. For both AS and AP configurations, Cyclone devices use a 24-bit boot start address in which you set the most significant 22 bits. Cyclone devices do not support pgmout ports.

During AS configuration, the Arria GX, Arria II GX, Stratix IV, Stratix III, Stratix II, Stratix II GX, or Cyclone III device uses this 24-bit page start address to obtain configuration data from the serial configuration devices.

Factory Configuration


Factory configuration is the default configuration data setup. In enhanced configuration devices, the default page address is 000. Factory configuration data is written into the memory device only once by the system manufacturer and should not be remotely updated or altered. In remote configuration mode, the factory configuration loads into Arria GX, Stratix series, and Cyclone series devices upon power-up. Similarly, in the AS remote configuration scheme, which uses a serial configuration device, the factory configuration loads into the device upon power-up.


If a system encounters an error while loading application configuration data or if the device reconfigures due to nCONFIG assertion, the device loads the factory configuration. The remote system configuration register determines the reason for factory reconfiguration. Based on this information, the factory configuration determines which application configuration to load.

For the Cyclone series devices, upon power-up in remote update in the AP configuration scheme, the Cyclone series devices load the default factory configuration located at address

```
boot_address[23:0] = 24'h010000 = 24'b1 0000 0000 0000 0000.
```

You can change the default factory configuration address to any desired address using the JTAG instruction APFC_BOOT_ADDR. The factory configuration image is stored in non-volatile memory and is never updated or modified using remote access. This corresponds to the default start address location 0x010000 (or the updated address if the default address is changed) in the supported parallel flash memory. Note that 0x010000 is the 16-bit word address for the AP flash memory. However, the Quartus II software implements 8-bit byte addressing, and therefore the correct Quartus II software setting for this address is 0x020000.

 For more information, refer to the *Configuring Cyclone III Devices* chapter and the *Remote System Upgrade with Cyclone III Devices* chapter in the *Cyclone III Device Handbook*, and to *AN 521: Cyclone III Active Parallel Remote System Upgrade Reference Design*.


 For more information on factory configuration in Stratix III devices, refer to the *Configuring Stratix III Devices* chapter and the *Remote System Upgrades with Stratix III Devices* chapter in the *Stratix III Device Handbook*.

Application Configuration

The application configuration is the configuration data received from the remote source and is stored in different locations or pages of the memory storage device (excluding the factory default page).

Watchdog Timer

A watchdog timer is a circuit that determines whether another mechanism functions properly. The watchdog timer functions like a time delay relay that remains in the reset state while an application runs properly. Arria GX, Stratix series, and Cyclone III and Cyclone IV devices are equipped with a built-in watchdog timer for remote system configuration to prevent a faulty application configuration from indefinitely stalling the device. The timer is a 29-bit counter, but you use only the upper 12 bits to set the value for the watchdog timer. The timer begins counting once the device goes into user mode. If the application configuration does not reset the user watchdog timer before time expires, the dedicated circuitry reconfigures the device with the factory configuration and resets the user watchdog timer.

 The user watchdog timer is disabled in local configuration.



Remote Update Sub-Block

The remote update sub-block manages the remote configuration feature. This sub-block, which is controlled by a remote configuration state machine, generates the control signals required to control the various remote configuration registers.

Remote Configuration Registers

The remote configuration registers keep track of page addresses and the cause of configuration errors. You can control both the update and shift registers; the status and control registers are controlled by internal logic, but are read via the shift register. In Stratix III devices and later, the current control registers are 38-bit registers; in Stratix II devices, they are only 21-bit registers.

For Cyclone devices, the remote system upgrade status register has additional capabilities. Three sets of registers store the status for the current application configuration and the two previous application configurations.

-  For more information on this feature in Cyclone III devices, refer to the *Remote System Upgrade with Cyclone III Devices* chapter in the *Cyclone III Device Handbook*, and to *AN 521: Cyclone III Active Parallel Remote System Upgrade Reference Design*
-  For more information on configuration registers, refer to the *Configuration and Remote System Upgrades* section in the relevant device handbook.

Resource Utilization and Performance

For precise details about resource utilization and performance of a specific device, refer to the MegaWizard™ Plug-In Manager and the compilation reports for the device.

Parameter Settings

The most efficient way to customize your megafunction is with the MegaWizard interface. The MegaWizard interface displays only the features that are appropriate to the selected device. Within each selected device, certain features are available, depending on the modes that you specify.

This section provides descriptions of the options available on the individual pages of the ALTREMOTE_UPDATE wizard.

To start the MegaWizard interface from the Quartus II software, on the Tools menu, click **MegaWizard Plug-In Manager**.



For more information on the MegaWizard interface, refer the [Megafunction Overview User Guide](#).

Page 1 and Page 2a Settings

On page 1 of the MegaWizard interface, you can create, edit, or copy a custom megafunction variation.

On page 2a of the ALTREMOTE_UPDATE MegaWizard interface, specify the family of device you want to use, type of output file to create, and the name of the output file. Choose AHDL (.tdf), VHDL (.vhd), or Verilog HDL (.v) as the output file type.

Page 3 Settings

On page 3 of the ALTREMOTE_UPDATE MegaWizard interface, you specify the type of operation mode, the initialization settings of the remote update, the page to be loaded at reconfiguration, and whether you want to use the watchdog timer and enter a value for the timer. Specify the conditions under which reconfiguration should start.








If the currently selected device family is the Arria® GX, Stratix® II, or Stratix II GX device family, in the **Which operation mode will you be using?** pull-down list, select **ACTIVE_SERIAL_REMOTE**. This option allows the remote update block to use the active serial (AS) configuration scheme. Note that there are no `pgmout` output pins in the generated block when the ACTIVE_SERIAL_REMOTE operation mode is selected.



For details on the operation modes for these device families, refer to these documents: [Remote System Upgrades with Arria GX Devices](#) and [Remote System Upgrades with Stratix II and Stratix II GX Devices](#).

If the selected device family is the Stratix, or Stratix GX, the operation mode available is LOCAL and REMOTE. For details on this operation modes, refer to the [Configuration Handbook](#).

