

This chapter discusses how to create and manage projects, and how to migrate them from one computing platform to another.

Today's larger and more sophisticated FPGA designs are often developed by several engineers and are constantly changing throughout the project lifecycle. Designers must track their project changes to ensure efficient design coordination.

This chapter discusses the following topics:

- “Managing Your Quartus II Projects” on page 4–1
- “Exporting and Importing Version-Compatible Database Files” on page 4–6
- “Managing Projects in a Team-Based Design Environment” on page 4–15
- “Scripting Support” on page 4–16

Managing Your Quartus II Projects

To help you manage your FPGA designs, the Quartus® II software provides tools that assist you with your design tasks, including creating a project, creating assignments, managing revisions, and archiving projects.

A Quartus II project contains all your design files, setting files, and other files necessary for the successful compilation of your design.

- ❓ For more information about creating and opening a project, adding files to and removing files from a project, modifying project settings, saving project changes, and specifying the top-level entity, refer to *Managing Files in a Project* in Quartus II Help.

For more information about libraries, refer to “Specifying Libraries” on page 4–10.

- 👉 On the **General** page of the **Options** dialog box, you can also specify a default directory that automatically stores all project files.

After you create a new project, the Quartus II software automatically generates various project files necessary for successful compilation, including the Quartus II Project File (.qpf) and Quartus II Settings File (.qsf).

- ❓ For more information about the .qpf and .qsf, refer to *Quartus II Project File (.qpf)* and *Quartus II Setting File (.qsf)* in Quartus II Help.
- 🔗 For a list of supported Quartus II project files and design file types, refer to *Introduction to the Quartus II Software* manual.

File Association

Quartus II project files are files associated with a Quartus II project, but are not design files in the project hierarchy. Most project files do not contain design logic. The Quartus II software supports project files such as **.qpf**, Quartus II IP File (**.qip**), and **.qsf**, among others.

Design files are files that contain logic for a Quartus II project. The Compiler compiles the Quartus II design files. The Quartus II software also supports designs created from EDIF Input Files (**.edf**) or Verilog Quartus Mapping Files (**.vqm**) generated by EDA design entry and synthesis tools. You can also create Verilog HDL or VHDL designs in the Quartus II software and EDA design entry tools and either generate EDIF Input Files (**.edf**) and Verilog Quartus Mapping File (**.vqm**), or use the Verilog HDL or VHDL design files directly in Quartus II projects. The Quartus II software also supports use of Quartus II Exported Partition Files (**.qxp**) as source files containing entities you can add to your design.

The Quartus II software sets file type association when you run the Quartus II software version 9.1 or earlier; however, in the Quartus II software versions 10.0 and later, the Quartus II software sets file type association after installation, which can be overwritten if you run prior versions of the Quartus II software after installing Quartus II software versions 10.0 and later. If your files are associated with a different version of the Quartus II software, and you want to associate the files with the Quartus II software version 10.0, you can manually associate the files to the Quartus II software version 10.0.

Example 4–1 shows how to associate files with the current version of the Quartus II software manually:

Example 4–1. Command to Associate Files

```
<path to installation directory>\quartus\bin\qreg.exe --file_assoc ←
```

Editing Text-Based Designs with the Quartus II Text Editor

You can use any text editor with the Quartus II software; however, the Quartus II Text Editor allows you to take advantage of features available only in the Quartus II software, error location, and predefined templates to help you with coding.

The Quartus II software provides templates that allow you to insert predefined code directly into your design file; you can choose from several design languages, and you can directly add TimeQuest analyzer design constraints and megafunction information. You can also create and save your own templates.

- ❓ For more information about editing Quartus II Text Editor files, refer to *Editing Quartus II Text Editor Files* in Quartus II Help. For more information about text editors, refer to *About the Quartus II Text Editor* in Quartus II Help. For more information about the Quartus II Text Editor options and setting a preferred text editor, refer to *Setting Quartus II Text Editor Options* in Quartus II Help.

- 🔗 For more information about the Quartus II language template feature, refer to the “Quartus II Language Templates” section in the *Recommended HDL Coding Styles* chapter in volume 1 of the *Quartus II Handbook*.

Creating Assignments

Assignments control a variety of different functions of the Quartus II software and are an important part of an efficient and effective design. When used in conjunction with a good design practices, assignments can help you successfully compile your design. You can create assignments with different editors and dialog boxes in the Quartus II software or with Tcl scripts. Assignments are logic functions you assign to a physical resource on the device, or compilation resources you assign to logic functions.

Quartus II Settings File

As you create assignments in the Quartus II software, you can choose either to store the assignments in memory temporarily or write the assignment out to the **.qsf** with the **Update assignments to disk during design processing only** option located on the **Processing** page of the **Options** dialog box. You can open the **Options** dialog box by clicking **Options** on the Tools menu. If you turn on the **Update assignments to disk during design processing only** option, the Quartus II software stores all assignments in memory and writes to the **.qsf** when a compilation starts or when you save or close the project. The performance of the software improves when you save assignments in memory. You can view this performance improvement when the Quartus II software stores the project files on a remote data disk.



