

This chapter provides a complete reference of all available commands, options, and settings for the Nios® II Software Build Tools (SBT). This reference is useful for developing your own embedded software projects, packages, or device drivers.



Before using this chapter, read the *Getting Started from the Command Line* chapter of the *Nios II Software Developer's Handbook*, and familiarize yourself with the parts of the *Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook* that are relevant to your tasks.

This chapter includes the following sections:

- “Nios II Software Build Tools Utilities” on page 15–1
- “Nios II Design Example Scripts” on page 15–34
- “Settings Managed by the Software Build Tools” on page 15–37
- “Application and User Library Makefile Variables” on page 15–76
- “Software Build Tools Tcl Commands” on page 15–79
- “Software Build Tools Path Names” on page 15–125

Nios II Software Build Tools Utilities

The build tools utilities are an entry point to the Nios II SBT. Everything you can do with the tools, such as specifying settings, creating makefiles, and building projects, is made available by the utilities.

All Nios II SBT utilities share the following behavior:

- Sends error messages and warning messages to stderr.
- Sends normal messages (other than errors and warnings) to stdout.
- Displays one error message for each error.
- Returns an exit value of 1 if it detects any errors.
- Returns an exit value of 0 if it does not detect any errors. (Warnings are not errors.)
- If the help or version command-line option is specified, returns an exit value of 0, and takes no other action. Sends the output (help or version number) to stdout.
- When an error is detected, suppresses all subsequent operations (such as writing files).



Logging Levels

All the utilities support multiple status-logging levels. You specify the logging level on the command line. Table 15-1 shows the logging levels supported. At each level, the utilities report the status as listed under **Description**. Each level includes the messages from all lower levels.

Table 15-1. Nios II SBT Logging Levels

Logging Level	Description
silent (lowest)	No information is provided except for errors and warnings (sent to <code>stderr</code>).
default	Minimal information is provided (for example, start and stop operation of SBT phases).
verbose	Detailed information is provided (for example, lists of files written).
debug (highest)	Debug information is provided (for example, stack backtraces on errors). This level is for reporting problems to Altera.

Table 15-2 shows the command-line options used to select each logging level. Only one logging level is possible at a time, so these options are all mutually exclusive.

Table 15-2. Selecting Logging Level

Command-Line Option	Logging Level	Comments
none	default	If there is no command-line option, the default level is selected.
--silent	silent	Selects silent level of logging.
--verbose	verbose	Selects verbose level of logging.
--debug	debug	Selects debug level of logging.
--log <fname>	debug	All information is written to <fname> in addition to being sent to the <code>stdout</code> and <code>stderr</code> devices.

Setting Values

The value of a setting is specified with the `--set` command-line option to `nios2-bsp-create-settings` or `nios2-bsp-update-settings`, or with the `set_setting` Tcl command. The value of a setting is obtained with the `--get` command-line option to `nios2-bsp-query-settings` or with the `get_setting` Tcl command.

For more information about settings values and formats, refer to “Settings Managed by the Software Build Tools” on page 15-37.

Utility and Script Summary

The following command-line utilities and scripts are available:

- “nios2-app-generate-makefile”
- “nios2-bsp-create-settings” on page 15–6
- “nios2-bsp-generate-files” on page 15–8
- “nios2-bsp-query-settings” on page 15–9
- “nios2-bsp-update-settings” on page 15–11
- “nios2-lib-generate-makefile” on page 15–13
- “nios2-bsp-editor” on page 15–15
- “nios2-app-update-makefile” on page 15–16
- “nios2-lib-update-makefile” on page 15–19
- “nios2-swexample-create” on page 15–22
- “nios2-elf-insert” on page 15–23
- “nios2-elf-query” on page 15–24
- “nios2-convert-ide2sbt” on page 15–25
- “nios2-c2h-generate-makefile” on page 15–29
- “nios2-bsp” on page 15–31
- “nios2-bsp-console” on page 15–33

nios2-app-generate-makefile

Usage

```
nios2-app-generate-makefile [--app-dir <directory>]
  --bsp-dir <directory> [--debug]
  [--elf-name <filename>] [--extended-help] [--help]
  [--log <filename>] [--no-src] [--set <name> <value>]
  [--silent] [--src-dir <directory>]
  [--src-files <filenames>] [--src-rdir <directory>]
  [--use-lib-dir <directory>] [--verbose]
  [--version] [--c2h]
```

Options

- `--app-dir <directory>`: Directory to place the application makefile and executable and linking format file (.elf). If omitted, it defaults to the current directory.
- `--bsp-dir <directory>`: Specifies the path to the BSP generated files directory (populated using the **nios2-bsp-generate-files** command).
- `--debug`: Output debug, exception traces, verbose, and default information about the command's operation to stdout.
- `--elf-name <filename>`: Name of the .elf file to create. If omitted, it defaults to the first source file specified with the file name extension replaced with .elf and placed in the application directory.
- `--extended-help`: Displays full information about this command and its options.
- `--help`: Displays basic information about this command and its options.
- `--log <filename>`: Create a debug log and write to specified file. Also logs debug information to stdout.
- `--no-src`: Allows no sources files to be set in the Makefile. You must add source files in manually before compiling
- `--set <name> <value>`: Set the makefile variable called <name> to <value>. If the variable exists in the managed section of the makefile, <value> replaces the default settings. If the variable does not already exist, it is added. Multiple `--set` options are allowed.
- `--silent`: Suppress information about the command's operation normally sent to stdout.
- `--src-dir <directory>`: Searches for source files in <directory>. Use . to look in the current directory. Multiple `--src-dir` options are allowed.
- `--src-files <filenames>`: Adds a list of space-separated source file names to the makefile. The list of file names is terminated by the next option or the end of the command line. Multiple `--src-files` options are allowed.
- `--src-rdir <directory>`: Same as `--src-dir` option but recursively search for source files in or under <directory>. Multiple `--src-rdir` options are allowed and can be freely mixed with `--src-dir` options.
- `--use-lib-dir <directory>`: Specifies the path to a dependent user library directory. The user library directory must contain a makefile fragment called **public.mk**. Multiple `--use-lib-dir` options are allowed.

- `--verbose`: Output verbose, and default information about the command's operation to stdout.
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

The **nios2-app-generate-makefile** command generates an application makefile (called Makefile). The path to a BSP created by **nios2-bsp-generate-files** is a mandatory command-line option. If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

You can enable support for the Nios II C2H Compiler with the `--c2h` option, which creates a null C2H makefile fragment in your project, and includes it in the application makefile. This makefile fragment, `c2h.mk`, contains comments to help you fill in the makefile variables by hand.



This **c2h.mk** overwrites any existing **c2h.mk**.

You can use the command-line tool **nios2-c2h-generate-makefile** to generate a populated C2H makefile fragment.

The Nios II C2H Compiler is an optional feature. It is available only if you enable **Legacy Package: Nios II IDE / GCC3 Toolchain / C2H Compiler** when you install the Altera® Complete Design Suite.



For detailed information about installing the Altera Complete Design Suite, refer to the *Altera Software Installation and Licensing Manual*.



The **nios2-c2h-generate-makefile** script is available to support pre-existing command-line C2H projects. Create new C2H projects using the Nios II IDE.

For more details about this command, use the `--extended-help` option to display comprehensive usage information.

nios2-bsp-create-settings

Usage

```
nios2-bsp-create-settings [--bsp-dir <directory>]
  [--cmd <tcl command>] [--cpu-name <cpu name>]
  [--debug] [--extended-help] [--get-cpu-arch]
  [--help] [--jdi <filename>]
  [--librarian-factory-path <directory>]
  [--librarian-path <directory>] [--log <filename>]
  [--script <filename>] [--set <name> <value>]
  --settings <filename> [--silent]
  --sopc <filename> --type <OS name> [--type-version <version>]
  [--verbose] [--version]
```

Options

- **--bsp-dir <directory>**: Path to the directory where the BSP files are generated. Use . for the current directory. The directory <directory> must exist. This command overwrites preexisting files in <directory> without warning.
- **--cmd <tcl command>**: Runs the specified Tcl command. Multiple --cmd options are allowed. Available Tcl commands are described in [“Tcl Commands for BSP Settings” on page 15-79](#).
- **--cpu-name <cpu name>**: The name of the Nios II processor that the BSP supports. Optional for a single-processor system. Use ? to list available Nios II processor names.
- **--debug**: Sends debug information, exception traces, verbose output, and default information about the command’s operation, to stdout.
- **--extended-help**: Displays full information about this command and its options. Also displays Tcl command help for the --cmd and --script options.
- **--get-cpu-arch**: Queries for processor architecture from the processor specified. Does not create a BSP.
- **--help**: Displays basic information about this command and its options.
- **--jdi <filename>**: The location of the JTAG Debugging Information File (.jdi) generated by the Quartus® II software. The .jdi file specifies the name-to-node mappings for the JTAG chain elements. The tool inserts the JTAG Debugging Information File (.jdi) path in **public.mk**. If no .jdi path is specified, the command searches the directory containing the SOPC Information File (.sopcinfo), and uses the first .jdi file found.
- **--librarian-factory-path <directory>**: Comma-separated librarian search path. Use \$ for default factory search path.
- **--librarian-path <directory>**: Comma-separated librarian search path. Use \$ for default search path.
- **--log <filename>**: Creates a debug log and write to specified file. Also logs debug information to stdout.
- **--script <filename>**: Run the specified Tcl script with optional arguments. Multiple --script options are allowed. Scripts can use the Tcl commands described in [“Tcl Commands for BSP Settings” on page 15-79](#).

- `--set <name> <value>`: Sets the setting called `<name>` to `<value>`. Multiple `--set` options are allowed.
- `--settings <filename>`: File name of the BSP settings file to create. This file is created with a `.bsp` file extension. It overwrites any existing settings file.
- `--silent`: Suppresses information about the command's operation normally sent to stdout.
- `--sopc <filename>`: The **.sopcinfo** file used to create the BSP.
- `--type <OS name>`: BSP type. Use `?` or `types` to list available BSP types for this option. Use names to list the display names of available BSP types. For a Nios II DPX system, always set this argument to `lwhal`.
- `--type-version <version>`: BSP software component version. By default the latest version is used. `default` value can be used to reset to this default behavior. Use `?` to list available BSP types and versions.
- `--verbose`: Sends verbose output, and default information about the command's operation, to stdout.
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

If you use **nios2-bsp-create-settings** to create a settings file without any command-line options, Tcl commands, or Tcl scripts to modify the default settings, it creates a settings file that fails when running **nios2-bsp-generate-files**. Failure occurs because the **nios2-bsp-create-settings** command is able to create reasonable defaults for most settings, but the command requires additional information for system-dependent settings. The default Tcl scripts set the required system-dependent settings. Therefore it is better to use default Tcl scripts when calling **nios2-bsp-create-settings** directly. For an example of how to use the default Tcl scripts, refer to the **nios2-bsp** script.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to `stderr`.

For more details about this command, use the `--extended-help` option to display comprehensive usage information.

Example

```
nios2-bsp-create-settings --settings my_settings.bsp --sopc \  
  ../my_sopc.sopcinfo --type hal --script default_settings.tcl
```

nios2-bsp-generate-files

Usage

```
nios2-bsp-generate-files --bsp-dir <directory>
[--debug] [--extended-help] [--help]
[--librarian-factory-path <directory>]
[--librarian-path <directory>] [--log <filename>]
--settings <filename> [--silent] [--verbose]
[--version]
```

Options

- **--bsp-dir <directory>**: Path to the directory where the BSP files are generated. Use . for the current directory. The directory <directory> must exist. This command overwrites preexisting files in <directory> without warning.
- **--debug**: Sends debug, exception trace, verbose, and default information about the command's operation to stdout.
- **--extended-help**: Displays full information about this command and its options.
- **--help**: Displays basic information about this command and its options.
- **--librarian-factory-path <directory>**: Comma-separated librarian search path. Use \$ for default factory search path.
- **--librarian-path <directory>**: Comma-separated librarian search path. Use \$ for default search path.
- **--log <filename>**: Creates a debug log and writes to specified file. Also logs debug information to stdout.
- **--settings <filename>**: File name of an existing BSP Settings File (.bsp) to generate files from.
- **--silent**: Suppresses information about the command's operation normally sent to stdout.
- **--verbose**: Sends verbose and default information about the command's operation to stdout.
- **--version**: Displays the version of this command and exits with a zero exit status.

Description

The **nios2-bsp-generate-files** command populates the files in a BSP directory. The path to an existing .bsp file and the path to the BSP directory are mandatory command-line options. Files are written to the specified BSP directory. Generated files are created unconditionally. Copied files are copied from the Nios II EDS installation folder only if they are not present in the BSP directory, or if the existing files differ from the installation files.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

For more details about this command, use the **--extended-help** option to display comprehensive usage information.

nios2-bsp-query-settings

Usage

```
nios2-bsp-query-settings [--cmd <tcl command>]  
  [--debug] [--extended-help] [--get <name>]  
  [--get-all] [--help]  
  [--librarian-factory-path <directory>]  
  [--librarian-path <directory>] [--log <filename>]  
  [--script <filename>] --settings <filename>  
  [--show-descriptions] [--show-names] [--silent]  
  [--verbose] [--version]
```

Options

- `--cmd <tcl command>`: Run the specified Tcl command. Multiple `--cmd` options are allowed.
- `--debug`: Output debug, exception traces, verbose, and default information about the command's operation to stdout.
- `--extended-help`: Displays full information about this command and its options.
- `--get <name>`: Display the value of the setting called `<name>`. Multiple `--get` options are allowed. Each value appears on its own line in the order the `--get` options are specified. Mutually exclusive with the `--get-all` option.
- `--get-all`: Display the value of all BSP settings in order sorted by option name. Each option appears on its own line. Mutually exclusive with the `--get` option.
- `--help`: Displays basic information about this command and its options.
- `--librarian-factory-path <directory>`: Comma-separated librarian search path. Use `$` for default factory search path.
- `--librarian-path <directory>`: Comma-separated librarian search path. Use `$` for default search path.
- `--log <filename>`: Create a debug log and write to specified file. Also logs debug information to stdout.
- `--script <filename>`: Run the specified Tcl script with optional arguments. Multiple `--script` options are allowed.
- `--settings <filename>`: File name of an existing BSP settings file to query settings from.
- `--show-descriptions`: Displays the description of each option after the value.
- `--show-names`: Displays the name of each option before the value.
- `--silent`: Suppress information about the command's operation normally sent to stdout.
- `--verbose`: Output verbose, and default information about the command's operation to stdout.
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

The **nios2-bsp-query-settings** command provides information from a .bsp file. The path to an existing .bsp file is a mandatory command-line option. The command does not modify the settings file. Only requested information is displayed on stdout; no informational messages are displayed.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

For more details about this command, use the `--extended-help` option to display comprehensive usage information.

nios2-bsp-update-settings

Usage

```
nios2-bsp-update-settings [--bsp-dir <directory>]  
  [--cmd <tcl command>] [--cpu-name <cpu name>]  
  [--debug] [--extended-help] [--help] [--jdi <filename>]  
  [--librarian-factory-path <directory>]  
  [--librarian-path <directory>] [--log <filename>]  
  [--script <filename>] [--set <name> <value>]  
  --settings <filename> [--silent]  
  [--sopc <filename>] [--verbose] [--version]
```

Options

- **--bsp-dir <directory>**: Path to the directory where the BSP files are generated. Use . for the current directory. The directory <directory> must exist.
- **--cmd <tcl command>**: Run the specified Tcl command. Multiple --cmd options are allowed.
- **--cpu-name <cpu name>**: The name of the Nios II processor that the BSP supports. This argument is useful if the hardware design contains multiple Nios II processors. Optional for a single-processor design.
- **--debug**: Output debug, exception traces, verbose, and default information about the command's operation to stdout.
- **--extended-help**: Displays full information about this command and its options.
- **--help**: Displays basic information about this command and its options.
- **--jdi <filename>**: The location of the .jdi file generated by the Quartus II software. The .jdi file specifies the name-to-node mappings for the JTAG chain elements. The tool inserts the .jdi path in **public.mk**. If no .jdi path is specified, the command searches the directory containing the .sopcinfo file, and uses the first .jdi file found.
- **--librarian-factory-path <directory>**: Comma-separated librarian search path. Use \$ for default factory search path.
- **--librarian-path <directory>**: Comma-separated librarian search path. Use \$ for default search path.
- **--log <filename>**: Create a debug log and write to specified file. Also logs debug information to stdout.
- **--script <filename>**: Run the specified Tcl script with optional arguments. Multiple --script options are allowed.
- **--set <name> <value>**: Set the setting called <name> to <value>. Multiple --set options are allowed.
- **--settings <filename>**: File name of an existing BSP settings file to update.
- **--silent**: Suppress information about the command's operation normally sent to stdout.
- **--sopc <filename>**: The .sopcinfo file to update the BSP with. It is recommended to create a new BSP if the design has changed significantly. This argument is useful if the path to the original .sopcinfo file has changed.

- `--verbose`: Output verbose, and default information about the command's operation to `stdout`.
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

The **`nios2-bsp-update-settings`** command updates an existing Nios II **`.bsp`** file. The path to an existing **`.bsp`** file is a mandatory command-line option. The command modifies the settings file so the file must have write permissions. You might want to use the `--script` option to pass the default Tcl script to the **`nios2-bsp-update-settings`** command to make sure that your BSP is consistent with your system (this is what the **`nios2-bsp`** command does).

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to `stderr`.