-  If the currently selected device family is the Arria II GX, Cyclone III series, Cyclone IV series, HardCopy III, HardCopy IV, Stratix III, Stratix IV, or Stratix V devices, the only operation mode available is REMOTE. For details on the operation modes supported by these device families, refer to the relevant chapters in the *Configuration Handbook*. The **remote update simulation initialization** options are not available for these device families.
-  Only REMOTE and ACTIVE_SERIAL_REMOTE operation modes support writing configuration parameters.
-  The **Enable reconfig POF checking** option is only available for Arria II GX, HardCopy III, HardCopy IV, Stratix III, Stratix IV, and Stratix V device families. This option allows the remote system upgrade circuitry to check and verify the POF header of the application image, when a reconfiguration is triggered to configure the application image. If a valid POF header is found, the device will proceed to load the application image. Otherwise, an error signal will be flagged to indicate an invalid POF header is detected and the device will not start the application configuration. For more information, refer to the *Configuration Handbook*.
-  For a design reference on this feature, refer to *AN 603: Active Serial Remote System Upgrade Reference Design*.
-  You can trigger the remote system upgrade by turning on the check boxes under the **What should start the reconfiguration?** list, which is only available for the device families that support remote update simulation initialization options. If one of the selected items is triggered during the simulation, the reconfiguration process starts. For details about these items (nCONFIG, nSTATUS, and so on) refer to the *Configuration Handbook*.

Page 4 Settings

Page 4 of the ALTREMOTE_UPDATE MegaWizard interface lists the simulation model files needed to simulate the generated design files. You can enable the Quartus II software to generate a synthesis area and timing estimation netlist for use by third-party tools.

Page 5 Settings

On page 5 of the ALTREMOTE_UPDATE MegaWizard interface, you specify the files you wish to generate for your custom megafunction. The gray check marks indicate files that are always generated; the other files are optional and are generated only when selected.

Instantiating Megafunctions in HDL Code or Schematic Designs

When you use the MegaWizard interface to customize and parameterize a megafunction, the MegaWizard interface creates a set of output files that allows you to instantiate the customized function in your design. Depending on the language you choose in the MegaWizard interface, the wizard instantiates the megafunction with the correct parameter values and generates a megafunction variation file (wrapper file) in Verilog HDL format (**.v**), VHDL format (**.vhd**), or AHDL format (**.tdf**) along with other supporting files.

The MegaWizard interface provides options to create the following files:

- A sample instantiation template for the language of the variation file (**_inst.v**, **_inst.vhd**, or **_inst.tdf**)
- Component Declaration File (**.cmp**) that can be used in **.vhd** files
- ADHL Include File (**.inc**) that can be used in **.tdf**
- Quartus II Block Symbol File (**.bsf**) that can be used in schematic designs
- Verilog HDL module declaration file that can be used when instantiating the megafunction as a black box in a third-party synthesis tool (**_bb.v**)



For more information on the wizard-generated files, refer to Quartus II Help or to the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

Generating a Netlist for EDA Tool Use

If you use a third-party EDA synthesis tool, you can instantiate the megafunction variation file as a black box for synthesis. Use the VHDL component declaration or Verilog module declaration black-box file to define the function in your synthesis tool, and then include the megafunction variation file in your Quartus II project.

If you enable the option to generate a synthesis area and timing estimation netlist in the MegaWizard interface, the wizard generates an additional netlist file (**_syn.v**). The netlist file is a representation of the customized logic used in the Quartus II software. The file provides the connectivity of the architectural elements in the megafunction but might not represent true functionality. This information enables certain third-party synthesis tools to better report area and timing estimates. In addition, synthesis tools can use the timing information to focus timing-driven optimizations and improve the quality of results.



For more information on using megafunctions in your third-party synthesis tool, refer to the appropriate chapter in the *Synthesis* section in volume 1 of the *Quartus II Handbook*.

Using the Port and Parameter Definitions

Instead of the MegaWizard interface, you can instantiate the megafunction in your Verilog HDL, VHDL, or AHDL code by calling the megafunction and settings its parameters as you would any other module, component, or subdesign.



Altera recommends that you use the MegaWizard interface for complex megafunctions, which ensures that you properly set all the megafunction parameters.

For a list of the megafunction ports and parameters, refer to [Chapter 3, Specifications](#).

Identifying a Megafunction after Compilation

During Quartus II compilation, analysis and elaboration is performed to build the structure of your design. Locate your megafunction in the Project Navigator window by expanding the compilation hierarchy and locating the megafunction by its name.

Similarly, to search for node names within the megafunction (using the Node Finder), in the **Look in** dialog box, click **Browse (...)** and select the megafunction in the **Hierarchy** dialog box.

EDA Simulation

Depending on the simulation tool that you are using, refer to the appropriate EDA topic in the Quartus II software Help files. These tool-specific topics show you how to perform functional and gate-level timing simulations that include the megafunctions, and include the necessary files and directories in which the files are located.



For more information, refer to the [EDA Tool Support Resource Center](#) page of the Altera website.

SignalTap II Embedded Logic Analyzer

The SignalTap® II embedded logic analyzer provides you with a method to debug all Altera megafunctions in your design. With the SignalTap II embedded logic analyzer, you can capture and analyze data samples for the top-level ports of the Altera megafunctions in your design while your system is running at full speed.

To monitor signals from your Altera megafunctions, you must first configure the SignalTap II embedded logic analyzer in the Quartus II software, and then include the analyzer as part of your Quartus II project. The Quartus II software seamlessly embeds the analyzer along with your design in the selected device.



For more information on using the SignalTap II embedded logic analyzer, refer to the [Design Debugging Using the SignalTap II Embedded Logic Analyzer](#) chapter in volume 3 of the *Quartus II Handbook*.

Design Example 1: Parameter Writes and Reconfig

This design example illustrates the sequence of control signals to set the time out value of the watchdog timer to a required count, thus enabling it. It also illustrates the sequence to select an application page and assert `reconfig` signal to start reconfiguration of the device by using the selected page.

Design Files

The design files are available on the [Literature and Technical Documentation](#) page of the Altera website. The files are located under the following sections:

- [Quartus II Development Software Literature](#) page (expand the **Using Megafunctions** section and then expand the **I/O** section).

- **Literature:** [User Guide](#) section of the Altera website.

In this example, you can must perform the following activities:

- Instantiate remote update block using the ALTREMOTE_UPDATE megafunction and the MegaWizard interface.
- Implement the design and assign the EP1S10B672C6 device to the project.
- Compile and simulate the design.