For more information about the **.qsf**, refer to the *Quartus II Settings File Manual*.

Preserving QSF Format

When you create new assignments, the Quartus II software appends the assignments to the end of the **.qsf**. If you modify an assignment, the corresponding line in the **.qsf** changes to maintain the order of assignments in the **.qsf**, unless you add and remove project source files, or when you add, remove, and exclude members from an assignment group. In these cases, the Quartus II software appends all assignments to the end of the **.qsf**. For example, if you add a new design file to the project, the Quartus II software appends the list of all your design files to the end of the **.qsf**.

The Quartus II software preserves all spaces and tabs for all unmodified assignments and comments. When you create a new assignment or modify an existing assignment in the GUI, the Quartus II software writes the assignment with the default formatting.

Quartus II Default Settings File

You can ensure consistent results when defaults change between versions of the Quartus II software with the **assignment_defaults.qdf**, located in the **bin** or **bin64** directory of the Quartus II installation path.

The Quartus II software reads assignments from various files and stores the assignments in memory. The Quartus II software reads settings files in the following order and assignments in subsequent files take precedence over earlier ones:

1. **assignment_defaults.qdf** from *<Quartus II Installation directory>/bin* or *bin64*
2. **assignment_defaults.qdf** from the project directory
3. *<revision name>_assignment_defaults.qdf* from the project directory
4. *<revision name>.qsf* from the project directory

As the Quartus II software reads each new file, if an existing assignment from an existing project file matches, following rules of case sensitivity, multivalued fields, and other rules, the Quartus II software replaces the old value with a new value. For example, if the first file has an assignment A=1, and the second file has A=2, the software replaces assignment A=1 with assignment A=2.

Creating Timing Assignments

If you create timing assignments with the TimeQuest Timing Analyzer, the Quartus II software creates a Synopsys Design Constraints File (.sdc) that contains your SDC commands.



For more information about TimeQuest analyzer and SDC constraints, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Creating Revisions

In the Quartus II software, a revision is a set of assignments and settings. A project may have multiple revisions, and each revision has its own set of assignments and settings. You can create multiple revisions in a project, and you can create a unique revision based on an existing revision. Creating a unique revision allows you to optimize a design for different results; creating a revision based on an existing revision allows you to try new settings and assignments and then compare the revisions.

Creating a revision of your design allows you to create a new set of assignments and settings for a set of design files without losing your previous assignments and settings. You can perform the following tasks with revisions:

- Create a revision not based on a previous revision. Creating a unique revision allows you to optimize a design for different fundamental reasons, such as to optimize by area in one revision and then optimize for f_{MAX} in another revision. When you create a unique revision, the Quartus II software uses all default settings.
- Create a revision based on an existing revision, but try new settings and assignments in the new revision. A new revision includes all the assignments and settings in the existing revision. You can revert from the new revision to the original revision. You can compare revisions manually, or with features in the Quartus II software.

Managing Project Revisions

The **Revisions** dialog box manages your revisions by allowing you to create and delete a revision, specify the current revision, and compare revisions.

Each time you create a new revision of a project, the Quartus II software creates a new .qsf and adds the name of the new revision to the list of revisions in the .qsf. The name of a new .qsf matches the revision name.

You can compare the compilation results of multiple revisions side by side with the **Compare Revisions** dialog box. The **Compare Revisions** dialog box compares the compilation results of each revision in three categories:

- Analysis & Synthesis
- Fitter
- TimeQuest Timing Analyzer

In addition to viewing the compilation results of each revision, you can also compare the assignments for each revision. Comparing the compilation results and assignments for each revision allows you to gain a better understanding of how different optimization options affect your design.

- ❓ For more information about creating, deleting, specifying, and comparing revisions, refer to *Managing Project Revisions* in Quartus II Help.

Creating New Copies of Your Design

If your design requires that you have two separate copies of your project, rather than just a separate revision, you can create a second copy of your project with the **Copy Project** command. For example, if you have a design that is compatible with a 32-bit data bus and you require a new copy of your design to interface with a 64-bit data bus, you may want a separate copy of the project.


The Quartus II software provides utilities to copy and save different copies of your project. Creating a copy of your project with the **Copy Project** command directs the Quartus II software to copy all your design files, your **.qsf**, and all your associated revisions.

If you are creating a new copy of a project that contains an **.edf** or a **.vqm** from a third-party EDA synthesis tool, first create a copy of your project and then replace any **.edf** or **.vqm** files with the newly generated **.edf** or **.vqm**.

- ❓ For more information about the **Copy Project** command, refer to *Copy Project Dialog Box* in Quartus II Help.