For more details about this command, use the `--extended-help` option to display comprehensive usage information.

nios2-lib-generate-makefile

Usage

```
nios2-lib-generate-makefile [--bsp-dir <directory>]
  [--debug] [--extended-help] [--help]
  [--lib-dir <directory>] [--lib-name <filename>]
  [--log <filename>] [--no-src]
  [--public-inc-dir <directory>] [--set <name> <value>]
  [--silent] [--src-dir <directory>]
  [--src-files <filenames>] [--src-rdir <directory>]
  [--use-lib-dir <directory>] [--verbose]
  [--version]
```

Options

- **--bsp-dir <directory>**: Path to the BSP generated files directory (populated using the **nios2-bsp-generate-files** command).
- **--debug**: Output debug, exception traces, verbose, and default information about the command's operation to stdout.
- **--extended-help**: Displays full information about this command and its options.
- **--help**: Displays basic information about this command and its options.
- **--lib-dir <directory>**: Destination directory for the user library archive file (**.a**), the user library makefile, and **public.mk**. If omitted, it defaults to the current directory.
- **--lib-name <filename>**: Name of the user library being created. The user library file name is the user library name with a **lib** prefix and **.a** suffix added. Do not include these in the user library name itself. If the user library name option is omitted, the user library name defaults to the name of the first source file with its extension removed.
- **--log <filename>**: Create a debug log and write to specified file. Also logs debug information to stdout.
- **--no-src**: Allows no sources files to be set in the Makefile. You must add source files manually before compiling.
- **--public-inc-dir <directory>**: Path to a directory that contains C header files (**.h**) that are to be made available (that is, public) to users of the user library. This directory is added to the appropriate variable in **public.mk**. Multiple **--public-inc-dir** options are allowed.
- **--set <name> <value>**: Set the makefile variable called **<name>** to **<value>**. If the variable exists in the managed section of the makefile, **<value>** replaces the default settings. It adds the makefile variable if it does not already exist. Multiple **--set** options are allowed.
- **--silent**: Suppress information about the command's operation normally sent to stdout.
- **--src-dir <directory>**: Search for source files in **<directory>**. Use **.** to look in the current directory. Multiple **--src-dir** options are allowed.

- `--src-files <filenames>`: A list of space-separated source file names added to the makefile. The list of file names is terminated by the next option or the end of the command line. Multiple `--src-files` options are allowed.
- `--src-rdir <directory>`: Same as `--src-dir` option but recursively search for source files in or under `<directory>`. Multiple `--src-rdir` options are allowed and can be freely mixed with `--src-dir` options.
- `--use-lib-dir <directory>`: Path to a dependent user library directory. The user library directory must contain a makefile fragment called **public.mk**. Multiple `--use-lib-dir` options are allowed.
- `--verbose`: Output verbose, and default information about the command's operation to stdout.
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

The **nios2-lib-generate-makefile** command generates a user library makefile (called Makefile). The path to a BSP created by **nios2-bsp-generate-files** is an optional command-line option.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

For more details about this command, use the `--extended-help` option to display comprehensive usage information.

nios2-bsp-editor

Usage

```
nios2-bsp-editor [--extended-help]
  [--fontsize <point size>] [--help]
  [--librarian-factory-path <directory>]
  [--librarian-path <directory>] [--log <filename>]
  [--settings <filename>] [--version]
```

Options

- `--extended-help`: Displays full information about this command and its options.
- `--fontsize <point size>`: The default point size for GUI fonts is 11. Use this option to adjust the point size.
- `--help`: Displays basic information about this command and its options.
- `--librarian-factory-path <directory>`: Comma-separated librarian search path. Use \$ for default factory search path.
- `--librarian-path <directory>`: Comma-separated librarian search path. Use \$ for default search path.
- `--log <filename>`: Create a debug log and write to specified file.
- `--settings <filename>`: File name of an existing BSP settings file to update.
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

The **nios2-bsp-editor** command is a GUI application for creating and editing board support packages for Nios II designs.

For more details about this command, use the `--extended-help` option to display comprehensive usage information.

nios2-app-update-makefile

Usage

```
nios2-app-update-makefile --app-dir <directory>
  [--add-lib-dir <directory>] [--add-src-dir <directory>]
  [--add-src-files <filenames>] [--add-src-rdir <directory>] [--debug]
  [--extended-help] [--force] [--get <name>] [--get-all]
  [--get-asflags] [--get-bsp-dir] [--get-debug-level]
  [--get-defined-symbols] [--get-elf-name] [--get-optimization]
  [--get-undefined-symbols] [--get-user-flags] [--get-warnings]
  [--help] [--list-lib-dir] [--list-src-files] [--lock]
  [--log <filename>] [--no-src] [--remove-lib-dir <directory>]
  [--remove-src-dir <directory>] [--remove-src-files <filenames>]
  [--remove-src-rdir <directory>] [--set <name>]
  [--set-asflags <value>] [--set-bsp-dir <directory>]
  [--set-debug-level <value>] [--set-defined-symbols <value>]
  [--set-elf-name <name>] [--set-optimization <value>]
  [--set-undefined-symbols <value>] [--set-user-flags <value>]
  [--set-warnings <value>] [--show-managed-section] [--show-names]
  [--silent] [--unlock] [--verbose] [--version]
```

Options

- **--app-dir <directory>**: Path to the Application Directory with the generated makefile.
- **--add-lib-dir <directory>**: Add a path to dependent user library directory
- **--add-src-dir <directory>**: Add source files in <directory>. Use . to look in the current directory. Multiple --add-src-dir options are allowed.
- **--add-src-files <filenames>**: A list of space-separated source file names to be added to the makefile. The list of file names is terminated by the next option or the end of the command line. Multiple --src-files options are allowed.
- **--add-src-rdir <directory>**: Same as --add-src-dir option but recursively search for source files in or under <directory>. Multiple --add-src-rdir options are allowed and can be freely mixed with --src-dir options.
- **--debug**: Output debug, exception traces, verbose, and default information about the command's operation to stdout.
- **--extended-help**: Displays full information about this command and its options.
- **--force**: Update the Makefile even if it's locked
- **--get <name>**: Get the values of Makefile variables
- **--get-all**: Get all variables in the managed section of the Makefile
- **--get-asflags**: Get user assembler flags
- **--get-bsp-dir**: Get the BSP generated files directory
- **--get-debug-level**: Get debug level flag
- **--get-defined-symbols**: Get defined symbols flag
- **--get-elf-name**: Get the name of .elf file
- **--get-optimization**: Get optimization flag
- **--get-undefined-symbols**: Get undefined symbols flag

- `--get-user-flags`: Get user flags
- `--get-warnings`: Get warnings flag
- `--help`: Displays basic information about this command and its options.
- `--list-lib-dir`: List all paths to dependent user library directories
- `--list-src-files`: List all source files in the makefile.
- `--lock`: Lock the Makefile to prevent it from being updated
- `--log <filename>`: Create a debug log and write to specified file. Also logs debug information to stdout.
- `--no-src`: Remove all source files in the makefile
- `--remove-lib-dir <directory>`: Remove a path to dependent user library directory
- `--remove-src-dir <directory>`: Remove source files in *<directory>*. Use `.` to look in the current directory. Multiple `--remove-src-dir` options are allowed.
- `--remove-src-files <filenames>`: A list of space-separated source file names to be removed from the makefile. The list of file names is terminated by the next option or the end of the command line. Multiple `--src-files` options are allowed.
- `--remove-src-rdir <directory>`: Same as `--remove-src-dir` option but recursively search for source files in or under *<directory>*. Multiple `--remove-src-rdir` options are allowed and can be freely mixed with `--src-dir` options.
- `--set <name> <value>`: Set the value of a Makefile variable called *<name>*
- `--set-asflags <value>`: Set user assembler flags
- `--set-bsp-dir <directory>`: Set the BSP generated files directory
- `--set-debug-level <value>`: Set debug level flag
- `--set-defined-symbols <value>`: Set defined symbols flag
- `--set-elf-name <name>`: Set the name of `.elf` file
- `--set-optimization <value>`: Set optimization flag
- `--set-undefined-symbols <value>`: Set undefined symbols flag
- `--set-user-flags <value>`: Set user flags
- `--set-warnings <value>`: Set warnings flag
- `--show-managed-section`: Show the managed section in the Makefile
- `--show-names`: Show name of the variables
- `--silent`: Suppress information about the command's operation normally sent to stdout.
- `--unlock`: Unlock the Makefile
- `--verbose`: Output verbose, and default information about the command's operation to stdout.
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

The **nios2-app-update-makefile** command updates an application makefile to add or remove source files.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to `stderr`.

For more details about this command, use the `--extended-help` option to display comprehensive usage information.



The `--add-src-dir`, `--add-src-rdir`, `--remove-src-dir`, and `--remove-src-rdir` options add and remove files found in *<directory>* at the time the command is executed. Files subsequently added to or removed from the directory are not reflected in the makefile.

nios2-lib-update-makefile

Usage

```
nios2-lib-update-makefile --lib-dir <directory>
  [--add-lib-dir <directory>] [--add-public-inc-dir <directory>]
  [--add-src-dir <directory>] [--add-src-files <filenames>]
  [--add-src-rdir <directory>] [--debug] [--extended-help] [--force]
  [--get <name>] [--get-all] [--get-asflags] [--get-bsp-dir]
  [--get-debug-level] [--get-defined-symbols] [--get-lib-name]
  [--get-optimization] [--get-undefined-symbols] [--get-user-flags]
  [--get-warnings] [--help] [--list-lib-dir] [--list-public-inc-dir]
  [--list-src-files] [--lock] [--log <filename>] [--no-src]
  [--remove-lib-dir <directory>] [--remove-public-inc-dir <directory>]
  [--remove-src-dir <directory>] [--remove-src-files <filenames>]
  [--remove-src-rdir <directory>] [--set <name> <value>]
  [--set-asflags <value>] [--set-bsp-dir <directory>]
  [--set-debug-level <value>] [--set-defined-symbols <value>]
  [--set-lib-name <name>] [--set-optimization <value>]
  [--set-undefined-symbols <value>] [--set-user-flags <value>]
  [--set-warnings <value>] [--show-managed-section] [--show-names]
  [--silent] [--unlock] [--verbose] [--version]
```

Options

- `--add-lib-dir <directory>`: Add a path to dependent user library directory
- `--add-public-inc-dir <directory>`: Add a directory that contains C-language header files
- `--add-src-dir <directory>`: Add source files in `<directory>`. Use `.` to look in the current directory. Multiple `--add-src-dir` options are allowed.
- `--add-src-files <filenames>`: A list of space-separated source file names to be added to the makefile. The list of file names is terminated by the next option or the end of the command line. Multiple `--src-files` options are allowed.
- `--add-src-rdir <directory>`: Same as `--add-src-dir` option but recursively search for source files in or under `<directory>`. Multiple `--add-src-rdir` options are allowed and can be freely mixed with `--src-dir` options.
- `--debug`: Output debug, exception traces, verbose, and default information about the command's operation to stdout.
- `--extended-help`: Displays full information about this command and its options.
- `--force`: Update the Makefile even if it is locked
- `--get <name>`: Get the values of Makefile variables
- `--get-all`: Get all variables in the managed section of the Makefile
- `--get-asflags`: Get user assembler flags
- `--get-bsp-dir`: Get the BSP generated files directory
- `--get-debug-level`: Get debug level flag
- `--get-defined-symbols`: Get defined symbols flag
- `--get-lib-name`: Get the name of user library
- `--get-optimization`: Get optimization flag

- `--get-undefined-symbols`: Get undefined symbols flag
- `--get-user-flags`: Get user flags
- `--get-warnings`: Get warnings flag
- `--help`: Displays basic information about this command and its options.
- `--list-lib-dir`: List all paths to dependent user library directories
- `--list-public-inc-dir`: List all public include directories
- `--list-src-files`: List all source files in the makefile.
- `--lock`: Lock the Makefile to prevent it from being updated
- `--log <filename>`: Create a debug log and write to specified file. Also logs debug information to stdout.
- `--no-src`: Remove all source files
- `--remove-lib-dir <directory>`: Remove a path to dependent user library directory
- `--remove-public-inc-dir <directory>`: Remove a include directory
- `--remove-src-dir <directory>`: Remove source files in *<directory>*. Use `.` to look in the current directory. Multiple `--remove-src-dir` options are allowed.
- `--remove-src-files <filenames>`: A list of space-separated source file names to be removed from the makefile. The list of file names is terminated by the next option or the end of the command line. Multiple `--src-files` options are allowed.
- `--remove-src-rdir <directory>`: Same as `--remove-src-dir` option but recursively search for source files in or under *<directory>*. Multiple `--remove-src-rdir` options are allowed and can be freely mixed with `--src-dir` options.
- `--set <name> <value>`: Set the value of a Makefile variable called *<name>*
- `--set-asflags <value>`: Set user assembler flags
- `--set-bsp-dir <directory>`: Set the BSP generated files directory
- `--set-debug-level <value>`: Set debug level flag
- `--set-defined-symbols <value>`: Set defined symbols flag
- `--set-lib-name <name>`: Set the name of user library
- `--set-optimization <value>`: Set optimization flag
- `--set-undefined-symbols <value>`: Set undefined symbols flag
- `--set-user-flags <value>`: Set user flags
- `--set-warnings <value>`: Set warnings flag
- `--show-managed-section`: Show the managed section in the Makefile
- `--show-names`: Show name of the variables
- `--silent`: Suppress information about the command's operation normally sent to stdout.
- `--unlock`: Unlock the Makefile

- `--verbose`: Output verbose, and default information about the command's operation to stdout.
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

The **nios2-lib-update-makefile** command updates a user library makefile to add or remove source files.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

For more details about this command, use the `--extended-help` option to display comprehensive usage information.



The `--add-src-dir`, `--add-src-rdir`, `--remove-src-dir`, and `--remove-src-rdir` options add and remove files found in *<directory>* at the time the command is executed. Files subsequently added to or removed from the directory are not reflected in the makefile.

nios2-swexample-create

Usage

```
nios2-create-swexample [--name] --sopc-dir --type [--list] [--app-dir]
  [--bsp-dir] [--no-app] [--no-bsp] [--elf-name] [--describe]
  [--describeX] [--describeAll] [--search] [--help] [--silent]
  [--version]
```

Options

- **--name**: Specify the name of the software project to create.
- **--sopc-dir**: Specify the hardware design directory. Required.
- **--type**: Specify the software design example template type. Required.
- **--list**: List all valid software design example template types.
- **--app-dir**: Specify the application directory to create. Default: *<sopc-dir>/software_examples/app/<name>*
- **--bsp-dir**: Specify the bsp directory to create. Default: *<sopc-dir>/software_examples/bsp/<bsp-type>*
- **--no-app**: Do not generate the **create-this-app** script
- **--no-bsp**: Do not generate the **create-this-bsp** script
- **--elf-name**: Name of the .elf file to create.
- **--describe**: Describe the software example type specified and exit.
- **--describeX**: Verbosely describe the software example type specified and exit.
- **--describeAll**: Describe all the software example types and exit.
- **--search**: Search for software example templates in the specified directory. Default: *<Nios II EDS install path>/examples/software*
- **--help**: Displays basic information about this command and its options.
- **--silent**: Do not echo commands.
- **--version**: Print the version number of nios2-create-swexample and exit.

Description

This utility creates a template software example for a given SOPC system.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

nios2-elf-insert

Usage

```
nios2-elf-insert <filename> [--cpu_name <cpuName>]  
  [--stderr_dev <stderrDev>] [--stdin_dev <stdinDev>]  
  [--stdout_dev <stdoutDev>] [--sidp <sysidBase>] [--id <sysidHash>]  
  [--timestamp <sysidTime>] [--sof <sofFile>]  
  [--sopcinfo <sopcinfoFile>] [--jar <jarFile>] [--jdi <jdiFile>]  
  [--quartus_project_dir <quartusProjectDir>]  
  [--sopc_system_name <sopcSystemName>]  
  [--profiling_enabled <profilingEnabled>]  
  [--simulation_enabled <simulationEnabled>]
```

Options

- <filename>: the ELF filename
- --cpu_name <cpuName>
- --stderr_dev <stderrDev>
- --stdin_dev <stdinDev>
- --stdout_dev <stdoutDev>
- --sidp <sysidBase>
- --id <sysidHash>
- --timestamp <sysidTime>
- --sof <sofFile>
- --sopcinfo <sopcinfoFile>
- --jar <jarFile>
- --jdi <jdiFile>
- --quartus_project_dir <quartusProjectDir>
- --sopc_system_name <sopcSystemName>
- --profiling_enabled <profilingEnabled>
- --simulation_enabled <simulationEnabled>

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

nios2-elf-query

Usage

```
nios2-elf-query <filename> [--cpu_name] [--stderr_dev] [--stdin_dev]  
  [--stdout_dev] [--sidp] [--id] [--timestamp] [--sof] [--sopcinfo]  
  [--jar] [--jdi] [--quartus_project_dir] [--sopc_system_name]  
  [--profiling_enabled] [--simulation_enabled]
```

Options

- <filename>: the ELF filename
- --cpu_name
- --stderr_dev
- --stdin_dev
- --stdout_dev
- --sidp
- --id
- --timestamp
- --sof
- --sopcinfo
- --jar
- --jdi
- --quartus_project_dir
- --sopc_system_name
- --profiling_enabled
- --simulation_enabled

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

nios2-convert-ide2sbt

Usage

```
nios2-convert-ide2sbt --input-dir --output-dir [--build-config]
  [--help] [--silent] [--version]
```

Options

- **--input-dir**: Specify the input application project directory to create. Required.
- **--output-dir**: Specify the directory to create your output projects. Required.
- **--build-config**: Specify the build configuration type: for example Release, Debug.
- **--help**: Print this message and exit.
- **--silent**: Suppress information about the command's operation normally sent to stdout.
- **--version**: Displays the version of this command and exits.