Generating the Megafunction Variation

To generate the ALTREMOTE_UPDATE megafunction variation and add the variation to the top-level file of your Quartus II project, follow these steps:

1. Unzip the contents of **ALTREMOTE_UPDATE_DesignExample_ex1.zip** to any working hard drive on your computer.
2. In the Quartus II software, extract the **Remote_Update_ex1_1.1.qar** to your working directory, and open the **remote_update.qpf** project.
3. On the Tools menu, select **MegaWizard Plug-In Manager**.
4. Select **Create a new custom megafunction variation**, and click **Next**; the MegaWizard Plug-In Manager [page 2a] appears.
5. In the **Which device family will you be using?** pull-down list, select **Stratix**.
6. Under **Which type of output file do you want to create?** option, select **Verilog HDL**.
7. Under **Installed Plug-Ins**, in the I/O folder, select **ALTREMOTE_UPDATE**.
8. Name the output file as **remote_update_ex1**. Click **Next**. Page 3 appears.
9. In the **Which operation mode will you be using?** pull-down list, select **REMOTE**.
10. Turn on **Add support for writing configuration parameters**.
11. In the **Would you like Remote Update to initiate with a default or application specific settings?** pull-down list, select **FACTORY**.
12. Turn off the **Use the Watchdog timer and set timer to check box**.
13. In the **Which page should be loaded at reconfiguration?** pull-down list, select **3**.
14. Under the **What should start the reconfiguration?** list, turn on the **CRC, POF ID, SW ID Error, Core nConfig asserted, nStatus asserted, and Pin nConfig asserted** check boxes. Turn off the **Watchdog timed out** check box.
15. Click **Next**. Page 4 appears.
16. Turn off the **Generate a netlist** check box.
17. Click **Next**. Page 5 appears.
18. Turn on the **Quartus II symbol file, Instantiation template file, and Verilog HDL black box file** check boxes, and turn off the **AHDL Include file and VHDL Component declaration file** options. Click **Finish**.

The **remote_update_ex1** module is now built.

Compiling the Design Example

To assign the device and compile your design, follow these steps:

1. In the Quartus II software, with the **remote_update.qpf** open, on the Assignments menu, choose **Device**. The **Device** dialog box appears.
2. In the **Family** pull-down list, select **Stratix**.
3. Under **Target device**, select **Specific device selected in 'Available devices' list**.
4. Under **Available devices**, select **EP1S10B672C6**.
5. Click the **Device and Pin Options** button. The **Device and Pin Options** dialog box appears.
6. In the **Category** list, select **Configuration**.
7. In the Configuration scheme, select **Passive Serial (can use Configuration Device)**.
8. In the **Configuration mode** pull-down list, select **Remote**.
9. Under **Configuration device**, turn on **Use configuration device** and select **Auto**.
10. Click **OK** to close the **Device and Pin Options** dialog box.
11. Click **OK** to close the **Device** dialog box.
12. On the Processing menu, choose **Start Compilation**.
13. When the **Full compilation was successful** dialog box displays, click **OK**.

Simulating the Design Example



If you are unfamiliar with using the ModelSim-Altera software, refer to the [Model-Sim Altera Support](#) section on the Altera website. On the support page, there are various links to topics such as installation, usage, and troubleshooting.

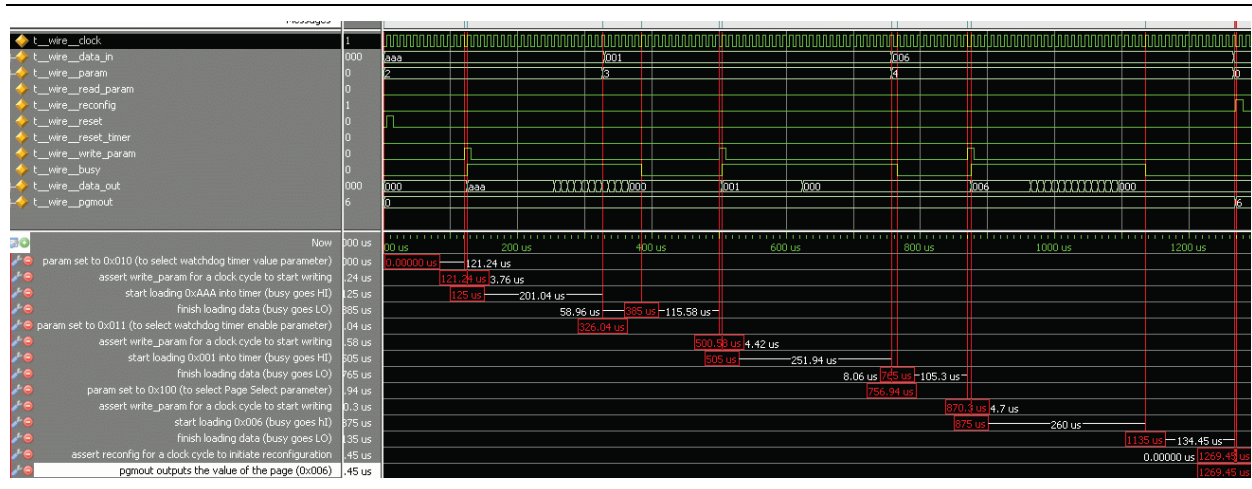
To simulate the design example, follow these steps:

1. Unzip the **ALTREMOTE_UPDATE_ex1_msim.zip** file to any working directory on your PC.
2. Browse to the folder in which you unzipped the files.
3. Open the **remote_update_ex1.do** file in a text editor.
4. In line 1 of the **remote_update_ex1.do** file, replace *<directory_path>* with the directory path of the appropriate library files. For example,
`C:/Modeltech_ae/altera/verilog/stratix`
5. On the File menu, click **Save**.
6. Start the ModelSim-Altera software.
7. On the File menu, click **Change Directory**.
8. Select the folder in which you unzipped the files. Click **OK**.
9. On the Tools menu, point to **Tcl**, and click **Execute Macro**.

10. Select the **remote_update_ex1.do** file and click **Open**. This is a script file for ModelSim-Altera software that automates all the necessary settings for the simulation.
11. Verify the results by looking at the Waveform Viewer window.

Figure 2–1 shows the expected simulation results in the ModelSim-Altera software.

Figure 2–1. Simulation Waveforms Using the ModelSim-Altera Software



The stimulus sets the watchdog timer timeout value to **AAA** (357,826,560 clock cycles), enables the watchdog timer, selects page 6 for reconfiguration, and initiates reconfiguration.