Archiving and Restoring Projects

To share large projects between engineers or to transfer your project to a new version of the Quartus II software, you can archive your project. Archiving your project creates a single compressed Quartus II Archive File (**.qar**) that contains all your design, project, and settings files. The **.qar** contains all the **.qdf** files required to compile your design and restore the original compilation results. When you restore the archive in a different version of the Quartus II software, you must include the **.qdf** in the archive to preserve previous compilation results. For more information about the **.qdf**, refer to “*Quartus II Default Settings File*” on page 4-3.

-  You can copy files listed in the **Source Control** file set for the Archiver with the **Copy Project** command. If you cannot find your source file in the **Source Control** file set, add the source file to your project before copying.

- ② For more information about archiving a project and restoring an archived project, refer to *About Archiving Projects* and *Archiving Projects* in Quartus II Help.

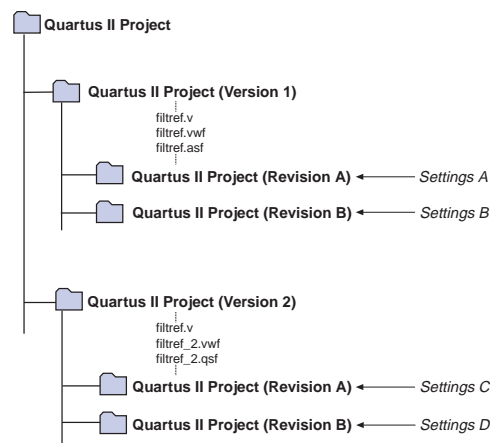
Exporting and Importing Version-Compatible Database Files

The Quartus II software generates version-compatible database files that are a representation of the internal database files. The Quartus II software allows you to export and import version-compatible database files for a project to use compilation databases in different versions of the Quartus II software. If you export version-compatible database files for a project, you can import these files in a future version of the Quartus II software. By importing version-compatible database files and rerunning timing analysis, you can check a project's fitting and timing results in newer versions of the Quartus II software.

Version-compatible databases allows you to use the same project database when you upgrade to a newer version of the Quartus II software, eliminating the need to recompile your project, which saves design time.

Figure 4-1 shows the Quartus II software version-compatible database structure.

Figure 4-1. Quartus II Version-Compatible Database Structure



- ② For more information about exporting and importing version-compatible database files, including device support, refer to *Exporting and Importing Version-Compatible Database Files* in Quartus II Help.

If you require the database files to reproduce the compilation results in the same Quartus II software version, you can use the command-line option to archive a full compilation database. For more information, refer to “*Archiving and Restoring Projects*” on page 4-5.

Migrating to a New Version of the Quartus II Software

To migrate your design to a newer version of the Quartus II software, follow these steps:

1. On the File menu, click **Open Project** and browse to select the Quartus II project file to open the older version of the Quartus II software project.
2. On the Project menu, click **Copy Project** to create a new copy of the project. The older version closes and the copied project opens.
3. Before exporting the database, you must run Analysis and Synthesis for the new version. On the Project menu, click **Export Database**. By default, the Quartus II software exports the database to the **export_db** directory of the copied project. You can also create a new directory.
4. Open the copied project from the new version of the Quartus II software. The Quartus II software deletes the existing database but not the exported database.
5. On the Project menu, click **Import Database**. By default, the Quartus II software selects the directory that contains the exported database you just created. Select the exported database and the Quartus II software imports the version-compatible database files.

Saving the Database in a Version-Compatible Format

To save the database in a version-compatible format during a full compilation, follow these steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Compilation Process Settings**. The **Compilation Process Settings** page appears.
3. Turn on the **Export version compatible database** option.
4. Browse to the directory in which you want to save the database.
5. Click **OK**.

You can also export a project database as version-compatible database files during a full compilation.

- ④ For more information about importing and exporting version-compatible databases, refer to *Exporting and Importing Version-Compatible Database Files* in Quartus II Help.

Quartus II Project Platform Migration

When moving your project from one computing platform to another, you must consider the following cross-platform issues:

- “File Names and Hierarchies”
- “Specifying Libraries”
- “Quartus II Search Path Precedence Rules”
- “Quartus II-Generated Files for Third-Party EDA Tools”
- “Migrating Database Files Between Platforms”

File Names and Hierarchies

To ensure a successful migration across platforms, consider the following differences between operating systems when naming source files, especially when interacting with the operating systems from the command prompt or a Tcl script:

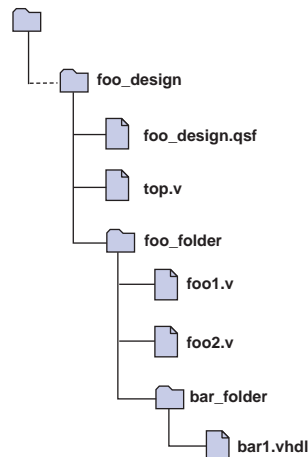
- Some operating system file systems are case sensitive. When writing scripts, ensure that you specify paths exactly, even if the current operating system is not case sensitive. Use lowercase letters when naming files.
- Use a character set common to all the used platforms.
- Do not change the forward-slash (/) and back-slash (\) path separators in the .qsf because the Quartus II software changes all back-slash (\) path separators to forward-slashes (/).
- Observe the shortest file name length limit of the different operating systems you are using.