Description

Convert a Nios II IDE project to a Nios II SBT project.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

nios2-flash-programmer-generate

Usage

```
nios2-flash-programmer-generate [--accept-bad-sysid]
  [--add-bin <fname> <flash-slave-desc> <offset>]
  [--add-elf <fname> <flash-slave-desc> <extra-elf2flash-arguments>]
  [--add-sof <fname> <flash-slave-desc> <offset>
    <extra-sof2flash-arguments>]
  [--cable <cable name>] [--cpu <processor_name>] [--debug]
  [--device <device name>] [--erase-first] [--extended-help]
  --flash-dir <directory> [--go] [--help] [--id <address>]
  [--instance <instance value>] [--log <filename>] [--mmu]
  [--program-flash] [--script-dir <directory>] [--sidp <address>]
  [--silent] --sopcinfo <filename> [--verbose] [--version]
```

Options

- `--accept-bad-sysid`: Continue even if the system identifier (ID) comparison fails.
- `--add-bin <fname> <flash-slave-desc> <offset>`: Specify a binary file to convert and program. The filename, target flash slave descriptor, and target offset amount are required. This option can be used multiple times for SRAM Object Files (**.sof**).
- `--add-elf <fname> <flash-slave-desc> <extra-elf2flash-arguments>`: Specify a **.elf** file to convert and program. The filename and target flash slave descriptor are required. This option can be used multiple times for **.elf** files. `<extra-elf2flash-arguments>` can be any of the following options supported by **elf2flash**:

- `save`
- `sim_optimize`

The following **elf2flash** options have default values computed, but are also supported as `<extra-elf2flash-arguments>` for manual override of those defaults:

- `base`
- `boot`
- `end`
- `reset`

- `--add-sof <fname> <flash-slave-desc> <offset> <extra-sof2flash-arguments>`: Specify a **.sof** file to convert and program. The filename, target flash slave descriptor, and target offset arguments are required. This option can be used multiple times for **.sof** files. `<extra-sof2flash-arguments>` can be any of the following options supported by **sof2flash**:

- `activeparallel`
- `compress`
- `save`
- `timestamp`
- `options`

- `--cable <cable name>`: Specifies which JTAG cable to use (not needed if you only have one cable). Not used without `--program-flash` option.

- `--cpu <processor_name>`: The Nios II processor name from the `.sopcinfo` file. Not required if only one Nios II processor in the system.
- `--debug`: Sends debug information, exception traces, verbose output, and default information about the command's operation, to stdout.
- `--device <device name>`: Specifies in which device you want to look for the Nios II debug core. Device 1 is the device nearest TDI. Not used without `--program-flash` option.
- `--erase-first`: Erase entire flash targets before programming them. Not used without `--program-flash` option.
- `--extended-help`: Displays full information about this command and its options.
- `--flash-dir <directory>`: Path to the directory where the flash files are generated. Use `.` for the current directory. This command overwrites pre-existing files in `<directory>` without warning.
- `--go`: Run processor from reset vector after program.
- `--help`: Displays basic information about this command and its options.
- `--id <address>`: Unique ID code for target system. Not used without `--program-flash` option.
- `--instance <instance value>`: Specifies the INSTANCE value of the debug core (not needed if there is exactly one on the chain). Not used without `--program-flash` option.
- `--log <filename>`: Creates a debug log and write to specified file. Also logs debug information to stdout.
- `--mmu`: Specifies if the processor with the corresponding INSTANCE value has an MMU (not needed if there is exactly one processor in the system). Not used without `--program-flash` option.
- `--program-flash`: Providing this flag causes calls to **nios2-flash-programmer** to be generated and executed. This results in flash targets being programmed.
- `--script-dir <directory>`: Path to the directory where a shell script of this tool's executed command lines is generated. This script can be used in place of this **nios2-flash-programmer-generate** command. Use `.` for the current directory. This command overwrites pre-existing files in `<directory>` without warning.
- `--sidp <address>`: Base address of system ID peripheral on the target. Not used without `--program-flash` option.
- `--silent`: Suppresses information about the command's operation normally sent to stdout.
- `--sopcinfo <filename>`: The `.sopcinfo` file.
- `--verbose`: Sends verbose output, and default information about the command's operation, to stdout.
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

The **nios2-flash-programmer-generate** command converts multiple files to a **.flash** in Motorola S-record format, and programs them to the designated target flash devices (**--program-flash**). This is a convenience utility that manages calls to the following command line utilities

- **bin2flash**
- **elf2flash**
- **sof2flash**
- **nios2-flash-programmer**

This utility also generates a script that captures the sequence of conversion and flash programmer commands.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to stderr.

Example

```
nios2-flash-programmer-generate --sopcinfo=C:\my_design.sopcinfo \  
  --flash-dir=. \  
  --add-sof C:\my_design\test.sof 0x0C000000 memory_0 compress save \  
  --add-elf C:\my_app\my_app.elf 0x08000000 memory_0 \  
  --program-flash
```

nios2-c2h-generate-makefile

Usage

```
nios2-c2h-generate-makefile --sopc=<sopc>  
  --app-dir=<dir>  
  --accelerator=<function>  
  --enable_quartus=<0/1>  
  --analyze_only=<0/1>  
  --use_existing_accelerators=<0/1>
```

Options

- **--sopc**: The path to the **.sopcinfo** file.
- **--app-dir**: Directory to place the application Makefile and executable file. If omitted, it defaults to the current directory.
- **--accelerator**: Specifies a function to be accelerated.
- **--enable_quartus**: Building the application compiles the associated Quartus II project. Defaults to 0.
- **--analyze_only**: Disables accelerator hardware generation, system hardware generation, and Quartus II compilation for all accelerators in the application. Building the project with this option only updates the report files. Defaults to 0.
- **--use_existing_accelerators**: Disables all hardware generation steps. The build behaves as if **c2h.mk** did not exist, with the exception of possible accelerator linking as specified in the **--accelerator** option. Defaults to 0.

Description

The **nios2-c2h-generate-makefile** command creates a C2H makefile fragment, **c2h.mk**, that specifies all accelerators and accelerator options for an application.

This command creates a new **c2h.mk** each time it is called, overwriting the existing **c2h.mk**

The **--accelerator** argument specifies a function to be accelerated. This argument accepts up to four comma-separated values:

- Target function name.
- Target function file.
- Link hardware accelerator instead of original software. 1 or 0. Defaults to 1.
- Flush data cache before each call. 1 or 0. Defaults to 1.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to **stderr**.



The **nios2-c2h-generate-makefile** script is available to support pre-existing command-line C2H projects. Create new C2H projects using the Nios II IDE.

Example

```
nios2-c2h-generate-makefile \  
  --sopc=../../NiosII_stratix_ls40_standard.sopcinfo \  
  --app_dir=./ \  
  --accelerator=filter,filter.c \  
  --accelerator=xmath,../../xmath.c,1,0 \  
  --use_existing_accelerators
```

nios2-bsp

Usage

```
nios2-bsp <bsp-type> <bsp-dir> [<sopc>] [<override>]...
```

Options

- <bsp-type>: hal or ucossii.
- <bsp-dir>: Path to the BSP directory.
- <sopc>: The path to the .sopcinfo file or its directory.
- <override>: Options to override defaults.

Description

The **nios2-bsp** script calls **nios2-bsp-create-settings** or **nios2-bsp-update-settings** to create or update a BSP settings file, and the **nios2-bsp-generate-files** command to create the BSP files. The Nios II Embedded Design Suite (EDS) supports the following BSP types:

- hal
- ucossii

BSP type names are case-insensitive.

This utility produces a BSP of <bsp-type> in <bsp-dir>. If the BSP does not exist, it is created. If the BSP already exists, it is updated to be consistent with the associated hardware system.

The default Tcl script is used to set the following system-dependent settings:

- stdio character device
- System timer device
- Default linker memory
- Boot loader status (enabled or disabled)

If the BSP already exists, **nios2-bsp** overwrites these system-dependent settings.

The default Tcl script is installed at <Nios II EDS install path>/sdk2/bin/**bsp-set-defaults.tcl**

When creating a new BSP, this utility runs **nios2-bsp-create-settings**, which creates **settings.bsp** in <bsp-dir>.

When updating an existing BSP, this utility runs **nios2-bsp-update-settings**, which updates **settings.bsp** in <bsp-dir>.

After creating or updating the **settings.bsp** file, this utility runs **nios2-bsp-generate-files**, which generates files in <bsp-dir>

Required arguments:

- **<bsp-type>**: Specifies the type of BSP. This argument is ignored when updating a BSP. This argument is case-insensitive. The **nios2-bsp** script supports the following values of **<bsp-type>**:
 - hal
 - ucosii
- **<bsp-dir>**: Path to the BSP directory. Use "." to specify the current directory.

Optional arguments:

- **<sopc>**: The path name of the **.sopcinfo** file. Alternatively, specify a directory containing a **.sopcinfo** file. In the latter case, the tool finds a file with the extension **.sopcinfo**. This argument is ignored when updating a BSP. If you omit this argument, it defaults to the current directory.
- **<override>**: Options to override defaults. The **nios2-bsp** script passes most overrides to **nios2-bsp-create-settings** or **nios2-bsp-update-settings**. It also passes the **--silent**, **--verbose**, **--debug**, and **--log** options to **nios2-bsp-generate-files**.

nios2-bsp passes the following overrides to the default Tcl script:

- **--default_stdio <device>|none|DONT_CHANGE**
Specifies stdio device.
- **--default_sys_timer <device>|none|DONT_CHANGE**
Specifies system timer device.
- **--default_memory_regions DONT_CHANGE**
Suppresses creation of new default memory regions when updating a BSP. Do not use this option when creating a new BSP.
- **--default_sections_mapping <region>|DONT_CHANGE**
Specifies the memory region for the default sections.
- **--use_bootloader 0|1|DONT_CHANGE**
Specifies whether a boot loader is required.

On a preexisting BSP, the value **DONT_CHANGE** prevents associated settings from changing their current value.



The "--" prefix is stripped when the option is passed to the underlying utility.

If no command-line arguments are specified, this command returns an exit value of 1 and sends a help message to **stderr**.

nios2-bsp-console

Usage

```
nios2-bsp-console [--cmd <tcl> <command>] [--extended-help] [--gui]  
                  [--help] [--script <filename>] [--version]
```

Options

- `--cmd <tcl> <command>`: Runs the specified Tcl command. Multiple `--cmd` options are allowed. Available Tcl commands are listed in [“Tcl Commands for BSP Settings” on page 15-79](#).
- `--extended-help`: Displays full information about this command and its options. Lists Altera BSP Tcl command help for the `--cmd` and `--script` options
- `--gui`: This option opens a Tcl console for creating, editing, and generating Altera BSPs.
- `--help`: Displays basic information about this command and its options.
- `--script <filename>`: Run the specified Tcl script with optional arguments. Multiple `--script` options are allowed. Available Tcl commands are listed in [“Tcl Commands for BSP Settings” on page 15-79](#).
- `--version`: Displays the version of this command and exits with a zero exit status.

Description

When invoked with no arguments, **nios2-bsp-console** starts an interactive command-line Tcl interpreter for creating, editing, and generating Altera BSPs. Available Tcl commands are listed in [“Tcl Commands for BSP Settings” on page 15-79](#).

Nios II Design Example Scripts

The Nios II SBT includes scripts that allow you to create sample BSPs and applications. This section describes each script and its location in the design example directory structure. Each hardware design example in the Nios II EDS includes a **software_examples** directory with **app** and **bsp** subdirectories.

The **bsp** subdirectory contains a variety of example BSPs. Table 15-3 lists all potential BSP examples provided in the **bsp** directory. The **bsp** directory for each hardware example only includes BSP examples supported by the associated hardware example.

Table 15-3. BSP Examples

Example BSP (1)	Summary
hal_reduced_footprint	Hardware abstraction layer (HAL) BSP configured to minimize memory footprint
hal_default	HAL BSP configured with all defaults
hal_zipfs	HAL BSP configured with the Altera® read-only zip file system
ucosii_net	MicroC/OS-II BSP configured with networking
ucosii_net_zipfs	MicroC/OS-II BSP configured with networking and the Altera read-only zip file system
ucosii_net_tse	MicroC/OS-II BSP configured with networking support for the Altera triple-speed Ethernet media access control (MAC)
ucosii_net_tse_zipfs	MicroC/OS-II BSP configured with networking support for the Altera triple-speed Ethernet MAC and the Altera read-only zip file system
ucosii_default	MicroC/OS-II BSP configured with all defaults

Note to Table 15-3:

(1) Some BSP examples might not be available on some hardware examples.

The **app** subdirectory contains a separate subdirectory for each software example supported by the hardware example, as listed in Table 15-4.

Table 15-4. Application Examples (1)

Application Name	Summary
Hello World	Prints "Hello from Nios II"
Board Diagnostics	Tests peripherals on the development boards
Count Binary	Displays a running count of 0x00 to 0xff
Hello Freestanding	Prints "Hello from Nios II" from a free-standing application
Hello MicroC/OS-II	Prints "Hello from Nios II" using the MicroC/OS-II RTOS
Hello World Small	Prints "Hello from Nios II" from a small footprint program
Memory Test	Runs diagnostic tests on both volatile and flash memory
Memory Test Small	Runs diagnostic tests on volatile memory from a small footprint
Simple Socket Server	Runs a TCP/IP socket server
Web Server	Runs a web server from a file system in flash memory
Zip File System	Reads from a file system in flash memory

Note to Table 15-4:

(1) Some application examples might not be available on some hardware examples, depending on BSP support.

create-this-bsp

Each BSP subdirectory contains a **create-this-bsp** script. The **create-this-bsp** script calls the **nios2-bsp** script to create a BSP in the current directory. The **create-this-bsp** script has a relative path to the directory containing the **.sopcinfo** file. The **.sopcinfo** file resides two directory levels above the directory containing the **create-this-bsp** script.

The **create-this-bsp** script takes no command-line arguments. Your current directory must be the same directory as the **create-this-bsp** script. The exit value is zero on success and one on error.

create-this-app

Each application subdirectory contains a **create-this-app** script. The **create-this-app** script copies the C/C++ application source code to the current directory, runs **nios2-app-generate-makefile** to create a makefile (named **Makefile**), and then runs **make** to build the Executable and Linking Format File (**.elf**) for your application. Each **create-this-app** script uses a particular example BSP. For further information, refer to the script to determine the associated example BSP. If the BSP does not exist when **create-this-app** runs, **create-this-app** calls the associated **create-this-bsp** script to create the BSP.

The **create-this-app** script takes no command-line arguments. Your current directory must be the same directory as the **create-this-app** script. The exit value is zero on success and one on error.

Finding create-this-app and create-this-bsp

The **create-this-app** and **create-this-bsp** scripts are installed with your Nios II design examples. You can easily find them from the following information:

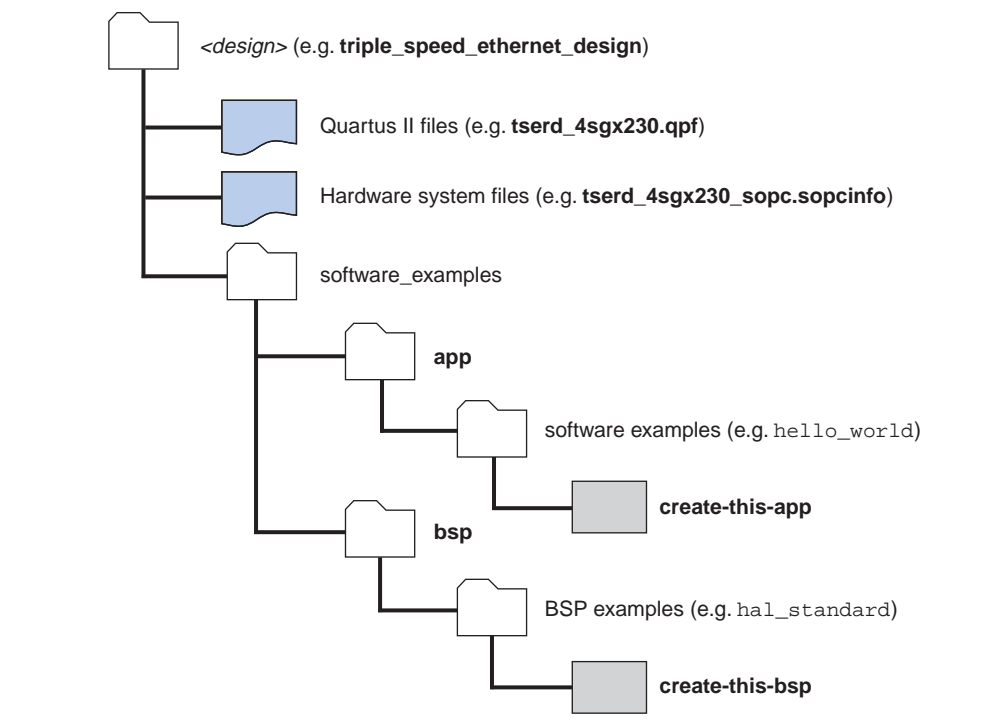
- Where the Nios II EDS is installed
- Which Altera development board you are using
- Which HDL you are using
- Which Nios II hardware design example you are using
- The name of the Nios II software example

The **create-this-app** script for each software design example is in `<Nios II EDS install path>/examples/<HDL>/niosII_<board type>/<design name>/software_examples/app/<example name>`. For example, the **create-this-app** script for the **Hello World** software example running on the triple-speed ethernet design example for the Stratix® IV GX FPGA Development Kit might be located in `c:/altera/100/nios2eds/examples/verilog/niosII_stratixIV_4sgx230/triple_speed_ethernet_design/software_examples/app/hello_world`.