- The first part of the simulation consists of setting the timeout value for the watchdog. The default of `data_in[]` is set to `1010_1010_1010` (357,826,560 clock cycles) to specify the time out value for the watchdog timer.
- The default of `param[]` is set to `010`. This selects the watchdog timer value parameter.
- At time $t = 121.4 \mu\text{s}$, `write_param` is asserted for one clock cycle. This begins the parameter writing operation for the timer value into the remote update block.
- Notice that the `busy` signal goes active at the start of data loading at $t = 125 \mu\text{s}$ until $t = 385.8 \mu\text{s}$, at which point the parameter is written successfully.

The second part of the simulation consists of enabling the watchdog timer.

- The default of `data_in[]` is set to `0001` to enable the watchdog timer. There are only two values for this parameter: 1 (enable) or 0 (disable).
- The default of `param[]` is set to `011`. This selects the watchdog enable parameter.
- At time $t = 501 \mu\text{s}$, `write_param` is asserted for one clock cycle. This begins the parameter writing operation for enabling the watchdog into the remote update block.
- Notice that the `busy` signal goes active at the start of data loading at $t = 505 \mu\text{s}$ until $t = 765 \mu\text{s}$, at which point the parameter is written successfully.

The third part of the simulation consists of setting the Page Select parameter to 6.

- `data_in[]` is set to 0110 (Page 6) to specify the desired page to be loaded during reconfiguration. There are only eight possible values for this parameter: 0 (factory configuration), or 1–7 (application configuration).
- `param[]` is set to 100. This selects the page select parameter.
- At time $t = 870 \mu\text{s}$, `write_param` is asserted for one clock cycle. This begins the parameter writing operation for setting the page select parameter into the remote update block.
- Notice that the `busy` signal goes active at the start of data loading at $t = 875 \mu\text{s}$ until $t = 1135 \mu\text{s}$, at which point the parameter is written successfully.

The final part of the simulation consists of initiating reconfiguration of the chip with the current settings. This is done when `reconfig` is asserted at $t = 1270 \mu\text{s}$. After this, the chip initiates reconfiguration, and the `pgmout[]` pins are driven with the value of the page (6 = 0110) that is loaded during reconfiguration.

Design Example 2: Parameter Read

This design example illustrates the sequence of control signals to read the watchdog enable status, watchdog timeout value, the page set for reconfiguration, and the current configuration state.

Design Files

The design files are available on the [Literature and Technical Documentation](#) page of the Altera website. The files are located under the following sections:

- [Quartus II Development Software Literature](#) page (expand the **Using Megafunctions** section and then expand the **I/O** section)
- [Literature: User Guide](#) section of the Altera website

In this example, you must perform the following activities:

- Instantiate remote update block using the `ALTREMOTE_UPDATE` megafunction and the MegaWizard interface.
- Implement the design and assign the EP1S25F1020C5 device to the project.
- Compile and simulate the design.

Generating the Megafunction Variation

To generate the `ALTREMOTE_UPDATE` megafunction variation and add the variation to the top-level file of your Quartus II project, follow these steps:

1. Unzip the contents of the `ALTREMOTE_UPDATE_DesignExample_ex2.zip` file to your working directory on your PC.
2. In the Quartus II software, extract the `Remote_Update_ex2_1.1.qar` project to your working directory, and open the `remote_update_ex2.qpf` file.
3. On the Tools menu, select **MegaWizard Plug-In Manager**.
4. Select **Create a new custom megafunction variation**, and then click **Next**; the MegaWizard Plug-In Manager [page 2a] appears.

5. In the **Which device family will you be using?** pull-down list, select **Stratix**.
6. Turn on the **Verilog HDL** option under the **Which type of output file do you want to create?** list.
7. Under **Installed Plug-Ins**, in the **I/O** folder, select **ALTREMOTE_UPDATE**.
8. Specify the output file **remote_update_ex2**.
9. Click **Next**. Page 3 appears.
10. In the **Which operation mode will you be using?** pull-down list, select **REMOTE**.
11. Turn on the **Add support for writing configuration parameters** check box.
12. In the **Would you like Remote Update to initiate with a default or application specific settings?** pull-down list, select **FACTORY**.
13. Turn on the **Use the Watchdog timer and set timer to check box**, and specify **100**.
14. In the **Which page should be loaded at reconfiguration?** pull-down list, select **3**.
15. Turn on all check boxes under the **What should start the reconfiguration?** list.
16. Click **Next**. Page 4 appears.
17. Turn off the **Generate a netlist** check box.
18. Click **Next**. Page 5 appears.
19. Turn on the **Quartus II symbol file**, **Instantiation template file**, and **Verilog HDL black-box file declaration file** check boxes. Turn off the **AHDL Include file** and **VHDL component declaration file** check boxes.
20. Click **Finish**.

The **remote_update_ex2** module is now built.

Compiling the Design Example

To assign the device and compile your design, follow these steps:

1. In the Quartus II software, with the **remote_update_ex2.qpf** file open, on the Assignments menu, choose **Device**. The **Device** dialog box appears.
2. In the **Family** pull-down list, select **Stratix**.
3. Under **Target device**, select **Specific device selected in 'Available devices' list**.
4. Under **Available devices**, select **EP1S25F1020C5**.
5. Click the **Device and Pin Options** button. The **Device and Pin Options** dialog box appears.
6. In the Configuration scheme, select **Passive Serial (can use Configuration Device)**.
7. In the **Configuration mode** pull-down list, select **Remote**.
8. Leave the other options in the default state and click **OK**.
9. Click **OK** to close the **Device and Pin Options** dialog box.
10. Click **OK** to close the **Device** dialog box.
11. On the Processing menu, choose **Start Compilation**.