Altera recommends that you avoid using spaces in the name of the project directory. You can rename the directory with a symbol such as the underscore (_) as a place holder instead of spaces (for example, “my_design” instead of “my design”).

You can specify files and directories inside a Quartus II project as paths relative to the project directory. For example, for a project titled **foo_design** with a directory structure shown in [Figure 4-2](#), specify the source files as: **top.v**, **foo_folder/foo1.v**, **foo_folder/foo2.v**, and **foo_folder/bar_folder/bar1.vhdl**.

Figure 4-2. All Inclusive Project Directory Structure

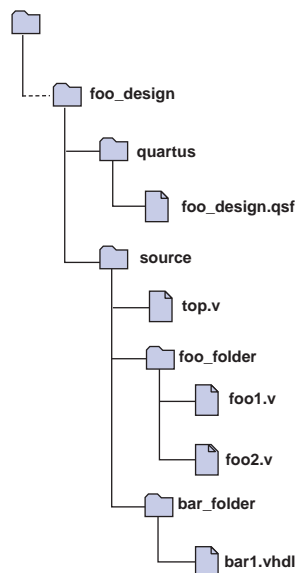



If the **.qsf** is in a directory that is separate from the source files, you can specify paths using the relative and absolute paths and libraries options.

Relative Paths

If the source files are very near to the Quartus II project directory, you can express relative paths using the **..** notation. For example, in the directory structure shown in [Figure 4-3](#), you can specify **top.v** as **../source/top.v** and **foo1.v** as **../source/foo_folder/foo1.v**.

Figure 4-3. Quartus II Project Directory Separate from Design Files




 When you copy a directory structure to a different platform, ensure that all the subdirectories are in the same hierarchical structure and relative path as in the original platform.

Specifying Libraries

You can specify the directory containing source files as a library that the Quartus II software searches when you compile your project. A Quartus II library is a directory containing your Quartus II project design files. You can specify the following libraries in the Quartus II software:

- Project libraries—Apply to a specific project
- Global libraries—Apply to all projects

 The project directory takes precedence over the project libraries.

All files in your libraries are relative to the libraries. For example, if you specify the **user_lib1** directory as a project library and you want to add the **/user_lib1/foo1.v** file to the library, you can specify the **foo1.v** file in the **.qsf** as **foo1.v**. The Quartus II software searches the file in directories that the Quartus II software specifies as libraries.

- ② For more information about libraries, refer to *Libraries Page (Settings Dialog Box)* in Quartus Help.

Specifying Project Libraries

To specify project libraries from the GUI, on the Assignments menu, click **Settings** and select **Libraries**. Type the name of the directory in the **Project Library name** box, or browse to the name of the directory. The **.qsf** of the current revision stores project libraries.

You can also specify project libraries in the **Libraries** page in the **General** category in the **Options** dialog box.

Specifying Global Libraries



To specify global libraries from the GUI, on the Tools menu, click **Options** and select **Libraries**. Type the name of the directory in the **Global Library name** box, or browse to the name of the directory. The **quartus2.ini** file stores global libraries.

To specify libraries from the GUI, on the Assignments menu, click **Settings** and select **Libraries**.

For Windows, the Quartus II software searches for the **quartus2.ini** file in the following directories and order:

1. USERPROFILE, for example, **C:\Documents and Settings\<user name>**
2. Directory specified by the **TMP** environmental variable
3. Directory specified by the **TEMP** environmental variable
4. Root directory, for example, **C:**

For Linux, the Quartus II software creates the file in the **altera.quartus** directory under the **<home>** directory.

-  If the **altera.quartus** directory does not exist, the Quartus II software creates the file in the *<home>* directory.
-  Whenever you specify a directory name in the GUI or in Tcl, the Quartus II software maintains the directory name you use in the **.qsf** rather than resolved to an absolute path.


If the directory is outside of the project directory, the path returned in the dialog box is an absolute path. You can use the **Browse** button in either the **Settings** dialog box or the **Options** dialog box to select a directory. You can change the absolute path to a relative path by editing the absolute path displayed in the **library name** field to create a relative path before you click **Add** to put the directory in the **Libraries** list. Alternatively, you can also select from the **Libraries** list and double-click to edit the path.

When copying projects that specify project libraries, you must either copy your project library files along with the project directory or ensure that your project library files exist in the target platform.