Similarly, the **create-this-bsp** script for each software design example is in `<Nios II EDS install path>/examples/<HDL>/niosII_<board type>/<design name>/software_examples/bsp/<BSP_type>`. For example, the **create-this-bsp** script for the basic HAL BSP to run on the triple-speed ethernet design example for the Stratix IV GX FPGA Development Kit might be located in `c:/altera/100/nios2eds/examples/verilog/niosII_stratixIV_4sgx230/triple_speed_ethernet_design/software_examples/bsp/hal_default`.

Figure 15-1 shows the directory structure under each hardware design example.

Figure 15-1. Software Design Example Directory Structure



Settings Managed by the Software Build Tools

Settings are central to how you create and work with BSPs, software packages, and device drivers. You control the characteristics of your project by controlling the settings. The settings determine things like whether or not an operating system is supported, and the device drivers and other packages that are included.

Every example in the *Getting Started from the Command Line* and *Nios II Software Build Tools* chapters of the *Nios II Software Developer's Handbook* involves specifying or manipulating settings. Sometimes these settings are specified automatically, by scripts such as **create-this-bsp**, and sometimes explicitly, with Tcl commands. Either way, settings are always involved.

This section contains a complete list of available settings for BSPs and for Altera-supported device drivers and software packages. It does not include settings for device drivers or software packages furnished by Altera partners or other third parties. If you are using a third-party driver or component, refer to the supplier's documentation.

Settings used in the Nios II SBT are organized hierarchically, for logical grouping and to avoid name conflicts. Each setting's position in the hierarchy is indicated by one or more prefixes. A prefix is an identifier followed by a dot (.). For example, `hal.enable_c_plus_plus` is a hardware abstraction layer (HAL) setting, while `ucosii.event_flag.os_flag_accept_en` is a member of the event flag subgroup of MicroC/OS-II settings.

Setting names are case-insensitive.

Overview of BSP Settings

A BSP setting consists of a name/value pair.

The format in which you specify the setting value depends on the setting type. Several settings types are supported. Table 15-5 shows the allowed formats for each setting type.

Table 15-5. Setting Formats

Setting Type	Format When Setting	Format When Getting
boolean	0/1 or false/true	0/1
number	0x prefix for hexadecimal or no prefix for a decimal number	decimal
string	Quoted string Use "none" to set empty string. In the SBT command line, to specify a string value with embedded spaces, surround the string with a backslash-double-quote sequence (\"). For example: <code>--set APP_INCLUDE_DIRS \"lcd board\"</code>	Quoted string

There are several contexts for BSP settings, as shown in [Table 15-6](#).

Table 15-6. BSP Setting Contexts

Setting Context	Description
Altera HAL	Settings available with the Altera HAL BSP or any BSP based on it (for example, Micrium MicroC/OS-II).
Micrium MicroC/OS-II	Settings available if using the Micrium MicroC/OS-II BSP. All Altera HAL BSP settings are also available because MicroC/OS-II is based on the Altera HAL BSP.
Altera BSP Makefile Generator	Settings available if using the Altera BSP makefile generator (generates the Makefile and public.mk files). These settings control the contents of makefile variables. This generator is always present in Altera HAL BSPs or any BSPs based on the Altera HAL.
Altera BSP Linker Script Generator	Settings available if using the Altera BSP linker script generator (generates the linker.x and linker.h files). This generator is always present in Altera HAL BSPs or any BSPs based on the Altera HAL.

Do not confuse BSP settings with BSP Tcl commands. This section covers BSP settings, including their types, meanings, and legal values. The Tcl commands, which include tools for manipulating the settings, are covered in [“Tcl Commands for BSP Settings” on page 15-79](#).

Overview of Component and Driver Settings

The Nios II EDS includes a number of standard software packages and device drivers. All of the software packages, and several drivers, have settings that you can manipulate when creating a BSP. This section lists the packages and drivers that have settings.

You can enable a software package or driver in the Nios II BSP Editor. In the SBT command line flow, use the `enable_sw_package` Tcl command, described in [“Tcl Commands for BSP Settings” on page 15-79](#).

Altera Host-Based File System Settings

The Altera host-based file system has one setting. If the Altera host-based file system is enabled, you must debug (not run) applications based on the BSP for the host-based file system to function. The host-based file system relies on the GNU debugger running on the host to provide host-based file operations.

The host-based file system enables debugging information in your project, by setting `APP_CFLAGS_OPTIMIZATION` to `-g` in the makefile.

To include the host-based file system in your BSP, enable the `altera_hostfs` software package.

Altera Read-Only Zip File System Settings

The Altera read-only zip file system has several settings. If the read-only zip file system is enabled, it adds `-DUSE_RO_ZIPFS` to `ALT_CPPFLAGS` in **public.mk**.

To include the read-only zip file system in your BSP, enable the `altera_ro_zipfs` software package.

Altera NicheStack TCP/IP - Nios II Edition Stack Settings

The Altera NicheStack® TCP/IP Network Stack - Nios II Edition has several settings. The stack is only available in MicroC/OS-II BSPs. If the NicheStack TCP/IP stack is enabled, it adds `-DALT_INICHE` to `ALT_CPPFLAGS` in **public.mk**.

To include the NicheStack TCP/IP networking stack in your BSP, enable the `altera_iniche` software package.

Altera Avalon-MM JTAG UART Driver Settings

The Altera Avalon Memory-Mapped® (Avalon-MM) JTAG UART driver settings are available if the `altera_avalon_jtag_uart` driver is present. By default, this driver is used if your hardware system has an `altera_avalon_jtag_uart` module connected to it.

Altera Avalon-MM UART Driver Settings

The Altera Avalon-MM UART driver settings are available if the `altera_avalon_uart` driver is present. By default, this driver is used if your hardware system has an `altera_avalon_uart` module connected to it.

Settings Reference

This section lists all settings for BSPs, software packages, and device drivers.

hal.enable_instruction_related_exceptions_api

Identifier:	none
Type:	Boolean definition
Default Value:	false
Destination File:	none
Description:	Enables application program interface (API) for registering handlers to service instruction-related exceptions. These exception types can be generated if various processor options are enabled, such as the memory management unit (MMU), memory protection unit (MPU), or other advanced exception types. Enabling this setting increases the size of the exception entry code.
Restrictions:	none

hal.max_file_descriptors

Identifier:	none
Type:	Decimal number
Default Value:	32
Destination File:	none
Description:	Determines the number of file descriptors statically allocated.
Restrictions:	If <code>hal.enable_lightweight_device_driver_api</code> is true, there are no file descriptors so this setting is ignored. If <code>hal.enable_lightweight_device_driver_api</code> is false, this setting must be at least 4 because HAL needs a file descriptor for <code>/dev/null</code> , <code>/dev/stdin</code> , <code>/dev/stdout</code> , and <code>/dev/stderr</code> . This setting defines the value of <code>ALT_MAX_FD</code> in system.h .

hal.exclude_default_exception

Identifier:	ALT_EXCLUDE_DEFAULT_EXCEPTION
Type:	Boolean definition
Default Value:	false
Destination File:	system.h
Description:	Excludes default exception vector. If true, this setting defines the macro <code>ALT_EXCLUDE_DEFAULT_EXCEPTION</code> in system.h .
Restrictions:	none

hal.sys_clk_timer

Identifier:	none
Type:	Unquoted string
Default Value:	none
Destination File:	none
Description:	Slave descriptor of the system clock timer device. This device provides a periodic interrupt (“tick”) and is typically required for RTOS use. This setting defines the value of ALT_SYS_CLK in system.h .
Restrictions:	none

hal.timestamp_timer

Identifier:	none
Type:	Unquoted string
Default Value:	none
Destination File:	none
Description:	Slave descriptor of timestamp timer device. This device is used by Altera HAL timestamp drivers for high-resolution time measurement. This setting defines the value of ALT_TIMESTAMP_CLK in system.h .
Restrictions:	none

ucosii.os_max_tasks

Identifier:	OS_MAX_TASKS
Type:	Decimal number
Default Value:	10
Destination File:	system.h
Description:	Maximum number of tasks
Restrictions:	none

ucosii.os_lowest_prio

Identifier:	OS_LOWEST_PRIO
Type:	Decimal number
Default Value:	20
Destination File:	system.h
Description:	Lowest assignable priority
Restrictions:	none

ucosii.os_thread_safe_newlib

Identifier:	OS_THREAD_SAFE_NEWLIB
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Thread safe C library
Restrictions:	none

ucosii.miscellaneous.os_arg_chk_en

Identifier:	OS_ARG_CHK_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Enable argument checking
Restrictions:	none

ucosii.miscellaneous.os_cpu_hooks_en

Identifier:	OS_CPU_HOOKS_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Enable MicroC/OS-II hooks
Restrictions:	none

ucosii.miscellaneous.os_debug_en

Identifier:	OS_DEBUG_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Enable debug variables
Restrictions:	none

ucosii.miscellaneous.os_sched_lock_en

Identifier:	OS_SCHED_LOCK_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSSchedLock() and OSSchedUnlock()
Restrictions:	none

ucosii.miscellaneous.os_task_stat_en

Identifier: OS_TASK_STAT_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Enable statistics task
Restrictions: none

ucosii.miscellaneous.os_task_stat_stk_chk_en

Identifier: OS_TASK_STAT_STK_CHK_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Check task stacks from statistics task
Restrictions: none

ucosii.miscellaneous.os_tick_step_en

Identifier: OS_TICK_STEP_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Enable tick stepping feature for uCOS-View
Restrictions: none

ucosii.miscellaneous.os_event_name_size

Identifier: OS_EVENT_NAME_SIZE
Type: Decimal number
Default Value: 32
Destination File: **system.h**
Description: Size of name of Event Control Block groups
Restrictions: none

ucosii.miscellaneous.os_max_events

Identifier: OS_MAX_EVENTS
Type: Decimal number
Default Value: 60
Destination File: **system.h**
Description: Maximum number of event control blocks
Restrictions: none

ucosii.miscellaneous.os_task_idle_stk_size

Identifier: OS_TASK_IDLE_STK_SIZE
Type: Decimal number
Default Value: 512
Destination File: **system.h**
Description: Idle task stack size
Restrictions: none

ucosii.miscellaneous.os_task_stat_stk_size

Identifier: OS_TASK_STAT_STK_SIZE
Type: Decimal number
Default Value: 512
Destination File: **system.h**
Description: Statistics task stack size
Restrictions: none

ucosii.task.os_task_change_prio_en

Identifier: OS_TASK_CHANGE_PRIO_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTaskChangePrio()
Restrictions: none

ucosii.task.os_task_create_en

Identifier: OS_TASK_CREATE_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTaskCreate()
Restrictions: none

ucosii.task.os_task_create_ext_en

Identifier: OS_TASK_CREATE_EXT_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTaskCreateExt()
Restrictions: none

ucosii.task.os_task_del_en

Identifier: OS_TASK_DEL_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTaskDel()
Restrictions: none

ucosii.task.os_task_name_size

Identifier: OS_TASK_NAME_SIZE
Type: Decimal number
Default Value: 32
Destination File: **system.h**
Description: Size of task name
Restrictions: none

ucosii.task.os_task_profile_en

Identifier: OS_TASK_PROFILE_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include data structure for run-time task profiling
Restrictions: none

ucosii.task.os_task_query_en

Identifier: OS_TASK_QUERY_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTaskQuery
Restrictions: none

ucosii.task.os_task_suspend_en

Identifier: OS_TASK_SUSPEND_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTaskSuspend() and OSTaskResume()
Restrictions: none

ucosii.task.os_task_sw_hook_en

Identifier: OS_TASK_SW_HOOK_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTaskSwHook()
Restrictions: none

ucosii.time.os_time_tick_hook_en

Identifier: OS_TIME_TICK_HOOK_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTimeTickHook()
Restrictions: none

ucosii.time.os_time_dly_resume_en

Identifier: OS_TIME_DLY_RESUME_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTimeDlyResume()
Restrictions: none

ucosii.time.os_time_dly_hmsm_en

Identifier: OS_TIME_DLY_HMSM_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTimeDlyHMSM()
Restrictions: none

ucosii.time.os_time_get_set_en

Identifier: OS_TIME_GET_SET_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSTimeGet and OSTimeSet()
Restrictions: none

ucosii.os_flag_en

Identifier:	OS_FLAG_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Enable code for Event Flags. This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.event_flag.os_flag_wait_clr_en

Identifier:	OS_FLAG_WAIT_CLR_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for Wait on Clear Event Flags. This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.event_flag.os_flag_accept_en

Identifier:	OS_FLAG_ACCEPT_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSFlagAccept(). This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.event_flag.os_flag_del_en

Identifier:	OS_FLAG_DEL_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSFlagDel(). This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.event_flag.os_flag_query_en

Identifier:	OS_FLAG_QUERY_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSFlagQuery(). This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.event_flag.os_flag_name_size

Identifier:	OS_FLAG_NAME_SIZE
Type:	Decimal number
Default Value:	32
Destination File:	system.h
Description:	Size of name of Event Flags group. CAUTION: This is required by the HAL and many Altera device drivers; use caution in reducing this value.
Restrictions:	none

ucosii.event_flag.os_flags_nbits

Identifier:	OS_FLAGS_NBITS
Type:	Decimal number
Default Value:	16
Destination File:	system.h
Description:	Event Flag bits (8,16,32). CAUTION: This is required by the HAL and many Altera device drivers; use caution in changing this value.
Restrictions:	none

ucosii.event_flag.os_max_flags

Identifier:	OS_MAX_FLAGS
Type:	Decimal number
Default Value:	20
Destination File:	system.h
Description:	Maximum number of Event Flags groups. CAUTION: This is required by the HAL and many Altera device drivers; use caution in reducing this value.
Restrictions:	none

ucosii.os_mutex_en

Identifier: OS_MUTEX_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Enable code for Mutex Semaphores
Restrictions: none

ucosii.mutex.os_mutex_accept_en

Identifier: OS_MUTEX_ACCEPT_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSMutexAccept()
Restrictions: none

ucosii.mutex.os_mutex_del_en

Identifier: OS_MUTEX_DEL_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSMutexDel()
Restrictions: none

ucosii.mutex.os_mutex_query_en

Identifier: OS_MUTEX_QUERY_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSMutexQuery
Restrictions: none

ucosii.os_sem_en

Identifier:	OS_SEM_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Enable code for semaphores. This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.semaphore.os_sem_accept_en

Identifier:	OS_SEM_ACCEPT_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSSemAccept(). This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.semaphore.os_sem_set_en

Identifier:	OS_SEM_SET_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSSemSet(). This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.semaphore.os_sem_del_en

Identifier:	OS_SEM_DEL_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSSemDel(). This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.semaphore.os_sem_query_en

Identifier:	OS_SEM_QUERY_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSSemQuery(). This setting is enabled by default in MicroC-OS/II BSPs, because it is required for correct functioning of Altera device drivers and the HAL in a multithreaded environment. Avoid disabling it.
Restrictions:	none

ucosii.os_mbox_en

Identifier:	OS_MBOX_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Enable code for mailboxes
Restrictions:	none

ucosii.mailbox.os_mbox_accept_en

Identifier:	OS_MBOX_ACCEPT_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSMboxAccept()
Restrictions:	none

ucosii.mailbox.os_mbox_del_en

Identifier:	OS_MBOX_DEL_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSMboxDel()
Restrictions:	none

ucosii.mailbox.os_mbox_post_en

Identifier: OS_MBOX_POST_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSMboxPost()
Restrictions: none

ucosii.mailbox.os_mbox_post_opt_en

Identifier: OS_MBOX_POST_OPT_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSMboxPostOpt()
Restrictions: none

ucosii.mailbox.os_mbox_query_en

Identifier: OS_MBOX_QUERY_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSMboxQuery()
Restrictions: none

ucosii.os_q_en

Identifier: OS_Q_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Enable code for Queues
Restrictions: none

ucosii.queue.os_q_accept_en

Identifier: OS_Q_ACCEPT_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSQAccept()
Restrictions: none

ucosii.queue.os_q_del_en

Identifier: OS_Q_DEL_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSQDel()
Restrictions: none

ucosii.queue.os_q_flush_en

Identifier: OS_Q_FLUSH_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSQFlush()
Restrictions: none

ucosii.queue.os_q_post_en

Identifier: OS_Q_POST_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code of OSQFlush()
Restrictions: none

ucosii.queue.os_q_post_front_en

Identifier: OS_Q_POST_FRONT_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSQPostFront()
Restrictions: none

ucosii.queue.os_q_post_opt_en

Identifier: OS_Q_POST_OPT_EN
Type: Boolean assignment
Default Value: 1
Destination File: **system.h**
Description: Include code for OSQPostOpt()
Restrictions: none

ucosii.queue.os_q_query_en

Identifier:	OS_Q_QUERY_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSQQuery()
Restrictions:	none

ucosii.queue.os_max_qs

Identifier:	OS_MAX_QS
Type:	Decimal number
Default Value:	20
Destination File:	system.h
Description:	Maximum number of Queue Control Blocks
Restrictions:	none

ucosii.os_mem_en

Identifier:	OS_MEM_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Enable code for memory management
Restrictions:	none

ucosii.memory.os_mem_query_en

Identifier:	OS_MEM_QUERY_EN
Type:	Boolean assignment
Default Value:	1
Destination File:	system.h
Description:	Include code for OSMemQuery()
Restrictions:	none

ucosii.memory.os_mem_name_size

Identifier:	OS_MEM_NAME_SIZE
Type:	Decimal number
Default Value:	32
Destination File:	system.h
Description:	Size of memory partition name
Restrictions:	none