12. When the **Full compilation** was successful dialog box displays, click **OK**.

Simulating the Design Example

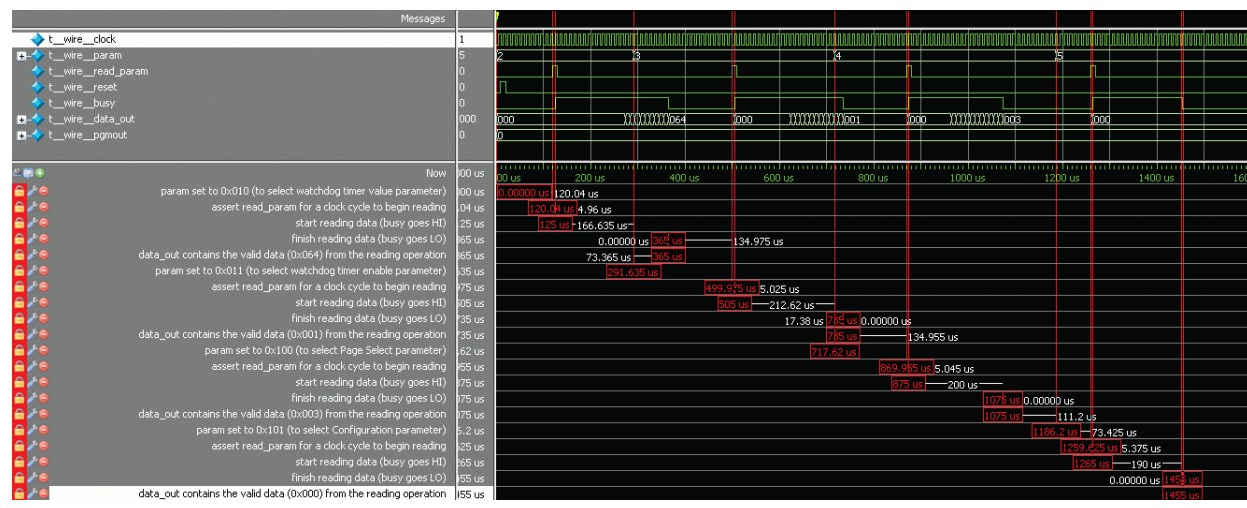
If you are unfamiliar with using the ModelSim-Altera software, refer to the [Model-Sim Altera Support](#) section on the Altera website. On the support page, there are various links to topics such as installation, usage, and troubleshooting.

To simulate the design example, follow these steps.

1. Unzip the **ALTREMOTE_UPDATE_ex2_msim.zip** file to your working directory on your PC.
2. Browse to the folder in which you unzipped the files.
3. Open the **remote_update_ex2.do** file in a text editor.
4. In line 1 of the **remote_update_ex2.do** file, replace `<directory_path>` with the directory path of the appropriate library files. For example,
`C:/Modeltech_ae/altera/verilog/stratix`
5. On the File menu, click **Save**.
6. Start the ModelSim-Altera software.
7. On the File menu, click **Change Directory**.
8. Select the folder in which you unzipped the files. Click **OK**.
9. On the Tools menu, click **Execute Macro**.
10. Select the **remote_update_ex2.do** file and click **Open**. This is a script file for the ModelSim-Altera software that automates all the necessary settings for the simulation.
11. Verify the results by looking at the Waveform Viewer window.

Figure 2-2 shows the expected simulation results in the ModelSim-Altera software.

Figure 2-2. Simulation Waveforms Using the ModelSim-Altera Software



The stimulus reads the watchdog timer timeout value, the watchdog timeout enable parameter, the value of the page set, and the value of configuration parameter.

The first part of simulation consists of reading the value of the watchdog timeout value parameter. The following shows how this was achieved:

- The default setting of `param[]` is set to 010 to select the watchdog timer value parameter.
- The default setting of `read_param` is asserted for one clock cycle, in which $t = 121 \mu\text{s}$; data read begins from the remote update block.
- Notice that the busy signal goes active at the start of data reading at $t = 125 \mu\text{s}$ until $t = 365 \mu\text{s}$, at which point the parameter is read successfully.

`data_out[]` is now valid and contains the current value of the parameter read (in this case, the value is 0000_0110_0100, which is the upper 12 bits of the 29-bit timeout counter, and implies the watchdog timer will timeout at a counter value of 100).

The second part of the simulation consists of reading the value of the watchdog timeout enable parameter. The following shows how this was achieved:

- `param[]` is set to 011 to select watchdog timeout enable parameter
- `read_param` is asserted for one clock cycle where at $t = 500 \mu\text{s}$ data read begins from the remote update block
- Notice that the busy signal goes active at the start of data reading at $t = 505 \mu\text{s}$ until $t = 735 \mu\text{s}$, at which point the parameter is read successfully

`data_out[]` is now valid and contains 0000_0000_0001. The LSB bit being set indicates that the watchdog timer is enabled.

The third part of the simulation consists of reading the value of the page set. The following shows how this was achieved:

- `param[]` is set to 100 to select the Page Select parameter.
- `read_param` is asserted for one clock cycle where at $t = 870 \mu\text{s}$ data read begins from the remote update block.
- Notice that the busy signal goes active at the start of data reading at $t = 875 \mu\text{s}$ until $t = 1075 \mu\text{s}$, at which point the parameter is read successfully.

`data_out[]` is now valid and contains 0000_0000_0011. The three LSB bits indicate the page set value is 3.

The fourth part of the simulation consists of reading the value of the configuration. The following shows what was done:

- `param[]` is set to 101 to select the configuration parameter.
- `read_param` is asserted for one clock cycle, where at $t = 1260 \mu\text{s}$ data read begins from the remote update block.
- Notice that the busy signal goes active at the start of data reading at $t = 1265 \mu\text{s}$ until $t = 1455 \mu\text{s}$, at which point the parameter is read successfully.

`data_out[]` is now valid and contains 0000_0000_0000. The LSB bit indicates the current configuration is factory configuration.

Verilog HDL Prototype

The following Verilog HDL prototype is located in the Verilog Design File (.v) `altera_mf.v` in the `<Quartus II installation directory>\eda\synthesis` directory.

```

module altremote_update
#(
  parameter    check_app_pof = "false",
  parameter    intended_device_family = "unused",
  parameter    in_data_width = 12,
  parameter    operation_mode = "remote",
  parameter    out_data_width = 12,
  parameter    sim_init_config = "factory",
  parameter    sim_init_page_select = 0,
  parameter    sim_init_status = 0,
  parameter    sim_init_watchdog_value = 0,
  parameter    lpm_type = "altremote_update",
  parameter    lpm_hint = "unused")
(
  outputwire   [23:0]asmi_addr,
  inputwire    asmi_busy,
  inputwire    asmi_data_valid,
  inputwire    [7:0]asmi_dataout,
  outputwire   asmi_rden,
  outputwire   asmi_read,
  outputwire   busy,
  inputwire    clock,
  inputwire    [in_data_width-1:0]data_in,
  outputwire   [out_data_width-1:0]data_out,
  inputwire    [2:0]param,
  outputwire   [2:0]pgmout,
  outputwire   pof_error,
  inputwire    read_param,
  inputwire    [1:0]read_source,
  inputwire    re_config,
  inputwire    reset,
  inputwire    reset_timer,
  inputwire    write_param)/* synthesis syn_black_box=1 */;

endmodule //altremote_update