Quartus II Search Path Precedence Rules

If two files have the same file name, the Quartus II software's search path precedence rules determine the found file. The Quartus II software resolves relative paths by searching for the file in the following directories and order:

1. The project directory.
2. The project's database (**db**) directory.
3. Project libraries are searched in the order specified by the **SEARCH_PATH** setting of the **.qsf** for the current revision.

 Altera recommends that you use the **SEARCH_PATH** assignment to define the project libraries. You can have multiple **SEARCH_PATH** assignments. However, you can specify only one source directory for each **SEARCH_PATH** assignment. For more information about **SEARCH_PATH** assignments, refer to [Example 4-18 on page 4-19](#).

4. Global user libraries are searched in the order specified by the **SEARCH_PATH** setting on the **Global User Libraries** page in the **Options** dialog box.
5. The Quartus II software **libraries** directory, for example, *<Quartus II Software Installation directory>\libraries*. For more information about libraries, refer to ["Specifying Libraries Using Scripts" on page 4-19](#).

Quartus II-Generated Files for Third-Party EDA Tools

The project archive and copy features in the Quartus II software do not include Quartus II generated files for third-party EDA tools such as:

- Verilog Output Files (.vo)
- VHDL Output Files (.vho)
- Standard Delay Format Output Files (.sdo) output netlist files
- Stamp model files
- PartMiner XML-Format Files (.xml)
- IBIS Output Files (.ibs)

When you archive your design project, you can save the database in a version-compatible format during a full compilation and include the version-compatible database files in your project archive.



Version-compatible databases may not be available for all device families because the archive does not include the compilation database. If you require the database files to reproduce the compilation results in the same Quartus II software version, you can use the command-line option to archive a full database. For more information, refer to [“Archiving and Restoring Projects” on page 4-5](#).

For more information about saving the database in a version-compatible format and archiving projects, refer to [“Saving the Database in a Version-Compatible Format” on page 4-7](#) and [“Archiving Projects” on page 4-17](#).

To copy your project to another platform, you can regenerate the output netlist or output files by following these steps:

1. Import the version-compatible database. For more information, refer to [“Migrating to a New Version of the Quartus II Software” on page 4-7](#).
2. From the Tools menu, run the TimeQuest analyzer.
3. Run the EDA Netlist Writer.

To restore your project, you can regenerate the output netlist or output files by performing the following steps:

1. Restore your design project. For more information about restoring an archived project, refer to [“Archiving and Restoring Projects” on page 4-5](#).
2. Import the version-compatible database. For more information about migrating to a new version, refer to [“Migrating to a New Version of the Quartus II Software” on page 4-7](#).
3. From the Processing menu, run the TimeQuest analyzer.
4. Run the EDA Netlist Writer.



When you create version-compatible databases, you do not need to recompile your design as you move across platforms.

Migrating Database Files Between Platforms

There is nothing inherent in the file format and syntax of the exported version-compatible database files that might cause problems when migrating the files to other platforms. However, the contents of the database can cause problems for platform migration. For example, using the absolute paths in version-compatible database files generated by the Quartus II software can cause problems for migration. Altera recommends that you change the absolute paths to relative paths before migrating files whenever possible.

Working with Messages

The Quartus II software generates various types of messages, including Information, Warning, Extra Info, Critical Warning, and Error messages. Some messages include information about software status during a compilation and alert you to possible problems with your design. The Messages box in the Quartus II GUI displays messages, and these messages are written to `stdout` when you use command-line executables. In both cases, Quartus II report files write messages.

You can right-click a message in the Message window to get help for the message, locate the source of the message of your design, and manage messages.

Messages provide useful information if you take time to review them after each compilation. The following sections describe the Quartus II software features to help you manage messages.

Messages Window

The Messages window displays nine message tabs, enabling you to review all messages of a certain type. The **Info**, **Extra Info**, **Warning**, **Critical Warning**, and **Error** tabs display messages by type.

- ② For more information about the Messages window and message tabs, refer to [About the Messages Window](#) in Quartus II Help. For more information about managing messages in the Messages window, refer to [Managing Messages in the Messages Window](#) in Quartus II Help.

Message Suppression

You can use message suppression to reduce the number of messages after a compilation by preventing individual messages and entire categories of messages from displaying. For example, if you review a particular message and determine that your design is not the cause of the message, you can suppress the message for subsequent compilations. Message suppression saves time because you see only new messages during subsequent compilations.

Adding a suppressed message creates a suppression rule. Suppressing exact selected messages adds patterns that are exact strings to the suppression rules. Suppressing all similar messages adds patterns with wildcards to the suppression rules.

Furthermore, you can suppress all messages of a particular type in a particular stage of the compilation flow. On the Tools menu, click **Options**. In the **Category** list, select **Suppression** in the **Messages** section.