ucosii.memory.os_max_mem_part

Identifier: OS_MAX_MEM_PART
Type: Decimal number
Default Value: 60
Destination File: **system.h**
Description: Maximum number of memory partitions
Restrictions: none

ucosii.os_tmr_en

Identifier: OS_TMR_EN
Type: Boolean assignment
Default Value: 0
Destination File: **system.h**
Description: Enable code for timers
Restrictions: none

ucosii.timer.os_task_tmr_stk_size

Identifier: OS_TASK_TMR_STK_SIZE
Type: Decimal number
Default Value: 512
Destination File: **system.h**
Description: Stack size for timer task
Restrictions: none

ucosii.timer.os_task_tmr_prio

Identifier: OS_TASK_TMR_PRIO
Type: Decimal number
Default Value: 2
Destination File: **system.h**
Description: Priority of timer task (0=highest)
Restrictions: none

ucosii.timer.os_tmr_cfg_max

Identifier: OS_TMR_CFG_MAX
Type: Decimal number
Default Value: 16
Destination File: **system.h**
Description: Maximum number of timers
Restrictions: none

ucosii.timer.os_tmr_cfg_name_size

Identifier:	OS_TMR_CFG_NAME_SIZE
Type:	Decimal number
Default Value:	16
Destination File:	system.h
Description:	Size of timer name
Restrictions:	none

ucosii.timer.os_tmr_cfg_ticks_per_sec

Identifier:	OS_TMR_CFG_TICKS_PER_SEC
Type:	Decimal number
Default Value:	10
Destination File:	system.h
Description:	Rate at which timer management task runs (Hz)
Restrictions:	none

ucosii.timer.os_tmr_cfg_wheel_size

Identifier:	OS_TMR_CFG_WHEEL_SIZE
Type:	Decimal number
Default Value:	2
Destination File:	system.h
Description:	Size of timer wheel (number of spokes)
Restrictions:	none

altera_avalon_uart_driver.enable_small_driver

Identifier:	ALTERA_AVALON_UART_SMALL
Type:	Boolean definition
Default Value:	false
Destination File:	public.mk
Description:	Small-footprint (polled mode) driver
Restrictions:	none

altera_avalon_uart_driver.enable_ioctl

Identifier:	ALTERA_AVALON_UART_USE_IOCTL
Type:	Boolean definition
Default Value:	false
Destination File:	public.mk
Description:	Enable driver ioctl() support. This feature is not compatible with the small driver; ioctl() support is not compiled if either the UART <code>enable_small_driver</code> or the HAL <code>enable_reduced_device_drivers</code> setting is enabled.
Restrictions:	none

altera_avalon_jtag_uart_driver.enable_small_driver

Identifier:	ALTERA_AVALON_JTAG_UART_SMALL
Type:	Boolean definition
Default Value:	false
Destination File:	public.mk
Description:	Small-footprint (polled mode) driver
Restrictions:	none

altera_hostfs.hostfs_name

Identifier:	ALTERA_HOSTFS_NAME
Type:	Quoted string
Default Value:	/mnt/host
Destination File:	system.h
Description:	Mount point
Restrictions:	none

altera_iniche.iniche_default_if

Identifier:	INICHE_DEFAULT_IF
Type:	Quoted string
Default Value:	NOT_USED
Destination File:	system.h
Description:	Deprecated setting: Default media access control (MAC) interface. This setting is used in some legacy Altera networking examples. It is not needed in new projects. If this setting appears in an existing project, Altera recommends that you make any necessary changes to remove it. This setting might be removed in a future release.
Restrictions:	none

altera_iniche.enable_dhcp_client

Identifier:	DHCP_CLIENT
Type:	Boolean definition
Default Value:	true
Destination File:	system.h
Description:	Use dynamic host configuration protocol (DHCP) to automatically assign Internet protocol (IP) address
Restrictions:	none

altera_iniche.enable_ip_fragments

Identifier:	IP_FRAGMENTS
Type:	Boolean definition
Default Value:	true
Destination File:	system.h
Description:	Reassemble IP packet fragments
Restrictions:	none

altera_iniche.enable_include_tcp

Identifier:	INCLUDE_TCP
Type:	Boolean definition
Default Value:	true
Destination File:	system.h
Description:	Enable Transmission Control Protocol (TCP)
Restrictions:	none

altera_iniche.enable_tcp_zerocopy

Identifier:	TCP_ZEROCOPY
Type:	Boolean definition
Default Value:	false
Destination File:	system.h
Description:	Use TCP zero-copy
Restrictions:	none

altera_iniche.enable_net_stats

Identifier: NET_STATS
Type: Boolean definition
Default Value: false
Destination File: **system.h**
Description: Enable statistics
Restrictions: none

altera_ro_zipfs.ro_zipfs_name

Identifier: ALTERA_RO_ZIPFS_NAME
Type: Quoted string
Default Value: /mnt/rozipfs
Destination File: **system.h**
Description: Mount point
Restrictions: none

altera_ro_zipfs.ro_zipfs_offset

Identifier: ALTERA_RO_ZIPFS_OFFSET
Type: Hexadecimal number
Default Value: 0x100000
Destination File: **system.h**
Description: Offset of file system from base of flash
Restrictions: none

altera_ro_zipfs.ro_zipfs_base

Identifier: ALTERA_RO_ZIPFS_BASE
Type: Hexadecimal number
Default Value: 0x0
Destination File: **system.h**
Description: Base address of flash memory device
Restrictions: none

hal.linker.allow_code_at_reset

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	none
Description:	Indicates if initialization code is allowed at the reset address. If true, defines the macro ALT_ALLOW_CODE_AT_RESET in linker.h .
Restrictions:	This setting is typically false if an external bootloader (e.g. flash bootloader) is present.

hal.linker.enable_alt_load

Identifier:	none
Type:	Boolean assignment
Default Value:	1
Destination File:	none
Description:	Enables the alt_load() facility. The alt_load() facility copies sections from the .text memory into RAM. If true, this setting sets up the VMA/LMA (virtual memory address/low memory address) of sections in linker.x to allow them to be loaded into the .text memory.
Restrictions:	This setting is typically false if an external bootloader (e.g. flash bootloader) is present.

hal.linker.enable_alt_load_copy_exceptions

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	none
Description:	Causes the alt_load() facility to copy the .exceptions section. If true, this setting defines the macro ALT_LOAD_COPY_EXCEPTIONS in linker.h .
Restrictions:	none

hal.linker.enable_alt_load_copy_rodata

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	none
Description:	Causes the alt_load() facility to copy the .rodata section. If true, this setting defines the macro ALT_LOAD_COPY_RODATA in linker.h .
Restrictions:	none

hal.linker.enable_alt_load_copy_rwdata

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	none
Description:	Causes the initialization code to copy the <code>.rwdata</code> section. If true, this setting defines the macro <code>ALT_LOAD_COPY_RWDATA</code> in linker.h .
Restrictions:	none

hal.linker.enable_exception_stack

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	none
Description:	Enables use of a separate exception stack. If true, defines the macro <code>ALT_EXCEPTION_STACK</code> in linker.h , adds a memory region called <code>exception_stack</code> to <code>linker.x</code> , and provides the symbols <code>__alt_exception_stack_pointer</code> and <code>__alt_exception_stack_limit</code> in <code>linker.x</code> .
Restrictions:	The <code>hal.linker.exception_stack_size</code> and <code>hal.linker.exception_stack_memory_region_name</code> settings must also be valid. This setting must be false for MicroC/OS-II BSPs. The exception stack can be used to improve interrupt and other exception performance if an EIC is not implemented.

hal.linker.exception_stack_memory_region_name

Identifier:	none
Type:	Unquoted string
Default Value:	none
Destination File:	none
Description:	Name of the existing memory region to be divided up to create the <code>exception_stack</code> memory region. The selected region name is adjusted automatically when the BSP is generated to create the <code>exception_stack</code> memory region.
Restrictions:	Only used if <code>hal.linker.enable_exception_stack</code> is true.

hal.linker.exception_stack_size

Identifier:	none
Type:	Decimal number
Default Value:	1024
Destination File:	none
Description:	Size of the exception stack in bytes.
Restrictions:	Only used if <code>hal.linker.enable_exception_stack</code> is true.

hal.linker.enable_interrupt_stack

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	none
Description:	Enables use of a separate interrupt stack. If true, defines the macro <code>ALT_INTERRUPT_STACK</code> in linker.h , adds a memory region called <code>interrupt_stack</code> to linker.x , and provides the symbols <code>__alt_interrupt_stack_pointer</code> and <code>__alt_interrupt_stack_limit</code> in linker.x .
Restrictions:	The <code>hal.linker.interrupt_stack_size</code> and <code>hal.linker.interrupt_stack_memory_region_name</code> settings must also be valid. This setting must be false for MicroC/OS-II BSPs. Only enable this setting for systems with an EIC. If an EIC is not implemented, use the separate exception stack to improve interrupt and other exception performance.

hal.linker.interrupt_stack_memory_region_name

Identifier:	none
Type:	Unquoted String
Default Value:	none
Destination File:	none
Description:	Name of the existing memory region that is divided up to create the <code>interrupt_stack</code> memory region. The selected region name is adjusted automatically when the BSP is generated to create the <code>interrupt_stack</code> memory region.
Restrictions:	Only used if <code>hal.linker.enable_interrupt_stack</code> is true.

hal.linker.interrupt_stack_size

Identifier:	none
Type:	Decimal Number
Default Value:	1024
Destination File:	none
Description:	Size of the interrupt stack in bytes.
Restrictions:	Only used if <code>hal.linker.enable_interrupt_stack</code> is true.

hal.make.ar

Identifier:	AR
Type:	Unquoted string
Default Value:	nios2-elf-ar
Destination File:	BSP makefile
Description:	Archiver command. Creates library files.
Restrictions:	none

hal.make.ar_post_process

Identifier: AR_POST_PROCESS
Type: Unquoted string
Default Value: none
Destination File: BSP makefile
Description: Command executed after archiver execution.
Restrictions: none

hal.make.ar_pre_process

Identifier: AR_PRE_PROCESS
Type: Unquoted string
Default Value: none
Destination File: BSP makefile
Description: Command executed before archiver execution.
Restrictions: none

hal.make.as

Identifier: AS
Type: Unquoted string
Default Value: nios2-elf-gcc
Destination File: BSP makefile
Description: Assembler command. Note that CC is used for Nios II assembly language source files (.S).
Restrictions: none

hal.make.as_post_process

Identifier: AS_POST_PROCESS
Type: Unquoted string
Default Value: none
Destination File: BSP makefile
Description: Command executed after each assembly file is compiled.
Restrictions: none

hal.make.as_pre_process

Identifier: AS_PRE_PROCESS
Type: Unquoted string
Default Value: none
Destination File: BSP makefile
Description: Command executed before each assembly file is compiled.
Restrictions: none

hal.make.bsp_arflags

Identifier:	BSP_ARFLAGS
Type:	Unquoted string
Default Value:	-src
Destination File:	BSP makefile
Description:	Custom flags only passed to the archiver. This content of this variable is directly passed to the archiver rather than the more standard ARFLAGS. The reason for this is that GNU Make assumes some default content in ARFLAGS. This setting defines the value of BSP_ARFLAGS in Makefile.
Restrictions:	none

hal.make.bsp_asflags

Identifier:	BSP_ASFLAGS
Type:	Unquoted string
Default Value:	-Wa,-gdwarf2
Destination File:	BSP makefile
Description:	Custom flags only passed to the assembler. This setting defines the value of BSP_ASFLAGS in Makefile.
Restrictions:	none

hal.make.bsp_cflags_debug

Identifier:	BSP_CFLAGS_DEBUG
Type:	Unquoted string
Default Value:	-g
Destination File:	BSP makefile
Description:	C/C++ compiler debug level. <code>-g</code> provides the default set of debug symbols typically required to debug a typical application. Omitting <code>-g</code> removes debug symbols from the ELF. This setting defines the value of BSP_CFLAGS_DEBUG in Makefile.
Restrictions:	none

hal.make.bsp_cflags_defined_symbols

Identifier:	BSP_CFLAGS_DEFINED_SYMBOLS
Type:	Unquoted string
Default Value:	none
Destination File:	BSP makefile
Description:	Preprocessor macros to define. A macro definition in this setting has the same effect as a <code>#define</code> in source code. Adding <code>-DALT_DEBUG</code> to this setting has the same effect as <code>#define ALT_DEBUG</code> in a source file. Adding <code>-DFOO=1</code> to this setting is equivalent to the macro <code>#define FOO 1</code> in a source file. Macros defined with this setting are applied to all <code>.S</code> , C source (<code>.c</code>), and C++ files in the BSP. This setting defines the value of BSP_CFLAGS_DEFINED_SYMBOLS in the BSP makefile.
Restrictions:	none

hal.make.bsp_cflags_optimization

Identifier:	BSP_CFLAGS_OPTIMIZATION
Type:	Unquoted string
Default Value:	-O0
Destination File:	BSP makefile
Description:	C/C++ compiler optimization level. -O0 = no optimization, -O2 = normal optimization, etc. -O0 is recommended for code that you want to debug since compiler optimization can remove variables and produce nonsequential execution of code while debugging. This setting defines the value of BSP_CFLAGS_OPTIMIZATION in Makefile.
Restrictions:	none

hal.make.bsp_cflags_undefined_symbols

Identifier:	BSP_CFLAGS_UNDEFINED_SYMBOLS
Type:	Unquoted string
Default Value:	none
Destination File:	BSP makefile
Description:	Preprocessor macros to undefine. Undefined macros are similar to defined macros, but replicate the #undef directive in source code. To undefine the macro FOO use the syntax -u FOO in this setting. This is equivalent to #undef FOO in a source file. Note: the syntax differs from macro definition (there is a space, i.e. -u FOO versus -DFOO). Macros defined with this setting are applied to all .S, .c, and C++ files in the BSP. This setting defines the value of BSP_CFLAGS_UNDEFINED_SYMBOLS in the BSP Makefile.
Restrictions:	none

hal.make.bsp_cflags_user_flags

Identifier:	BSP_CFLAGS_USER_FLAGS
Type:	Unquoted string
Default Value:	none
Destination File:	BSP makefile
Description:	Custom flags passed to the compiler when compiling C, C++, and .S files. This setting defines the value of BSP_CFLAGS_USER_FLAGS in Makefile.
Restrictions:	none

hal.make.bsp_cflags_warnings

Identifier:	BSP_CFLAGS_WARNINGS
Type:	Unquoted string
Default Value:	-Wall
Destination File:	BSP makefile
Description:	C/C++ compiler warning level. -Wall is commonly used. This setting defines the value of BSP_CFLAGS_WARNINGS in Makefile.
Restrictions:	none

hal.make.bsp_cxx_flags

Identifier:	BSP_CXXFLAGS
Type:	Unquoted string
Default Value:	none
Destination File:	BSP makefile
Description:	Custom flags only passed to the C++ compiler. This setting defines the value of BSP_CXXFLAGS in Makefile.
Restrictions:	none

hal.make.bsp_inc_dirs

Identifier:	BSP_INC_DIRS
Type:	Unquoted string
Default Value:	none
Destination File:	BSP makefile
Description:	Space separated list of extra include directories to scan for header files. Directories are relative to the top-level BSP directory. The <code>-I</code> prefix is added by the makefile, therefore you must not include it in the setting value. This setting defines the value of <code>BSP_INC_DIRS</code> in the makefile.
Restrictions:	none

hal.make.build_post_process

Identifier:	BUILD_POST_PROCESS
Type:	Unquoted string
Default Value:	none
Destination File:	BSP makefile
Description:	Command executed after BSP built.
Restrictions:	none

hal.make.build_pre_process

Identifier:	BUILD_PRE_PROCESS
Type:	Unquoted string
Default Value:	none
Destination File:	BSP makefile
Description:	Command executed before BSP built.
Restrictions:	none

hal.make.cc

Identifier: CC
Type: Unquoted string
Default Value: nios2-elf-gcc -xc
Destination File: BSP makefile
Description: C compiler command
Restrictions: none

hal.make.cc_post_process

Identifier: CC_POST_PROCESS
Type: Unquoted string
Default Value: none
Destination File: BSP makefile
Description: Command executed after each **.c** or **.S** file is compiled.
Restrictions: none

hal.make.cc_pre_process

Identifier: CC_PRE_PROCESS
Type: Unquoted string
Default Value: none
Destination File: BSP makefile
Description: Command executed before each **.c** or **.S** file is compiled.
Restrictions: none

hal.make.cxx

Identifier: CXX
Type: Unquoted string
Default Value: nios2-elf-gcc -xc++
Destination File: BSP makefile
Description: C++ compiler command
Restrictions: none

hal.make.cxx_post_process

Identifier: CXX_POST_PROCESS
Type: Unquoted string
Default Value: none
Destination File: BSP makefile
Description: Command executed before each C++ file is compiled.
Restrictions: none

hal.make.cxx_pre_process

Identifier:	CXX_PRE_PROCESS
Type:	Unquoted string
Default Value:	none
Destination File:	BSP makefile
Description:	Command executed before each C++ file is compiled.
Restrictions:	none

hal.make.ignore_system_derived.big_endian

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query if SOPC system is big endian. If true ignores export of 'ALT_CFLAGS += -EB' to public.mk if big endian system. If true ignores export of 'ALT_CFLAGS += -EL' if little endian system.
Restrictions:	none

hal.make.ignore_system_derived.fpu_present

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query if SOPC system has FPU present. If true ignores export of 'ALT_CFLAGS += -mhard-float' to public.mk if FPU is found in the system. If true ignores export of 'ALT_CFLAGS += -mhard-soft' if FPU is not found in the system.
Restrictions:	none