```

VHDL Component Declaration

The following VHDL component declaration is located in the VHDL Design File (.vhd) `altera_mf_components.vhd` in the `<Quartus II installation directory>\libraries\vhdl\altera_mf` directory.

```

component altremote_update
  generic (
    check_app_pof:string := "false";
    intended_device_family:string := "unused";
    in_data_width:natural := 12;
    operation_mode:string := "remote";
    out_data_width:natural := 12;
    sim_init_config:string := "factory";
    sim_init_page_select:natural := 0;
    sim_init_status:natural := 0;
    sim_init_watchdog_value:natural := 0;
    lpm_hint:string := "UNUSED";
  )

```

```

    lpm_type:string := "altremote_update"
  );
  port(
    asmi_addr:out std_logic_vector(23 downto 0);
    asmi_busy:in std_logic := '0';
    asmi_data_valid:in std_logic := '0';
    asmi_dataout:in std_logic_vector(7 downto 0) := (others => '0');
    asmi_rden:out std_logic;
    asmi_read:out std_logic;
    busy: out std_logic;
    clock: in std_logic;
    data_in:in std_logic_vector(in_data_width-1 downto 0) := (others => '0');
    data_out:out std_logic_vector(out_data_width-1 downto 0);
    param: in std_logic_vector(2 downto 0) := (others => '0');
    pgmout:out std_logic_vector(2 downto 0);
    pof_error:out std_logic;
    read_param:in std_logic := '0';
    read_source:in std_logic_vector(1 downto 0) := (others => '0');
    reconfig:in std_logic := '0';
    reset: in std_logic;
    reset_timer:in std_logic := '0';
    write_param:in std_logic := '0'
  );
end component;
```

VHDL LIBRARY-USE Declaration

The VHDL LIBRARY-USE declaration is not required if you use the VHDL Component Declaration.

```

LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;
```

Ports and Parameters

The options listed in this section describe all the ports and parameters available for each device to customize the ALTREMOTE_UPDATE megafunction according to your application.


 **Table 3–1** describes the input ports of the ALTREMOTE_UPDATE megafunction, **Table 3–2** describes the output ports of the ALTREMOTE_UPDATE megafunction, and **Table 3–3** describes the parameters for the ALTREMOTE_UPDATE megafunction. The parameter details are only relevant if you bypass the MegaWizard interface and use the megafunction as a directly parameterized instantiation in your design.

Table 3-1. ALTREMOTE_UPDATE Megafunction Input Ports (Part 1 of 5)

Port Name	Required	Description	Comments
asmi_busy	No	Input from altasmi_parallel component.	Available when the <code>check_app_pof</code> parameter is set to <code>TRUE</code> . A logic high on this pin indicates that the ALTASMI_PARALLEL megafunction is busy processing the operation. The ALTREMOTE_UPDATE megafunction waits for this pin to go low before initiating another operation. Wire this pin to the <code>asmi_busy</code> output port of the ALTASMI_PARALLEL megafunction.
asmi_data_valid	No	Input from altasmi_parallel component.	Available when the <code>check_app_pof</code> parameter is set to <code>TRUE</code> . A logic high on this pin indicates that there is valid data in the <code>asmi_dataout[7..0]</code> output port of the ALTASMI_PARALLEL megafunction. Wire this pin to the <code>asmi_data_valid</code> output port of the ALTASMI_PARALLEL megafunction.
asmi_dataout	No	Input from altasmi_parallel component.	Available when the <code>check_app_pof</code> parameter is set to <code>TRUE</code> . The ALTREMOTE_UPDATE megafunction presents the address information on this pin before initiating the read operation on the ALTASMI_PARALLEL megafunction. Wire this pin to the <code>asmi_addr[23..0]</code> input port of the ALTASMI_PARALLEL megafunction.
clock	Yes	Clock input to the remote update block.	Clock input to control the machine, as well as to drive the remote update block during the update of parameters. This port must be connected to a valid clock.
data_in[]	No	Data input for writing parameter data into the remote update block.	Input bus for parameter data. For some parameters, not all the bits are used. In this case the lower-order bits are used (e.g. status values use bits 0–4). This bus defaults to 0 if left unconnected. The port is ignored if the current configuration is the Application configuration. The width of this bus is device-dependent. <ul style="list-style-type: none"> ■ A 12-bit bus in Arria® GX, Stratix® II, Stratix II GX, Stratix, and Stratix GX devices ■ A 24-bit bus in Stratix III, Stratix IV, and Stratix V devices ■ A 22-bit bus in Cyclone III and Cyclone IV devices

Table 3-1. ALTREMOTE_UPDATE Megafunction Input Ports (Part 2 of 5)

Port Name	Required	Description	Comments
param[]	No	Bus that specifies which parameter need to be read or updated.	<p>A 3-bit bus that selects the parameter to be read or updated. If left unconnected, the default value for this port is 000.</p> <p>For Arria GX, Stratix III, Stratix II, Stratix II GX, Stratix, and Stratix GX devices, mapping to each parameter type and corresponding parameter bit width is defined as follow:</p> <ul style="list-style-type: none"> ■ 000 - Reconfiguration trigger conditions (Read Only) - width of 5. Trigger bits are defined as follow: <ul style="list-style-type: none"> Bit 4: wdtimer_source: User Watchdog Timer timeout Bit 3: nconfig_source: external configuration reset (nCONFIG) assertion Bit 2: runconfig_source: configuration reset triggered from logic array Bit 1: nstatus_source: nSTATUS asserted by an external device as the result of an error Bit 0: crcerror_source: CRC error during application configuration ■ 001 - (illegal value) ■ 010 - Watchdog Timeout Value - width of 12 ■ 011 - Watchdog Enable - width of 1

Table 3-1. ALTREMOTE_UPDATE Megafunction Input Ports (Part 3 of 5)