Suppressing individual messages is controlled in two locations in the Quartus II GUI. You can right-click on a message in the Messages window and choose commands in the Suppress sub-menu entry. To open the Message Suppression Manager, right-click in the Messages window. From the Suppress sub-menu, click **Message Suppression Manager**. For more information about the Message Suppression Manager, refer to [“Message Suppression Manager” on page 4-15](#).

Message Suppression Methods

You can use the following methods to create suppression rules:

- **Suppress Exact Selected Messages**
- **Suppress All Similar Messages**
- **Suppress All Flagged Messages**

If you suppress a message with the **Suppress Exact Selected Messages** option, the Quartus II software suppresses only messages that match the exact text during subsequent compilations. The **Suppress All Similar Messages** option behaves like a wildcard pattern on variable fields in messages and the **Suppress All Flagged Messages** option only suppresses flagged messages.

[Example 4-2](#) shows an example of suppressing common Info type of messages:

Example 4-2. Example of Suppressing Common Info Type Message

```
Info: Found 1 design units, including 1 entities, in source file mult.v.
```

This Info type of message is common during synthesis. The Quartus II software displays the message for each processed source file with varying information about the number of design units, entities, and source file name.

[Example 4-3](#) shows an example of this message in Help:

Example 4-3. Example of Suppressing Common Info Type Message

```
Found <number> design units, including <number> entities, in source file <name>.
```

Choosing to suppress all similar messages effectively replaces the variable parts of that message (<number>, <number>, and <name>) with wildcards.

[Example 4-4](#) shows the suppression rule to suppress common Info type of messages:

Example 4-4. Suppression Rule to Suppress Common Info Type of Messages

```
Info: Found * design units, including * entities, in source file *.
```

The Quartus II software suppresses all messages that match the pattern.

Message Suppression Details and Limitations

The following rules describe which messages that you can suppress and how to suppress them:

- You cannot suppress error messages or messages with information about Altera legal agreements.
- Suppressing a message also suppresses all its submessages, if any.
- Suppressing a submessage causes the Quartus II software to suppress matching submessages only if the parent messages are the same.
- You cannot create your own custom wildcards to suppress messages.
- You must use the Quartus II GUI to manage message suppression, including choosing messages to suppress. The Quartus II software suppresses these messages during compilation in the GUI and when using command-line executables.
- The Quartus II software suppresses the messages on a per-revision basis, not for an entire project. The Quartus II software stores information about which messages to suppress in a file called *<revision>.srf*. If you create a revision based on a suppressed messages revision, the Quartus II software copies the suppression rules file to the new revision. You cannot make all revisions in one project using the same suppression rules file.
- You cannot remove messages or modify message suppression rules while a compilation is running.

Message Suppression Manager

You can use the Message Suppression Manager to view and suppress messages, view and delete suppression rules, and view suppressed messages. The Message Suppression Manager has three tabs labeled **Suppressible Messages**, **Suppression Rules**, and **Suppressed Messages**.

- ② For more information about the Message Suppression Manager, refer to *About Message Suppression* in Quartus II Help.


Managing Projects in a Team-Based Design Environment

The Quartus II software provides several methods to help you manage efficient design coordination across multiple designers and design iterations. You can use the following features to preserve and track project changes:

- Creating Revisions
- Managing Different Design Versions
- Creating Version-Compatible Databases
- Archiving Projects

- ② For more information about creating revisions, managing different design versions, creating version-compatible databases, and archiving projects in a team-based design environment, refer to *About Project Management* in Quartus II Help.

The Quartus II software supports top-down incremental compilation flows. With top-down compilation, one designer or project lead compiles the entire design in the software. Different designers or IP providers can design and verify different parts of the design, and the project lead can add design entities to the project as they are completed. However, the project lead compiles and optimizes the top-level project as a whole. Completed parts of the design can have fitting results and performance fixed as other parts of the design change.

 For more information about incremental compilation for team-based design, refer to the *Quartus II Incremental Compilation for Hierarchical and Team-Based Design* chapter in volume 1 of the *Quartus II Handbook*. For more information about best practices for incremental compilation partitions and floorplan assignments, refer to *Best Practices for Incremental Compilation Partitions and Floorplan Assignments* chapter in volume 1 of the *Quartus II Handbook*.

Scripting Support

You can run procedures and create settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. For more information about scripting command options, refer to the Quartus II Command-Line and Tcl API Help browser.

[Example 4-5](#) shows the command to run the Help browser:

Example 4-5. Command to Run the Help Browser

```
quartus_sh --qhelp ↵
```

Managing Revisions

You can use the following commands to create and manage revisions. For more information about managing revisions, including creating and deleting revisions, setting the current revision, and getting a list of revisions, refer to “[Creating Revisions](#)” on page 4-4.

Creating Revisions

The `-based_on` and `-set_current` options are optional. You can also use `-copy_results` option to copy results from the “`based_on`” revision.