hal.make.ignore_system_derived.hardware_divide_present

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query if SOPC system has hardware divide present. If true ignores export of 'ALT_CFLAGS += -mno-hw-div' to public.mk if no division is found in system. If true ignores export of 'ALT_CFLAGS += -mhw-div' if division is found in the system.
Restrictions:	none

hal.make.ignore_system_derived.hardware_fp_cust_inst_divider_present

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query if SOPC system floating point custom instruction with a divider is present. If true ignores export of 'ALT_CFLAGS += -mcustom-fpu-cfg=60-2' and 'ALT_LDFLAGS += -mcustom-fpu-cfg=60-2' to public.mk if the custom instruction is found in the system.
Restrictions:	none

hal.make.ignore_system_derived.hardware_fp_cust_inst_no_divider_present

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query if SOPC system floating point custom instruction without a divider is present. If true ignores export of 'ALT_CFLAGS += -mcustom-fpu-cfg=60-1' and 'ALT_LDFLAGS += -mcustom-fpu-cfg=60-1' to public.mk if the custom instruction is found in the system.
Restrictions:	none

hal.make.ignore_system_derived.sopc_simulation_enabled

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query if SOPC system has simulation enabled. If true ignores export of 'ELF_PATCH_FLAG += --simulation_enabled' to public.mk.
Restrictions:	none

hal.make.ignore_system_derived.debug_core_present

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query if SOPC system has a debug core present. If true ignores export of 'CPU_HAS_DEBUG_CORE = 1' to public.mk if a debug core is found in the system. If true ignores export of 'CPU_HAS_DEBUG_CORE = 0' if no debug core is found in the system.
Restrictions:	none

hal.make.ignore_system_derived.hardware_multiplier_present

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query if SOPC system has multiplier present. If true ignores export of 'ALT_CFLAGS += -mno-hw-mul' to public.mk if no multiplier is found in the system. If true ignores export of 'ALT_CFLAGS += -mhw-mul' if multiplier is found in the system.
Restrictions:	none

hal.make.ignore_system_derived.hardware_mulx_present

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query if SOPC system has hardware mulx present. If true ignores export of 'ALT_CFLAGS += -mno-hw-mulx' to public.mk if no mulx is found in the system. If true ignores export of 'ALT_CFLAGS += -mhw-mulx' if mulx is found in the system.
Restrictions:	none

hal.make.ignore_system_derived.sopc_system_base_address

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query SOPC system for system ID base address. If true ignores export of 'SOPC_SYSID_FLAG += --sidp=<address>' and 'ELF_PATCH_FLAG += --sidp=<address>' to public.mk.
Restrictions:	none

hal.make.ignore_system_derived.sopc_system_id

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query SOPC system for system ID. If true ignores export of 'SOPC_SYSID_FLAG += --id=<sysid>' and 'ELF_PATCH_FLAG += --id=<sysid>' to public.mk.
Restrictions:	none

hal.make.ignore_system_derived.sopc_system_timestamp

Identifier:	none
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enable BSP generation to query SOPC system for system timestamp. If true ignores export of 'SOPC_SYSID_FLAG += --timestamp=<timestamp>' and 'ELF_PATCH_FLAG += --timestamp=<timestamp>' to public.mk.
Restrictions:	none

hal.make.rm

Identifier:	RM
Type:	Unquoted string
Default Value:	rm -f
Destination File:	BSP makefile
Description:	Command used to remove files when building the <code>clean</code> target.
Restrictions:	none

hal.custom_newlib_flags

Identifier:	CUSTOM_NEWLIB_FLAGS
Type:	Unquoted string
Default Value:	none
Destination File:	public.mk
Description:	Build a custom version of newlib with the specified space-separated compiler flags.
Restrictions:	The custom newlib build is placed in the <i><bsp root>/newlib</i> directory, and is used only for applications that utilize this BSP.

hal.enable_c_plus_plus

Identifier:	ALT_NO_C_PLUS_PLUS
Type:	Boolean assignment
Default Value:	1
Destination File:	public.mk
Description:	Enable support for a subset of the C++ language. This option increases code footprint by adding support for C++ constructors. Certain features, such as multiple inheritance and exceptions are not supported. If false, adds <code>-DALT_NO_C_PLUS_PLUS</code> to <code>ALT_CPPFLAGS</code> in public.mk , and reduces code footprint.
Restrictions:	none

hal.enable_clean_exit

Identifier:	ALT_NO_CLEAN_EXIT
Type:	Boolean assignment
Default Value:	1
Destination File:	public.mk
Description:	When your application exits, close file descriptors, call C++ destructors, etc. Code footprint can be reduced by disabling clean exit. If disabled, adds <code>-DALT_NO_CLEAN_EXIT</code> to <code>ALT_CPPFLAGS</code> and <code>-Wl,--defsym,exit=_exit</code> to <code>ALT_LDFLAGS</code> in public.mk .
Restrictions:	none

hal.enable_exit

Identifier:	ALT_NO_EXIT
Type:	Boolean assignment
Default Value:	1
Destination File:	public.mk
Description:	Add <code>exit()</code> support. This option increases code footprint if your <code>main()</code> routine returns or calls <code>exit()</code> . If false, adds <code>-DALT_NO_EXIT</code> to <code>ALT_CPPFLAGS</code> in public.mk , and reduces footprint.
Restrictions:	none

hal.enable_gprof

Identifier:	ALT_PROVIDE_GMON
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Causes code to be compiled with gprof profiling enabled and the application ELF to be linked with the GPROF library. If true, adds <code>-DALT_PROVIDE_GMON</code> to <code>ALT_CPPFLAGS</code> and <code>-pg</code> to <code>ALT_CFLAGS</code> in public.mk .
Restrictions:	none

hal.enable_lightweight_device_driver_api

Identifier:	ALT_USE_DIRECT_DRIVERS
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Enables lightweight device driver API. This reduces code and data footprint by removing the HAL layer that maps device names (e.g. <code>/dev/uart0</code>) to file descriptors. Instead, driver routines are called directly. The <code>open()</code> , <code>close()</code> , and <code>lseek()</code> routines always fail if called. The <code>read()</code> , <code>write()</code> , <code>fstat()</code> , <code>ioctl()</code> , and <code>isatty()</code> routines only work for the stdio devices. If true, adds <code>-DALT_USE_DIRECT_DRIVERS</code> to <code>ALT_CPPFLAGS</code> in public.mk .
Restrictions:	The Altera Host and read-only ZIP file systems cannot be used if <code>hal.enable_lightweight_device_driver_api</code> is true.

hal.enable_mul_div_emulation

Identifier:	ALT_NO_INSTRUCTION_EMULATION
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	<p>Adds code to the BSP to emulate multiply and divide instructions. This code is independent of any emulation code added by the C/C++ compiler.</p> <p>If false, adds <code>-DALT_NO_INSTRUCTION_EMULATION</code> to <code>ALT_CPPFLAGS</code> in public.mk.</p> <p>You do not normally need to enable this option, because the C/C++ compiler detects whether the target Nios II processor core supports the multiply and divide instructions directly. If you compile for a core that lacks support for the instructions, the HAL includes the required software emulation in its run-time libraries.</p> <p>However, you might need to enable <code>hal.enable_mul_div_emulation</code> under the following circumstances:</p> <ul style="list-style-type: none">■ You expect to run the Nios II software on an implementation of the Nios II processor other than the one you compiled for. The best solution is to build your program for the correct Nios II processor implementation. Resort to the <code>hal.enable_mul_div_emulation</code> if it is not possible to determine the processor implementation at compile time.■ You have assembly language code that uses an implementation-dependent instruction.
Restrictions:	none

hal.enable_reduced_device_drivers

Identifier:	ALT_USE_SMALL_DRIVERS
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	<p>Certain drivers are compiled with reduced functionality to reduce code footprint. Not all drivers observe this setting.</p> <p>If true, adds <code>-DALT_USE_SMALL_DRIVERS</code> to <code>ALT_CPPFLAGS</code> in public.mk.</p> <p>Typically, drivers support this setting with a polled mode. For example, the <code>altera_avalon_uart</code> and <code>altera_avalon_jtag_uart</code> reduced drivers operate in polled mode.</p> <p>Several device drivers are disabled entirely in reduced drivers mode. These include the <code>altera_avalon_cfi_flash</code>, <code>altera_avalon_epcs_flash_controller</code>, and <code>altera_avalon_lcd_16207</code> drivers. As a result, certain API routines fail (HAL flash access routines). You can define a symbol provided by each driver to prevent it from being removed.</p>
Restrictions:	none

hal.enable_runtime_stack_checking

Identifier:	ALT_STACK_CHECK
Type:	Boolean assignment
Default Value:	0
Destination File:	public.mk
Description:	Turns on HAL runtime stack checking feature. Enabling this setting causes additional code to be placed into each subroutine call to generate an exception if a stack collision occurs with the heap or

statically allocated data. If true, adds `-DALT_STACK_CHECK` and `-mstack-check` to `ALT_CPPFLAGS` in **public.mk**.

Restrictions: none

hal.enable_sim_optimize

Identifier: `ALT_SIM_OPTIMIZE`

Type: Boolean assignment

Default Value: 0

Destination File: **public.mk**

Description: The BSP is compiled with optimizations to speedup HDL simulation such as initializing the cache, clearing the `.bss` section, and skipping long delay loops. If true, adds `-DALT_SIM_OPTIMIZE` to `ALT_CPPFLAGS` in **public.mk**.

Restrictions: When this setting is true, the BSP cannot run on hardware.

hal.enable_small_c_library

Identifier: none

Type: Boolean assignment

Default Value: 0

Destination File: **public.mk**

Description: Causes the small newlib (C library) to be used. This reduces code and data footprint at the expense of reduced functionality. Several newlib features are removed such as floating-point support in `printf()`, `stdin` input routines, and buffered I/O. The small C library is not compatible with Micrium MicroC/OS-II. If true, adds `-msmallc` to `ALT_LDFLAGS` and adds `-DSMALL_C_LIB` to `ALT_CPPFLAGS` in **public.mk**.

Restrictions: none

hal.enable_sopc_sysid_check

Identifier: none

Type: Boolean assignment

Default Value: 1

Destination File: **public.mk**

Description: Enables system ID check. If a System ID component is connected to the processor associated with this BSP, the system ID check is enabled in the creation of command-line arguments to download an ELF to the target. Otherwise, system ID and timestamp values are left out of **public.mk** for the application makefile `download-elf` target definition. With the system ID check disabled, the Nios II EDS tools do not automatically ensure that the application `.elf` file (and BSP it is linked against) corresponds to the hardware design on the target. If false, adds `--accept-bad-sysid` to `SOPC_SYSID_FLAG` in **public.mk**.

Altera strongly recommends leaving `hal.enable_sopc_sysid_check` enabled. This setting is exposed to support rare cases in which FPGA logic resources are in extremely short supply. When the system ID check is disabled, the software is unable to detect whether the software is running on the correct hardware version. This situation can lead to subtle errors that are difficult to diagnose.

Restrictions: none

hal.log_port

Identifier:	LOG_PORT
Type:	Unquoted string
Default Value:	none
Destination File:	system.h
Description:	Slave descriptor of debug logging character-mode device. If defined, it enables extra debug messages in the HAL source. This setting is used by the Altera logging functions.

hal.log_flags

Identifier:	ALT_LOG_FLAGS
Type:	Decimal Number
Default Value:	0
Destination File:	public.mk
Description:	The value is assigned to ALT_LOG_FLAGS in the generated public.mk . Refer to hal.log_port for further details. The valid range of this setting is 1 through 4.

hal.stderr

Identifier:	STDERR
Type:	Unquoted string
Default Value:	none
Destination File:	public.mk
Description:	Slave descriptor of STDERR character-mode device. This setting is used by the ALT_STDERR family of defines in system.h .

hal.stdin

Identifier:	STDIN
Type:	Unquoted string
Default Value:	none
Destination File:	system.h
Description:	Slave descriptor of STDIN character-mode device. This setting is used by the ALT_STDIN family of defines in system.h .

hal.stdout

Identifier:	STDOUT
Type:	Unquoted string
Default Value:	none
Destination File:	system.h
Description:	Slave descriptor of STDOUT character-mode device. This setting is used by the ALT_STDOUT family of defines in system.h .

Application and User Library Makefile Variables

The Nios II SBT constructs application and makefile libraries for you, inserting makefile variables appropriate to your project configuration. You can control project build characteristics by manipulating makefile variables at the time of project generation. You control a variable with the `--set` command line option, as in the following example:

```
nios2-bsp hal my_bsp --set APP_CFLAGS_WARNINGS "-Wall" 
```

The following utilities and scripts support modifying makefile variables with the `--set` option:

- **nios2-app-generate-makefile**
- **nios2-lib-generate-makefile**
- **nios2-app-update-makefile**
- **nios2-lib-update-makefile**
- **nios2-bsp**

Application Makefile Variables

You can modify the following application makefile variables on the command line:

- **CREATE_OBJDUMP**—Assign 1 to this variable to enable creation of an object dump file (**.objdump**) after linking the application. The **nios2-elf-objdump** utility is called to create this file. An object dump contains information about all object files linked into the **.elf** file. It provides a complete view of all code linked into your application. An object dump contains a disassembly view showing each instruction and its address.
- **OBJDUMP_INCLUDE_SOURCE**—Assign 1 to this variable to include source code inline with disassembled instructions in the object dump. When enabled, this includes the `--source` switch when calling the object dump executable. This is useful for debugging and examination of how the preprocessor and compiler generate instructions from higher level source code (such as C) or from macros.
- **OBJDUMP_FULL_CONTENTS**—Assign 1 to this variable to include a raw display of the contents of the **.text** linker section. When enabled, this variable includes the `--full-contents` switch when calling the object dump executable.
- **CREATE_ELF_DERIVED_FILES**—Setting this variable to 1 creates the HDL simulation and onchip memory initialization files when you invoke the makefile with the **all** target. When this variable is 0 (the default), these files are only created when you make the **mem_init_generate** or **mem_init_install** target.




Creating the HDL simulation and onchip memory initialization files increases project build time.

- **CREATE_LINKER_MAP**—Assign 1 to this variable to enable creation of a link map file (**.map**) after linking the application. A link map file provides information including which object files are included in the executable, the path to each object file, where objects and symbols are located in memory, and how the common symbols are allocated.


- **APP_CFLAGS_DEFINED_SYMBOLS**—This variable allows you to define macros using the `-D` argument, for example `-D <macro name>`. The contents of this variable are passed to the compiler and linker without modification.
- **APP_CFLAGS_UNDEFINED_SYMBOLS**—This variable allows you to remove macro definitions using the `-U` argument, for example `-U <macro name>`. The contents of this variable are passed to the compiler and linker without modification.
- **APP_CFLAGS_OPTIMIZATION**—The C/C++ compiler optimization level. For example, `-O0` provides no optimization and `-O2` provides standard optimization. `-O0` is recommended for debugging code, because compiler optimization can remove variables and produce non-sequential execution of code while debugging.
- **APP_CFLAGS_DEBUG_LEVEL**—The C/C++ compiler debug level. `-g` provides the default set of debug symbols typically required to debug an application. Omitting `-g` omits debug symbols from the `.elf`.
- **APP_CFLAGS_WARNINGS**—The C/C++ compiler warning level. `-Wall` is commonly used, enabling all warning messages.
- **APP_CFLAGS_USER_FLAGS**
- **APP_INCLUDE_DIRS**—Use this variable to specify paths for the preprocessor to search. These paths commonly contain C header files (`.h`) that application code requires. Each path name is formatted and passed to the preprocessor with the `-I` option.

You can add multiple directories by enclosing them in double quotes, for example `--set APP_INCLUDE_DIRS "../my_includes ../../other_includes"`.

- **APP_LIBRARY_DIRS**—Use this variable to specify paths for additional libraries that your application links with.

 When you specify a user library path with `APP_LIBRARY_DIRS`, you also need to specify the user library names with the `APP_LIBRARY_NAMES` variable.

`APP_LIBRARY_DIRS` specifies only the directory where the user library file(s) are located, not the library archive file (`.a`) name.

 Do not use this variable to specify the path to a BSP or user library created with the SBT. The paths to these libraries are specified in **public.mk** files included in the application makefile.

You can add multiple directories by enclosing them in double quotes, for example `--set APP_LIBRARY_DIRS "../my_libs ../../other_libs"`.

- **APP_LIBRARY_NAMES**—Use this variable to specify the names of additional libraries that your application must link with. Library files are `.a` files.



You do not specify the full name of the **.a** file. Instead, you specify the user library name **<name>**, and the SBT constructs the filename **lib<name>.a**. For example, if you add the string "math" to APP_LIBRARY_NAMES, the SBT assumes that your library file is named **libmath.a**.

Each specified user library name is passed to the linker with the **-l** option. The paths to locate these libraries must be specified in the APP_LIBRARY_DIRS variable.



You cannot use this variable to specify a BSP or user library created with the SBT. The paths to these libraries are specified in **public.mk** file included in the application makefile.

- BUILD_PRE_PROCESS—This variable allows you to specify a command to be executed prior to building the application, for example,
cp *.elf ../lastbuild.
- BUILD_POST_PROCESS—This variable allows you to specify a command to be executed after building the application, for example,
cp *.elf //production/test/nios2executables.