Port Name	Required	Description	Comments
param[] (continued)			<ul style="list-style-type: none"> ■ 100 - Page Select <p>For Stratix and Stratix GX devices:</p> <ul style="list-style-type: none"> ■ width of 3 when reading and when writing the page. Applies only when using an enhanced configuration device. <p>For Arria GX, Stratix II, and Stratix II GX devices:</p> <ul style="list-style-type: none"> ■ width of 3 when reading and when writing the page. Applies only when using an enhanced configuration device. ■ width of 7 when reading and writing the start address. The PGM[6..0] registers form bits [22..16] of the 24-bit start address. The other 17 bits are set to zero. Applies when using the serial configuration device (AS mode). <p>For Stratix III devices:</p> <ul style="list-style-type: none"> ■ width of 24 when reading and writing the start address. The PGM[23..0] registers form the 24-bit start address. The other 17 bits are set to zero. Applies when using the serial configuration device (AS mode). Stratix III devices use this mode for remote configuration only. ■ 101 - Configuration Mode (AnF) - width of 1. In local update mode, this parameter can only be read. It is set to 1 in application page and is set to 0 in factory page. In remote update mode, this parameter can be read and written. Before loading application page in remote update mode, Altera recommends that you set it to 1. ■ 110 - (illegal value) ■ 111 - (illegal value) <p>For Cyclone III devices, mapping to each parameter type and corresponding parameter bit width is defined as follow:</p> <ul style="list-style-type: none"> ■ 000 - Master State Machine Current State Mode (Read Only) - width of 2. Values are defined as follow: <ul style="list-style-type: none"> 00 Factory mode 01 Application mode 11 Application mode with Master State Machine User Watchdog Timer Enabled

Table 3-1. ALTREMOTE_UPDATE Megafunction Input Ports (Part 4 of 5)

Port Name	Required	Description	Comments
param[] (continued)			<ul style="list-style-type: none"> ■ 001 - Force early CONF_DONE (Cd_early) check - width of 1 ■ 010 - Watchdog Timeout Value - width of 29 when reading and width of 12 when writing. Note that the 12 bits for writing are the upper 12 bits of the 29-bit Watchdog Timeout Value. ■ 011 - Watchdog Enable - width of 1 ■ 100 - Boot Address-width of 24 when reading and width of 22 when writing. Note that the boot address should be written to the upper 22 bits of the 24-bit Boot Address. ■ 101 - (illegal value) ■ 110 - Force the internal oscillator as startup state machine clock (Osc_int) option bit - width of 1 ■ 111 - Reconfiguration trigger conditions (Read Only) - width of 5. Trigger bits are defined as follow: <ul style="list-style-type: none"> Bit 4: nconfig_source: external configuration reset (nCONFIG) assertion Bit 3: crcerror_source: CRC error during application configuration Bit 2: nstatus_source: nSTATUS asserted by an external device as the result of an error Bit 1: wdtimer_source: User Watchdog Timer timeout Bit 0: runconfig_source: configuration reset triggered from logic array <p>All parameters can be written in Factory configuration mode only. Different devices support different parameters. Refer to the <i>Remote System Upgrade</i> chapter in the relevant device handbook.</p>
read_param	No	Read signal for the parameter specified in param[] input port and fed to data_out[] output port.	<p>Signal indicating the parameter specified on the param[] port should be read. The number of bits set on data_out[] depends on the parameter type. The signal is sampled at the rising clock edge. The signal should be asserted for only one clock cycle to prevent the parameter from being read again on a subsequent clock cycle. The busy signal is activated as soon as read_param is read as active. While the parameter is being read, the busy signal remains asserted, and data_out[] has invalid data. When the busy signal is deactivated, data_out[] is valid, another parameter can be read.</p> <p>For information on how to use the read_source port in ALTREMOTE_UPDATE operations for Cyclone III devices, refer to AN 521: Cyclone III Active parallel Remote System Upgrade Reference Design.</p>

Table 3-1. ALTREMOTE_UPDATE Megafunction Input Ports (Part 5 of 5)

Port Name	Required	Description	Comments
read_source	Yes	Specifies whether a parameter value is read from the current or a previous state. Available only in Cyclone III and Cyclone IV devices.	In Cyclone III and Cyclone IV devices only, this 2-bit port specifies the state from which a parameter value is read. This signal is valid only when the <code>read_param</code> signal is valid. Mapping <code>read_source[1..0]</code> to Selected Source is defined as follow: <ul style="list-style-type: none"> ■ 00 - Current State Content in Status Register ■ 01 - Previous State Register 1 Content in Status Register ■ 10 - Previous State Register 2 Content in Status Register ■ 11 - Value in Input Register
reconfig	Yes	Indicates when reconfiguration begins using current parameters.	Signal indicating that reconfiguration of the part should begin using the current parameter settings. A value of 1 indicates reconfiguration should begin. This signal is ignored while <code>busy</code> is asserted to ensure all parameters are completely written before reconfiguration begins.
reset	Yes	Asynchronous reset input to the megafunction.	Asynchronous reset input to initialize the machine to a valid state. The machine must be reset before first use, otherwise the state is not guaranteed to be valid.
reset_timer	No	Reset signal for watchdog timer.	Signal indicating the internal watchdog timer should be reset. Unlike other inputs, this signal is not affected by the <code>busy</code> signal and can reset the timer even when <code>busy</code> is asserted. This signal cannot be used in Local update mode. For the timing specification of this parameter, refer to the specific device handbook.
write_param	No	Write signal for parameter specified in <code>param[]</code> and with value specified in <code>data_in[]</code> .	Signal indicating parameter specified with <code>param[]</code> should be written into RU Block with value specified in <code>data_in[]</code> . The number of bits read from <code>data_in[]</code> depends on the parameter type. The signal is sampled at the rising clock edge. The signal should be asserted for only one clock cycle to prevent the parameter from being rewritten on a subsequent clock cycle. The <code>busy</code> signal is activated as soon as <code>write_param</code> is read as being active. While the parameter is being written, the <code>busy</code> signal remains asserted, and input to <code>data_in[]</code> is ignored. When the <code>busy</code> signal is deactivated, another parameter can be written. This signal is only valid in Factory configuration mode because parameters cannot be written in Application configuration mode. The signal cannot be used in Local update mode.

Table 3–2 describes the output ports of the ALTREMOTE_UPDATE megafunction.