[Example 4-6](#) shows a Tcl command to create a new revision called `speed_ch`, based on a revision called `chiptrip`, and sets the new revision as the current revision:

Example 4-6. Creating Revisions Command

```
create_revision speed_ch -based_on chiptrip -set_current
```

Setting the Current Revision

The `-force` option enables you to open the revision that you specify under revision name and overwrite the compilation database if the database version is incompatible.

[Example 4-7](#) shows the Tcl command to specify the current revision:

Example 4-7. Specifying the Current Revision Command

```
set_current_revision -force <revision name>
```

Getting a List of Revisions

[Example 4-8](#) shows the Tcl command to get a list of revisions in the opened project:

Example 4-8. Getting a List of Revisions in an Opened Project Command

```
get_project_revisions <project_name>
```

Deleting Revisions

[Example 4-9](#) shows the Tcl command to delete a revision:

Example 4-9. Deleting a Revision Command

```
delete_revision <revision name>
```

Archiving Projects

You can archive projects with a Tcl command or a command run at the system command prompt.

[Example 4-10](#) shows the Tcl command to create a project archive with the default settings, overwriting the existing specified archived file:

Example 4-10. Creating a Project Archive with the Default Settings Command

```
project_archive archive.qar -overwrite ↵
```

You can change default settings with the `project_archive` command with options such as:

- `-all_revisions`
- `-include_libraries`
- `-include_outputs`
- `-use_file_set <file_set>`
- `-version_compatible_database`



For new device families, a version-compatible database might not be available because the archive does not include the compilation database. If you require the database files to reproduce the compilation results in the same Quartus II software version, you can use the `-use_file_set full_db` command-line option to archive a full database. For more information, refer to [“Archiving and Restoring Projects” on page 4-5](#).

[Example 4-11](#) shows the command to create a project archive called **top**:

Example 4-11. Creating a Project Archive Command

```
quartus_sh --archive top ↵
```

You can overwrite the existing archive file with the `-overwrite` option.

Restoring Archived Projects

You can restore archived projects with a Tcl command or with a command run at a command prompt. For more information about restoring archived projects, refer to [“Archiving and Restoring Projects”](#) on page 4-5.

[Example 4-12](#) shows the Tcl command to restore the project archive named **archive.qar** in the **restored** subdirectory and overwrite existing files:

Example 4-12. Restoring a Project Archive Command

```
project_restore archive.qar -destination restored -overwrite ↵
```

[Example 4-13](#) shows the command to restore a project archive:

Example 4-13. Restoring a Project Archive Command

```
quartus_sh --restore archive.qar ↵
```

Importing and Exporting Version-Compatible Databases

You can import and export version-compatible databases with either a Tcl command or a command run at a command prompt. For more information about importing and exporting version-compatible databases, refer to [“Exporting and Importing Version-Compatible Database Files”](#) on page 4-6.



The `flow` and `database_manager` packages contain commands to manage version-compatible databases.

[Example 4-14](#) shows the Tcl command to import or export version-compatible databases from the `database_manager` package.

Example 4-14. Importing and Exporting Version-Compatible Databases Command

```
export_database <directory> ↵
import_database <directory> ↵
```

[Example 4-15](#) shows the Tcl commands from the `flow` package to import or export version-compatible databases. If you use the `flow` package, you must specify the database directory variable name.

Example 4-15. Importing and Exporting Version-Compatible Databases from the flow Package Command

```
set_global_assignment -name VER_COMPATIBLE_DB_DIR <directory>
execute_flow -flow export_database
execute_flow -flow import_database
```

[Example 4-16](#) shows the Tcl commands to generate version-compatible databases after every compilation.

Example 4-16. Generating Version-Compatible Databases After Every Compilation Command

```
set_global_assignment -name AUTO_EXPORT_VER_COMPATIBLE_DB ON
set_global_assignment -name VER_COMPATIBLE_DB_DIR <directory>
```

[Example 4-17](#) shows the `quartus_cdb` and the `quartus_sh` executables to manage version-compatible databases:

Example 4-17. quartus_cdb and quartus_sh Executable

```
quartus_cdb <project> -c <revision>--export_database=<directory> ←
quartus_cdb <project> -c <revision> --import_database=<directory>←
quartus_sh -flow export_database <project> -c \ <revision> ←
quartus_sh -flow import_database <project> -c \ <revision> ←
```

Specifying Libraries Using Scripts

In Tcl, use commands in the `::quartus::project` package to specify project libraries. To specify project libraries, use the `set_global_assignment` command.

[Example 4-18](#) shows the typical usage of the `set_global_assignment` command:

Example 4-18. Commands to Specify Project Libraries Using the SEARCH_PATH Assignment


```
set_global_assignment -name SEARCH_PATH "../other_dir/library1"
set_global_assignment -name SEARCH_PATH "../other_dir/library2"
set_global_assignment -name SEARCH_PATH "../other_dir/library3"
```


To report any project libraries specified for a project and any global libraries specified for the current installation of the Quartus II software, use the `get_global_assignment` and `get_user_option` Tcl commands.