User Library Makefile Variables

You can modify the following user library makefile variables on the command line:

- LIB_CFLAGS_DEFINED_SYMBOLS—This variable allows you to define macros using the **-D** argument, for example **-D <macro name>**. The contents of this variable are passed to the compiler and linker without modification.
- LIB_CFLAGS_UNDEFINED_SYMBOLS—This variable allows you to remove macro definitions using the **-U** argument, for example **-U <macro name>**. The contents of this variable are passed to the compiler and linker without modification.
- LIB_CFLAGS_OPTIMIZATION—The C/C++ compiler optimization level. For example, **-O0** provides no optimization and **-O2** provides standard optimization. **-O0** is recommended for debugging code, because compiler optimization can remove variables and produce non-sequential execution of code while debugging.
- LIB_CFLAGS_DEBUG_LEVEL—The C/C++ compiler debug level. **-g** provides the default set of debug symbols typically required to debug an application. Omitting **-g** omits debug symbols from the **.elf**.
- LIB_CFLAGS_WARNINGS—The C/C++ compiler warning level. **-Wall** is commonly used, enabling all warning messages.
- LIB_CFLAGS_USER_FLAGS—
- LIB_INCLUDE_DIRS—You can add multiple directories by enclosing them in double quotes, for example **--set LIB_INCLUDE_DIRS
"../my_includes ../other_includes"**

Standard Build Flag Variables

The SBT creates makefiles supporting the following standard makefile command-line variables:

- CFLAGS
- CPPFLAGS
- ASFLAGS
- CXXFLAGS

You can define flags in these variables on the makefile command line, or in a script that invokes the makefile. The makefile passes these flags on to the corresponding GCC tool.

Software Build Tools Tcl Commands

Tcl commands are a crucial component of the Nios II SBT. Tcl commands allow you to exercise detailed control over BSP generation, as well as to define drivers and software packages. This section describes the Tcl commands, the environments in which they run, and how the commands work together.

Tcl Command Environments

The Nios II SBT supports Tcl commands in the following environments:

- BSP setting specification—In this environment, you manipulate BSP settings to control the static characteristics of the BSP. BSP setting commands are executed before the BSP is generated.
- BSP generation callbacks—In this environment, you exercise further control over BSP details, managing settings that interact with one another and with the hardware design. BSP callbacks run at BSP generation time.
- Device driver and software package definition—In this environment, you bundle source files into a custom driver or package. This process prepares the driver or package so that a BSP developer can include it in a BSP using the SBT.

The following sections describe each Tcl environment in detail, listing the available commands.

Tcl Commands for BSP Settings

[“Settings Managed by the Software Build Tools” on page 15-37](#) describes settings that are available in a Nios II project. This section describes the tools that you use to specify and manipulate these settings.

You manipulate project settings with BSP Tcl commands. The commands in this section are used with the utilities **nios2-bsp-create-settings**, **nios2-bsp-update-settings**, and **nios2-bsp-query-settings**. You can call the Tcl commands directly on a utility command line using the `--cmd` option, or you can put them in a Tcl script, specified with the `--script` option. For details about how to call Tcl commands from utilities, refer to [“Nios II Software Build Tools Utilities” on page 15-1](#).



For more information about creating Tcl scripts, refer to “Tcl Scripts for BSP Settings” in the *Nios II Software Build Tools* chapter of the *Nios II Software Developer’s Handbook*. This chapter includes a discussion of the default Tcl script, which provides excellent usage examples of many of the Tcl commands described in this section.

The following commands are available to manipulate BSP settings:

- “add_memory_device” on page 15-82
- “add_memory_region” on page 15-82
- “add_section_mapping” on page 15-83
- “are_same_resource” on page 15-83
- “delete_memory_region” on page 15-84
- “delete_section_mapping” on page 15-84
- “disable_sw_package” on page 15-84
- “enable_sw_package” on page 15-84
- “get_addr_span” on page 15-85
- “get_assignment” on page 15-85
- “get_available_drivers” on page 15-86
- “get_available_sw_packages” on page 15-86
- “get_base_addr” on page 15-86
- “get_break_offset” on page 15-87
- “get_break_slave_desc” on page 15-87
- “get_cpu_name” on page 15-87
- “get_current_memory_regions” on page 15-88
- “get_current_section_mappings” on page 15-88
- “get_default_memory_regions” on page 15-89
- “get_driver” on page 15-89
- “get_enabled_sw_packages” on page 15-90
- “get_exception_offset” on page 15-90
- “get_exception_slave_desc” on page 15-90
- “get_fast_tlb_miss_exception_offset” on page 15-91
- “get_fast_tlb_miss_exception_slave_desc” on page 15-91
- “get_interrupt_controller_id” on page 15-91
- “get_irq_interrupt_controller_id” on page 15-92
- “get_irq_number” on page 15-92
- “get_memory_region” on page 15-92
- “get_module_class_name” on page 15-93
- “get_module_name” on page 15-93

- “get_reset_offset” on page 15-93
- “get_reset_slave_desc” on page 15-94
- “get_section_mapping” on page 15-94
- “get_setting” on page 15-94
- “get_setting_desc” on page 15-95
- “get_slave_descs” on page 15-95
- “is_char_device” on page 15-96
- “is_connected_interrupt_controller_device” on page 15-96
- “is_connected_to_data_master” on page 15-96
- “is_connected_to_instruction_master” on page 15-97
- “is_ethernet_mac_device” on page 15-97
- “is_flash” on page 15-97
- “is_memory_device” on page 15-97
- “is_non_volatile_storage” on page 15-98
- “is_timer_device” on page 15-98
- “log_debug” on page 15-98
- “log_default” on page 15-99
- “log_error” on page 15-99
- “log_verbose” on page 15-99
- “set_driver” on page 15-99
- “set_ignore_file” on page 15-100
- “set_setting” on page 15-101
- “update_memory_region” on page 15-101
- “update_section_mapping” on page 15-102
- “add_default_memory_regions” on page 15-102
- “create_bsp” on page 15-102
- “generate_bsp” on page 15-102
- “get_available_bsp_type_versions” on page 15-103
- “get_available_bsp_types” on page 15-103
- “get_available_cpu_architectures” on page 15-103
- “get_available_cpu_names” on page 15-103
- “get_available_software” on page 15-104
- “get_available_software_setting_properties” on page 15-104
- “get_available_software_settings” on page 15-105
- “get_bsp_version” on page 15-105

- “get_cpu_architecture” on page 15-105
- “get_nios2_dpx_thread_num” on page 15-105
- “get_sopcinfo_file” on page 15-106
- “get_supported_bsp_types” on page 15-106
- “is_bsp_hal_extension” on page 15-106
- “is_bsp_lwhal_extension” on page 15-106
- “open_bsp” on page 15-106
- “save_bsp” on page 15-107
- “set_bsp_version” on page 15-107
- “set_logging_mode” on page 15-107

add_memory_device

Usage

`add_memory_device <device name> <base address> `

Options

- *<device name>*: String with the name of the memory device.
- *<base address>*: The base address of the memory device. Hexadecimal or decimal string.
- **: The size (span) of the memory device. Hexadecimal or decimal string.

Description

This command is provided to define a user-defined external memory device, outside the hardware system. Such a device would typically be mapped through a bridge component. This command adds an external memory device to the BSP’s memory map, allowing the BSP to define memory regions and section mappings for the memory as if it were part of the system. The external memory device parameters are stored in the BSP settings file.

add_memory_region

Usage

`add_memory_region <name> <slave_desc> <offset> `

Options

- *<name>*: String with the name of the memory region to create.
- *<slave_desc>*: String with the slave descriptor of the memory device for this region.
- *<offset>*: String with the byte offset of the memory region from the memory device base address.
- **: String with the span of the memory region in bytes.

Description

Creates a new memory region for the linker script. This memory region must not overlap with any other memory region and must be within the memory range of the associated slave descriptor. The offset and span are decimal numbers unless prefixed with 0x.

Example

```
add_memory_region onchip_ram0 onchip_ram0 0 0x100000
```

add_section_mapping

Usage

```
add_section_mapping <section_name> <memory_region_name>
```

Options

- *<section_name>*: String with the name of the linker section.
- *<memory_region_name>*: String with the name of the memory region to map.

Description

Maps the specified linker section to the specified linker memory region. If the section does not already exist, `add_section_mapping` creates it. If it already exists, `add_section_mapping` overrides the existing mapping with the new one. The linker creates the section mappings in the order in which they appear in the linker script.

Example

```
add_section_mapping .text onchip_ram0
```

are_same_resource

Usage

```
are_same_resource <slave_desc1> <slave_desc2>
```

Options

- *<slave_desc1>*: String with the first slave descriptor to compare.
- *<slave_desc2>*: String with the second slave descriptor to compare.

Description

Returns a boolean value that indicates whether the two slave descriptors are connected to the same resource. To connect to the same resource, the two slave descriptors must be associated with the same module. The module specifies whether two slaves access the same resource or different resources within that module. For example, a dual-port memory has two slaves that access the same resource (the memory). However, you could create a module that has two slaves that access two different resources such as a memory and a control port.

delete_memory_region

Usage

```
delete_memory_region <region_name>
```

Options

- *<region_name>*: String with the name of the memory region to delete.

Description

Deletes the specified memory region. The region must exist to avoid an error condition.

delete_section_mapping

Usage

```
delete_section_mapping <section_name>
```

Options

- *<section_name>*: String with the name of the section.

Description

Deletes the specified section mapping.

Example

```
delete_section_mapping .text
```

disable_sw_package

Usage

```
disable_sw_package <software_package_name>
```

Options

- *<software_package_name>*: String with the name of the software package.

Description

Disables the specified software package. Settings belonging to the package are no longer available in the BSP, and associated source files are not included in the BSP makefile. It is an error to disable a software package that is not enabled.

enable_sw_package

Usage

```
enable_sw_package <software_package_name>
```

Options

- *<software_package_name>*: String with the name of the software package, with the version number optionally appended with a ':'.
For example, `enable_sw_package my_package:1.0`.

Description

Enables a software package. Adds its associated source files and settings to the BSP. Specify the desired version in the form `<software_package_name>:<version>`. If you do not specify the version, `enable_sw_package` selects the latest available version.

Examples

- Example 1:

```
enable_sw_package altera_hostfs:7.2
```

- Example 2:

```
enable_sw_package my_sw_package
```

get_addr_span

Usage

```
get_addr_span <slave_desc>
```

Options

- `<slave_desc>`: String with the slave descriptor to query.

Description

Returns the address span (length in bytes) of the slave descriptor as an integer decimal number.

Example

```
puts [get_addr_span onchip_ram_64_kbytes]
```

Returns:

```
65536
```

get_assignment

Usage

```
get_assignment <module_name> <assignment_name>
```

Options

- `<module_name>`: Module instance name to query for assignment
- `<assignment_name>`: Module instance assignment name to query for

Description

Returns the name of the value of the assignment for a specified module instance name.

Example

```
puts [get_assignment "cpu0" "embeddedsw.configuration.breakSlave"]
```

Returns:

```
memory_0.s0
```

get_available_drivers

Usage

```
get_available_drivers <module_name>
```

Options

- *<module_name>*: String with the name of the module to query.

Description

Returns a list of available device driver names that are compatible with the specified module instance. The list is empty if there are no drivers available for the specified slave descriptor. The format of each entry in the list is the driver name followed by a colon and the version number (if provided).

Example

```
puts [get_available_drivers jtag_uart]
```

Returns:

```
altera_avalon_jtag_uart_driver:7.2 altera_avalon_jtag_uart_driver:6.1
```

get_available_sw_packages

Usage

```
get_available_sw_packages
```

Options

None

Description

Returns a list of software package names that are available for the current BSP. The format of each entry in the list is a string containing the package name followed by a colon and the version number (if provided).

Example

```
puts [get_available_sw_packages]
```

Returns:

```
altera_hostfs:7.2 altera_ro_zipfs:7.2
```

get_base_addr

Usage

```
get_base_addr <slave_desc>
```

Options

- *<slave_desc>*: String with the slave descriptor to query.

Description

Returns the base byte address of the slave descriptor as an integer decimal number.

Example

```
puts [get_base_addr jtag_uart]
```

Returns:

```
67616
```

get_break_offset

Usage

```
get_break_offset
```

Options

None

Description

Returns the byte offset of the processor break address.

Example

```
puts [get_break_offset]
```

Returns:

```
32
```

get_break_slave_desc

Usage

```
get_break_slave_desc
```

Options

None

Description

Returns the slave descriptor associated with the processor break address. If null, then the break device is internal to the processor (debug module).

Example

```
puts [get_break_slave_desc]
```

Returns:

```
onchip_ram_64_kbytes
```

get_cpu_name

Usage

```
get_cpu_name
```

Options

None

Description

Returns the name of the BSP specific processor.

Example

```
puts [get_cpu_name]
```

Returns:

```
cpu_0
```

get_current_memory_regions**Usage**

```
get_current_memory_regions
```

Options

None

Description

Returns a sorted list of records representing the existing linker script memory regions. Each record in the list represents a memory region. Each record is a list containing the region name, associated memory device slave descriptor, offset, and span, in that order.

Example

```
puts [get_current_memory_regions]
```

Returns:

```
{reset onchip_ram0 0 32} {onchip_ram0 onchip_ram0 32 1048544}
```

get_current_section_mappings**Usage**

```
get_current_section_mappings
```

Options

None

Description

Returns a list of lists for all the current section mappings. Each list represents a section mapping with the format {section_name memory_region}. The order of the section mappings matches their order in the linker script.

Example

```
puts [get_current_section_mappings]
```

Returns:

```
{.text onchip_ram0} {.rodata onchip_ram0} {.rwdata onchip_ram0}  
  {.bss onchip_ram0} {.heap onchip_ram0} {.stack onchip_ram0}
```


get_default_memory_regions

Usage

```
get_default_memory_regions
```

Options

None

Description

Returns a sorted list of records representing the default linker script memory regions. The default linker script memory regions are the best guess for memory regions based on the reset address and exception address of the processor associated with the BSP, and all other processors in the system that share memories with the processor associated with the BSP. Each record in the list represents a memory region. Each record is a list containing the region name, associated memory device slave descriptor, offset, and span, in that order.

Example

```
puts [get_default_memory_regions]
```

Returns:

```
{reset onchip_ram0 0 32} {onchip_ram0 onchip_ram0 32 1048544}
```

get_driver

Usage

```
get_driver <module_name>
```

Options

- *<module_name>*: String with the name of the module instance to query.

Description

Returns the driver name associated with the specified module instance. The format is *<driver name>* followed by a colon and the version (if provided). Returns the string "none" if there is no driver associated with the specified module instance name.

Examples

- Example 1:

```
puts [get_driver jtag_uart]
```

Returns:

```
altera_avalon_jtag_uart_driver:7.2
```

- Example 2:

```
puts [get_driver onchip_ram_64_kbytes]
```

Returns:

```
none
```

get_enabled_sw_packages

Usage

```
get_enabled_sw_packages
```

Options

None

Description

Returns a list of currently enabled software packages. The format of each entry in the list is the software package name followed by a colon and the version number (if provided).

Example

```
puts [get_enabled_sw_packages]
```

Returns:

```
altera_hostfs:7.2
```

get_exception_offset

Usage

```
get_exception_offset
```

Options

None

Description

Returns the byte offset of the processor exception address.

Example

```
puts [get_exception_offset]
```

Returns:

```
32
```

get_exception_slave_desc

Usage

```
get_exception_slave_desc
```

Options

None

Description

Returns the slave descriptor associated with the processor exception address.

Example

```
puts [get_exception_slave_desc]
```

Returns:

`onchip_ram_64_kbytes`

get_fast_tlb_miss_exception_offset

Usage

`get_fast_tlb_miss_exception_offset`

Options

None

Description

Returns the byte offset of the processor fast translation lookaside buffer (TLB) miss exception address. Only a processor with an MMU has such an exception address.

Example

```
puts [get_fast_tlb_miss_exception_offset]
```

Returns:

32

get_fast_tlb_miss_exception_slave_desc

Usage

`get_fast_tlb_miss_exception_slave_desc`

Options

None

Description

Returns the slave descriptor associated with the processor fast TLB miss exception address. Only a processor with an MMU has such an exception address.

Example

```
puts [get_fast_tlb_miss_exception_slave_desc]
```

Returns:

`onchip_ram_64_kbytes`

get_interrupt_controller_id

Usage

`get_interrupt_controller_id <slave_desc>`

Options

- `<slave_desc>`: String with the slave descriptor to query.

Description

Returns the interrupt controller ID of the slave descriptor (-1 if not a connected interrupt controller).

get_irq_interrupt_controller_id**Usage**

```
get_irq_interrupt_controller_id <slave_desc>
```

Options

- *<slave_desc>*: String with the slave descriptor to query.

Description

Returns the interrupt controller ID connected to the IRQ associated with the slave descriptor (-1 if none).

get_irq_number**Usage**

```
get_irq_number <slave_desc>
```

Options

- *<slave_desc>*: String with the slave descriptor to query.

Description

Returns the interrupt request number of the slave descriptor, or -1 if no interrupt request number is found.

get_memory_region**Usage**

```
get_memory_region <name>
```

Options

- *<name>*: String with the name of the memory region.

Description

Returns the linker script region information for the specified region. The format of the region is a list containing the region name, associated memory device slave descriptor, offset, and span in that order.

Example

```
puts [get_memory_region reset]
```

Returns:

```
reset onchip_ram0 0 32
```

get_module_class_name

Usage

```
get_module_class_name <module_name>
```

Options

- *<module_name>*: String with the module instance name to query.

Description

Returns the name of the module class associated with the module instance.

Example

```
puts [get_module_class_name jtag_uart0]
```

Returns:

```
altera_avalon_jtag_uart
```

get_module_name

Usage

```
get_module_name <slave_desc>
```

Options

- *<slave_desc>*: String with the slave descriptor to query.

Description

Returns the name of the module instance associated with the slave descriptor. If a module with one slave, or if it has multiple slaves connected to the same resource, the slave descriptor is the same as the module name. If a module has multiple slaves that do not connect to the same resource, the slave descriptor consists of the module name followed by an underscore and the slave name.