Table 3–2. ALTREMOTE_UPDATE Megafunction Output Ports (Part 1 of 2)

Port Name	Required	Description	Comments
asmi_addr	No	Address signal to altasmi_parallel component.	Available when the <code>check_app_pof</code> parameter is set to <code>TRUE</code> . The ALTREMOTE_UPDATE megafunction presents the address information on this pin before initiating the read operation on the ALTASMI_PARALLEL megafunction. Wire this pin to the <code>asmi_addr[23..0]</code> input port of the ALTASMI_PARALLEL megafunction.
asmi_rden	No	Read enable signal to altasmi_parallel component.	Available when the <code>check_app_pof</code> parameter is set to <code>TRUE</code> . This pin enables the read operation on the ALTASMI_PARALLEL megafunction. Wire this pin to the <code>asmi_rden</code> input port of the ALTASMI_PARALLEL megafunction.
asmi_read	No	Read signal to altasmi_parallel component.	Available when the <code>check_app_pof</code> parameter is set to <code>TRUE</code> . A logic high on this pin initiates the read operation on the ALTASMI_PARALLEL megafunction. Wire this pin to the <code>asmi_read</code> input port of the ALTASMI_PARALLEL megafunction.
busy	No	Busy signal that indicates when remote update block is reading or writing data.	Signal indicating machine is busy either reading or writing a parameter. While this signal is asserted, the machine ignores most of its inputs and cannot be altered until it deasserts this signal. Therefore, changes are made only when the machine is not busy. This signal goes high when <code>read_param</code> or <code>write_param</code> is asserted, and remains high until the read or write operation completes.
data_out[]	No	Data output when reading parameters.	This bus holds read parameter data from the RU block. The <code>param[]</code> value specifies the parameter to read. When the <code>read_param</code> signal is asserted, the parameter value is loaded and driven on this bus. Data is valid when the <code>busy</code> signal is deasserted. If left unconnected, the default value for the port is 000. The width of this bus is device-dependent. <ul style="list-style-type: none"> ■ A 12-bit bus in Arria GX, Stratix II, Stratix II GX, Stratix, and Stratix GX devices. ■ A 24-bit bus in Stratix III devices. ■ A 29-bit bus in Cyclone III and Cyclone IV devices.

Table 3-2. ALTREMOTE_UPDATE Megafunction Output Ports (Part 2 of 2)

Port Name	Required	Description	Comments
pgmout[]	Yes	Specifies which page address of configuration data is loaded when remote update block is reconfigured.	In Arria GX, Stratix II, Stratix II GX, and Stratix devices, a 3-bit bus that must be connected directly to output pins. This bus holds the page address (000 to 111) of the configuration data to be loaded when the part is reconfigured. The external device holding the different configurations reads this bus. In Arria GX, Stratix II, and Stratix II GX devices when ACTIVE_SERIAL_REMOTE is selected, the 7 PGM[6 : 0] bits are used instead. The bits are not sent over to Flash memory via page mode pins PGM[2 : 0] as in normal Remote and Local update mode (the block does not have pgmout output pins). Instead, they are loaded into the instruction registers in the ASMI controller when the ASMI controller issues the read instruction and then shifted over to Flash memory. These bits map to the add[22 : 16] bits of the address of Flash memory.
pof_error	No	Detects and invalid application configuration image.	Available when the check_app_pof parameter is set to TRUE. A logic high on this pin indicates that the ALTREMOTE_UPDATE megafunction detects an invalid application configuration image. If asserted high, you must take corrective action by reloading a new application configuration image or specifying a different address location in the EPCS that contains a valid application configuration image. Wire this pin based on your system requirement.

Table 3-3 describes the ALTREMOTE_UPDATE parameters.

Table 3-3. ALTREMOTE_UPDATE Megafunction Parameters (Part 1 of 2)

Parameter	Type	Comments
check_app_pof	String	Allows you to enable POF checking, which allows the remote update block to verify the existence of an application configuration image before the image is loaded. If set to TRUE, the ALTREMOTE_UPDATE megafunction will check the POF file and send the reconfig signal. The default is FALSE.
in_data_width	Integer	Allows you to specify a value of 24 to instantiate the Stratix III, Stratix IV, and Stratix V Remote Update megafunction.
operation_mode	String	Specifies the operation mode of the altremote_update megafunction. Values are LOCAL, REMOTE, or ACTIVE_SERIAL_REMOTE. If omitted, the default is REMOTE. The ACTIVE_SERIAL_REMOTE value is available for Stratix GX devices only.
out_data_width	Integer	Allows you to specify a value of 24 to instantiate the Stratix III, Stratix IV, and Stratix V Remote Update megafunction.
sim_init_config	String	Specifies the configuration of the remote update block only for simulation purposes. The sim_init_config parameter does not affect compilation or the resulting programming file. Values are FACTORY or APPLICATION. When the sim_init_config parameter is set to FACTORY, you can read and write remote update block parameters. When the sim_init_config parameter is set to APPLICATION, you can only read remote update block parameters. If omitted, the default is FACTORY.

Table 3-3. ALTREMOTE_UPDATE Megafunction Parameters (Part 2 of 2)

Parameter	Type	Comments
<code>sim_init_page_select</code>	Integer	Specifies the page select value only for simulation purposes. The <code>sim_init_page_select</code> parameter does not affect compilation or the resulting programming file. If omitted, the default is 0.
<code>sim_init_status</code>	Integer	Specifies the status register value only for simulation purposes. The <code>sim_init_status</code> parameter does not affect compilation or the resulting programming file. If omitted, the default is 0.
<code>sim_init_watchdog_value</code>	Integer	Specifies the initial watchdog timer value only for simulation purposes. The <code>sim_init_watchdog_value</code> parameter does not affect compilation or the resulting programming file. Values are 0 to 4095. When the <code>sim_init_watchdog_value</code> parameter is set to 0, watchdog is disabled. When the <code>sim_init_watchdog_value</code> parameter is set to a value greater than 0, watchdog is enabled. If omitted, the default is 0.

Document Revision History

The following table displays the revision history for the chapters in this user guide.

Date	Document Version	Changes Made
August 2010	2.5	Updated for Quartus II software v10.0, including: <ul style="list-style-type: none"> ■ Updated the Device Family Support section. ■ Add Parameters table to Specifications chapter. ■ Added new parameters and ports to Specifications chapter. ■ Added new prototypes and declarations sections to Specifications chapter. ■ Updated design example figures and steps.
April 2009	2.4	Updated for Quartus II software v9.0, including: <ul style="list-style-type: none"> ■ Updated the section. ■ Added the Maximum Clock Frequency (f_{MAX}) for the supported devices. ■ Updated ports and parameter tables.
May 2007	2.3	Updated for Quartus II software v7.1, including: <ul style="list-style-type: none"> ■ Updated to include support for Arria® GX devices. ■ Updated to include Cyclone® III device information. ■ Added Referenced Documents section.
March 2007	2.2	Updated Chapter 1 to include Cyclone III support.
December 2006	2.1	Updated Chapter 1 to include Stratix® III support.
September 2006	2.0	General update for Quartus II software version 6.0, including screenshots; added ModelSim®-Altera simulation tool section to Chapter 3.
March 2005	1.0	Initial release.