[Example 4-19](#) shows that the Tcl script outputs the user paths and global libraries for an open Quartus II project:

Example 4-19. Commands to Report Specified Project Libraries

```
get_global_assignment -name SEARCH_PATH
get_user_option -name SEARCH_PATH
```

 For more information about Tcl scripting, refer to the *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*. For more information about all settings and constraints in the Quartus II software, refer to the *Quartus II Settings File Manual*. For more information about command-line scripting, refer to the *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*.

 For more information about Tcl scripting, refer to the *API Functions for Tcl* in Quartus II Help.

Conclusion

Designers often try different settings and versions of their designs throughout the development process. The Quartus II project revisions facilitate the creation and management of different assignments and settings. Project archives are useful to save your results, or pass designs between different members of a team. In addition, understanding how to migrate your projects from one computing platform to another, controlling messages, and reducing compilation time are important as well. The Quartus II software facilitates efficient management of your design to accommodate today's sophisticated FPGA designs.

Document Revision History


Table 4-1 shows the revision history for this chapter.


Table 4-1. Document Revision History (Part 1 of 2)

Date	Version	Changes
November 2011	10.1.1	Template update.
December 2010	10.1.0	<ul style="list-style-type: none"> ■ Changed to new document template. ■ Removed Figure 4-1, Figure 4-6, Table 4-2. ■ Moved "Hiding Messages" to Help. Moved information in "Message Suppression Manager" on page 4-15 to Help. ■ Removed references about the <code>set_user_option</code> command in "Specifying Libraries Using Scripts" on page 4-19. ■ Removed Classic Timing Analyzer references.
July 2010	10.0.0	<ul style="list-style-type: none"> ■ Major reorganization done to this chapter. ■ Updated "Working with Messages" on page 4-17. Added a link to Help. Removed Figure 4-2 on page 4-7, Figure 4-11 on page 23, and Figure 4-12 on page. ■ Updated "Specifying Libraries" on page 4-14 section. Changed "User Libraries" to "Libraries". Removed "Reducing Compilation Time" on page 4-26. ■ Added "Managing Projects in a Team-Based Design Environment" on page 4-22 and "File Association" on page 4-2. ■ Updated Figure 4-1 on page 4-6, Figure 4-2 on page 4-8, Figure 4-6 on page 4-18, Figure 4-6 on page 4-19, and Figure 4-7 on page 4-21.
November 2009	9.1.0	<ul style="list-style-type: none"> ■ Updated "Creating a New Project" on page 4-4, "Archiving a Project" on page 4-9, "Restoring an Archived Project" on page 4-11. ■ Added "Quartus II Text Editor" on page 4-2, "Reducing Compilation Time" on page 4-32. ■ Updated Table 4-1 on page 4-10, Table 4-2 on page 4-20. ■ Updated Figure 4-4 on page 4-9, Figure 4-7 on page 4-19.

Table 4-1. Document Revision History (Part 2 of 2)

Date	Version	Changes
April 2009	9.0.0	Updated to fix “Document Revision History” for version 9.0.0.
March 2009	9.0.0	<ul style="list-style-type: none"> ■ Updated “Managing Quartus II Projects” on page 4-1, “Creating a New Project” on page 4-2, “Using Revisions with Your Design” on page 4-3, “Creating and Deleting Revisions” on page 4-4, “Creating New Copies of Your Design” on page 4-6, “Version-Compatible Databases” on page 4-11, “Quartus II Project Platform Migration” on page 4-12, “Filenames and Hierarchies” on page 4-12, “Quartus II Search Path Precedence Rules” on page 4-15, “Quartus II-Generated Files for Third-Party EDA Tools” on page 4-15, “Migrating Database Files between Platforms” on page 4-16, “Message Suppression” on page 4-20, “Quartus II Settings File” on page 4-24, “Quartus II Default Settings File” on page 4-25, “Managing Revisions” on page 4-26, “Archiving Projects” on page 4-26 and “Archiving Projects with the Quartus II Archive Project Feature” on page 4-7, “Importing and Exporting Version-Compatible Databases” on page 4-27, “Specifying Libraries Using Scripts” on page 4-28, “Conclusion” on page 4-30. ■ Updated Figure 4-1, Figure 4-7, Figure 4-8, and Figure 4-11. ■ Updated Table 4-1 and Table 4-2. ■ Updated Example 4-3, Example 4-4, Example 4-5, and Example 4-6.

 For previous versions of the *Quartus II Handbook*, refer to the [Quartus II Handbook Archive](#).

 Take an [online survey](#) to provide feedback about this handbook chapter.