Example

```
puts [get_module_name multi_jtag_uart0_s1]
```

Returns:

```
multi_jtag_uart0
```

get_reset_offset

Usage

```
get_reset_offset
```

Options

None

Description

Returns the byte offset of the processor reset address.

Example

```
puts [get_reset_offset]
```

Returns:

```
0
```

get_reset_slave_desc**Usage**

```
get_reset_slave_desc
```

Options

None

Description

Returns the slave descriptor associated with the processor reset address.

Example

```
puts [get_reset_slave_desc]
```

Returns:

```
onchip_ram_64_kbytes
```

get_section_mapping**Usage**

```
get_section_mapping <section_name>
```

Options

- *<section_name>*: String with the section name to query.

Description

Returns the name of the memory region for the specified linker section. Returns null if the linker section does not exist.

Example

```
puts [get_section_mapping .text]
```

Returns:

```
onchip_ram0
```

get_setting**Usage**

```
get_setting <name>
```

Options

- *<name>*: String with the name of the setting to get.

Description

Returns the value of the specified BSP setting. `get_setting` returns boolean settings with the value 1 or 0. If the value of the setting is an empty string, `get_setting` returns "none".

The `get_setting` command is equivalent to the `--get` command-line option.

Example

```
puts [get_setting hal.enable_gprof]
```

Returns:

```
0
```

get_setting_desc

Usage

```
get_setting_desc <name>
```

Options

- `<name>`: String with the name of the setting to get the description for.

Description

Returns a string describing the BSP setting.

Example

```
puts [get_setting_desc hal.enable_gprof]
```

Returns:

```
"This example compiles the code with gprof profiling enabled and links \
  the application ELF with the GPROF library. If true, adds \
  -DALT_PROVIDE_GMON to ALT_CPPFLAGS and -pg to ALT_CFLAGS in
public.mk."
```

get_slave_descs

Usage

```
get_slave_descs
```

Options

None

Description

Returns a sorted list of all the slave descriptors connected to the Nios II processor.

Example

```
puts [get_slave_descs]
```

Returns:

```
jtag_uart0 onchip_ram0
```

is_char_device

Usage

```
is_char_device <slave_desc>
```

Options

- *<slave_desc>*: String with the slave descriptor to query.

Description

Returns a boolean value that indicates whether the slave descriptor is a character device.

Examples

- Example 1:

```
puts [is_char_device jtag_uart]
```

Returns:

```
1
```

- Example 2:

```
puts [is_char_device onchip_ram_64_kbytes]
```

Returns:

```
0
```

is_connected_interrupt_controller_device

Usage

```
is_connected_interrupt_controller_device <slave_desc>
```

Options

- *<slave_desc>*: String with the slave descriptor to query.

Description

Returns a boolean value that indicates whether the slave descriptor is an interrupt controller device that is connected to the processor so that the interrupt controller sends interrupts to the processor.

is_connected_to_data_master

Usage

```
is_connected_to_data_master <slave_desc>
```

Options

- *<slave_desc>*: String with the slave descriptor to query.

Description

Returns a boolean value that indicates whether the slave descriptor is connected to a data master.

is_connected_to_instruction_master

Usage

`is_connected_to_instruction_master <slave_desc>`

Options

- `<slave_desc>`: String with the slave descriptor to query.

Description

Returns a boolean value that indicates whether the slave descriptor is connected to an instruction master.

is_ethernet_mac_device

Usage

`is_ethernet_mac_device <slave_desc>`

Options

- `<slave_desc>`: String with the slave descriptor to query.

Description

Returns a boolean value that indicates whether the slave descriptor is an Ethernet MAC device.

is_flash

Usage

`is_flash <slave_desc>`

Options

- `<slave_desc>`: String with the slave descriptor to query.

Description

Returns a boolean value that indicates whether the slave descriptor is a flash memory device.

is_memory_device

Usage

`is_memory_device <slave_desc>`

Options

- `<slave_desc>`: String with the slave descriptor to query.

Description

Returns a boolean value that indicates whether the slave descriptor is a memory device.

Examples

■ Example 1:

```
puts [is_memory_device jtag_uart]
```

Returns:

```
0
```

■ Example 2:

```
puts [is_memory_device onchip_ram_64_kbytes]
```

Returns:

```
1
```

is_non_volatile_storage

Usage

```
is_non_volatile_storage <slave_desc>
```

Options

- <slave_desc>: String with the slave descriptor to query.

Description

Returns a boolean value that indicates whether the slave descriptor is a non-volatile storage device.

is_timer_device

Usage

```
is_timer_device <slave_desc>
```

Options

- <slave_desc>: String with the slave descriptor to query.

Description

Returns a boolean value that indicates whether the slave descriptor is a timer device.

log_debug

Usage

```
log_debug <message>
```

Options

- <message>: String with message to log.

Description

Displays a message to the host's stdout when the logging level is debug.

log_default

Usage

```
log_default <message>
```

Options

- <message>: String with message to log.

Description

Displays a message to the host's stdout when the logging level is default or higher.

Example

```
log_default "This is a default message."
```

Displays:

```
INFO: Tcl message: "This is a default message."
```

log_error

Usage

```
log_error <message>
```

Options

- <message>: String with message to log.

Description

Displays a message to the host's stderr, regardless of logging level.

log_verbose

Usage

```
log_verbose <message>
```

Options

- <message>: String with message to log.

Description

Displays a message to the host's stdout when the logging level is verbose or higher.

set_driver

Usage

```
set_driver <driver_name> <module_name>
```

Options

- <driver_name>: String with the name of the device driver to use.
- <module_name>: String with the name of the module instance to set.

Description

Selects the specified device driver for the specified module instance. The `<driver_name>` argument includes a version number, delimited by a colon (:). If you omit the version number, `set_driver` uses the latest available version of the driver that is compatible with the component specified by the `<module_name>` argument.

If `<driver_name>` is none, the specified module instance does not use a driver. If `<driver_name>` is not none, it must be the name of the associated component class.

Examples

■ Example 1:

```
set_driver altera_avalon_jtag_uart_driver:7.2 jtag_uart
```

■ Example 2:

```
set_driver none jtag_uart
```

set_ignore_file

Usage

```
set_ignore_file <software_component_name> <file_name> <ignore>
```

Options

- `<software_component_name>`: Name of the driver, software package, or operating system to which the file belongs.
- `<file_name>`: Name of the file.
- `<ignore>`: Set to true to ignore (not generate or copy) the file, false to generate or copy the file normally.

Description

You can use this command to have a specific BSP file ignored (not generated or copied) during BSP generation. This command allows you to take ownership of a specific file, modify it, and prevent the SBT from overwriting your modifications.

`<software_component_name>` can have one of the following values:

- `<driver_name>`—The name of a driver, as specified with the `create_driver` command in the `*_sw.tcl` file that defines the driver. Specifies that `<file_name>` is a copied file associated with a device driver.
- `<software_package_name>`—The name of a software package, specified with the `create_sw_package` command in the `*_sw.tcl` file that defines the package. Specifies that `<file_name>` is a copied file associated with a software package.
- `<OS_name>`—The name of an OS, specified with the `create_os` command in the `*_sw.tcl` file that defines the OS, and is used in the **nios2-bsp-create-settings** to specify the BSP type. Specifies that `<file_name>` is a copied file associated with an OS.

- **generated**—Specifies that *<file_name>* is a generated top-level BSP file. The list of generated BSP files depends on the BSP type. For a list of generated files associated with HAL and MicroC/OS-II BSPs, refer to “Details of BSP Creation” in the *Nios II Software Build Tools* chapter of the *Nios II Software Developer’s Handbook*. For a list of generated files associated with a third-party OS, refer to the OS supplier’s documentation.

set_setting

Usage

```
set_setting <name> <value>
```

Options

- **<name>**: String with the name of the setting.
- **<value>**: String with the value of the setting.

Description

Sets the value for the specified BSP setting. Legal values for boolean settings are true, false, 1, and 0. Use the keyword none instead of an empty string to set a string to an empty value. The set_setting command is equivalent to the --set command-line option.

Example

```
set_setting hal.enable_gprof true
```

update_memory_region

Usage

```
update_memory_region <name> <slave_desc> <offset> <span>
```

Options

- **<name>**: String with the name of the memory region to update.
- **<slave_desc>**: String with the slave descriptor of the memory device for this region.
- **<offset>**: String with the byte offset of the memory region from the memory device base address.
- ****: String with the span of the memory region in bytes.

Description

Updates an existing memory region for the linker script. This memory region must not overlap with any other memory region and must be within the memory range of the associated slave descriptor. The offset and span are decimal numbers unless prefixed with 0x.

Example

```
update_memory_region onchip_ram0 onchip_ram0 0 0x100000
```

update_section_mapping

Usage

```
update_section_mapping <section_name> <memory_region_name>
```

Options

- `<section_name>`: String with the name of the linker section.
- `<memory_region_name>`: String with the name of the memory region to map.

Description

Updates the specified linker section. The linker creates the section mappings in the order in which they appear in the linker script.

Example

```
update_section_mapping .text onchip_ram0
```

add_default_memory_regions

Usage

```
add_default_memory_regions
```

Description

Defaults the BSP to use default linker script memory regions. The default linker script memory regions are the best guess for memory regions based on the reset address and exception address of the processor associated with the BSP, and all other processors in the system that share memories with the processor associated with the BSP.

create_bsp

Usage

```
create_bsp <bspType> <bsp version> <processor name> <sopcinfo>
```

Options

- `<bspType>`: Type of BSP to create.
- `<bsp version>`: Version of BSP software element to utilize.
- `<processor name>`: Name of processor instance for BSP
- `<sopcinfo>`: .sopcinfo generated file that describes the system the BSP is for.

Description

Creates a new BSP.

generate_bsp

Usage

```
generate_bsp <bspDir>
```

Options

- `<bspDir>`: BSP directory to generate files to.

Description

Generates a new BSP.

get_available_bsp_type_versions

Usage

```
get_available_bsp_type_versions <bsp_type> <sopcinfolpath>
```

Options

- `<bsp_type>`: BSP type identifier (e.g. hal, ucousii).
- `<sopcinfolpath>`: SOPC Information File path. Its parent folder might include custom BSP IP software components (*_sw.tcl).

Description

Gets the available BSP type versions.

get_available_bsp_types

Usage

```
get_available_bsp_types <sopcinfolpath>
```

Options

- `<sopcinfolpath>`: SOPC Information File path. Its parent folder might include custom BSP IP software components (*_sw.tcl).

Description

Gets the available BSP type identifiers.

get_available_cpu_architectures

Usage

```
get_available_cpu_architectures
```

Description

Gets the available processor architectures.

get_available_cpu_names

Usage

```
get_available_cpu_names <sopcinfolpath>
```

Options

- `<sopcinfolpath>`: SOPC Information File path that contains processor instances

Description

Gets the processor names given a SOPC system.

get_available_software**Usage**

```
get_available_software <bsp_type> <filter> <sopcinfol_path>
```

Options

- <bsp_type>: BSP type identifier (e.g. hal, ucousii).
- <sopcinfol_path>: SOPC Information File path. Its parent folder might include custom BSP IP software components (*_sw.tcl).
- <filter>: A filter can be applied to restrict results. The following filters are available:
 - all
 - drivers
 - sw_packages
 - os_elements

Comma-separated tokens are acceptable.

Description

Gets the available software (drivers, software packages, and bsp components) for a given BSP type.

get_available_software_setting_properties**Usage**

```
get_available_software_setting_properties <setting_name> \
  <software_name> <software_version> <sopcinfol_path>
```

Options

- <software_name>: Name of a software component (e.g. "altera_avalon_uart_driver", or "hal").
- <software_version>: Enter "default" for latest version, or a specific version number.
- <setting_name>: Name of a selected software component setting to get properties for (e.g. hal.linker.allow_code_at_reset).
- <sopcinfol_path>: SOPC Information File path. Its parent folder might include custom BSP IP software components (*_sw.tcl).

Description

Gets the available setting names for a software component.

get_available_software_settings

Usage

```
get_available_software_settings <software_name> <software_version> \  
    <sopcinfolpath>
```

Options

- <software_name>: Name of a software component (e.g. altera_avalon_uart_driver).
- <software_version>: Enter "default" for latest version, or a specific version number.
- <sopcinfolpath>: SOPC Information File path. Its parent folder can include custom BSP IP software components (*_sw.tcl).

Description

Gets the available setting names for a software component.

get_bsp_version

Usage

```
get_bsp_version
```

Description

Gets the version of the BSP operating system software element.

get_cpu_architecture

Usage

```
get_cpu_architecture <processor_name> <sopcinfolpath>
```

Options

- <processor_name>: processor instance name
- <sopcinfolpath>: SOPC Information File path that contains processor_name instance

Description

Gets the processor architecture (e.g. nios2) of a specified processor instance given a SOPC system.

get_nios2_dpx_thread_num

Usage

```
get_nios2_dpx_thread_num
```

Description

If the BSP is mastered by a Nios II DPX processor, then this function returns the number of threads the processor supports. Otherwise it returns null.

get_sopcinfo_file

Usage

```
get_sopcinfo_file
```

Description

Returns the path of the BSP specific SOPC Information File.

get_supported_bsp_types

Usage

```
get_supported_bsp_types <processor_name> <sopcinfo_path>
```

Options

- <processor_name>: processor instance name
- <sopcinfo_path>: SOPC Information File path. Its parent folder can include custom BSP IP software components (*_sw.tcl).

Description

Gets the BSP types supported for a given processor and SOPC system.

is_bsp_hal_extension

Usage

```
is_bsp_hal_extension
```

Description

Returns a boolean value that indicates whether the BSP instantiated is of a type based on Altera HAL.

is_bsp_lwhal_extension

Usage

```
is_bsp_lwhal_extension
```

Description

Returns a boolean value that indicates whether the BSP instantiated is of a type based on Altera Lightweight HAL.

open_bsp

Usage

```
open_bsp <settingsFile>
```

Options

- <settingsFile>: .bsp settings file to open.

Description

Opens an existing BSP.

save_bsp

Usage

```
save_bsp <settingsFile>
```

Options

- <settingsFile>: .bsp settings file to save BSP to.

Description

Saves a new BSP.

set_bsp_version

Usage

```
set_bsp_version <version>
```

Options

- <version>: Version of BSP type software element to use.

Description

Sets the version of the BSP operating system software element to a specific value. The value 'default' uses the latest version available. If this call is not used, the BSP is created using the 'default' BSP software element version.

set_logging_mode

Usage

```
set_logging_mode <mode>
```

Options

- <mode>: Logging Mode: 'silent', 'default', 'verbose', 'debug'

Description

Sets the verbosity level of the logger. Useful to eliminate tool informative messages

Tcl Commands for BSP Generation Callbacks

If you are defining a device driver or a software package, you can define Tcl callback functions to run whenever a BSP is generated containing your driver or package. Tcl callback functions enable you to create settings dynamically for the driver or package. This capability is essential when the driver or package settings must be customized to the hardware configuration, or to other BSP settings.

Tcl callback scripts are defined and controlled from the *_sw.tcl file associated with the driver or package. In *_sw.tcl, you can specify where the Tcl functions come from, when function runs, and the scope of each Tcl function's operation.

When the BSP is generated with your driver or software package, the settings you define in the callback scripts are inserted in **settings.bsp**.

You specify the source of the callback functions with the `set_sw_property` command, using the `callback_source_file` property.

A Tcl callback function can run at one of the following times:

- BSP initialization
- BSP generation
- BSP validation



Although you can specify a new setting's value when you create the setting at BSP initialization, the setting's value can change between initialization and generation. For example, the BSP developer might edit the setting in the BSP Editor.

A Tcl callback can function in either of the following scopes:

- Component class
- Component instance

You specify each callback function's runtime environment by using the appropriate property in the `set_sw_property` command, as shown in [Table 15-7](#).

Table 15-7. Callback Properties

Property as specified in <code>set_sw_property</code>	Run time	Scope	Callback Arguments
<code>initialization_callback</code>	Initialization	Component instance	Component instance name
<code>validation_callback</code>	Validation	Component instance	Component instance name
<code>generation_callback</code>	Generation	Component instance	Component instance name, BSP generate target directory, driver BSP subdirectory (1)
<code>class_initialization_callback</code>	Initialization	Component class	Driver class name
<code>class_validation_callback</code>	Validation	Component class	Driver class name
<code>class_generation_callback</code>	Generation	Component class	Driver class name, BSP generate target directory, driver BSP subdirectory (1)
Note to Table 15-7: (1) The BSP subdirectory into which the driver or package files are copied			

Tcl callbacks have access to a specialized set of commands, described in this section. In addition, Tcl callbacks can use any read-only BSP setting Tcl command.



Refer to [“Tcl Commands for BSP Settings” on page 15-79](#) for details about BSP setting Tcl commands.



When a Tcl callback creates a setting, it can specify the value. However, callbacks cannot change the value of a pre-existing setting.

add_class_sw_setting

Usage

```
add_class_sw_setting <setting-name> <setting-type>
```

Options

- *<setting-name>*: Name of the setting to persist in the BSP settings file. This is prepended with the driver class name associated with this callback script
- *<setting-type>*: Type of the setting to persist in the BSP settings file.

Description

Creates a BSP setting that is associated with a particular software driver element class. The `set_class_sw_setting_property` command is required to set the values for fields pertaining to a BSP software setting definition. This command is only valid for a callback script. A callback script is set in the driver's `*_sw.tcl` file, using the command `set_sw_property callback_source_file <filename>`.

Example

```
add_class_sw_setting MY_FAVORITE_SETTING String
```

add_class_systemh_line

Usage

```
add_class_systemh_line <macro-name> <macro-value>
```

Options

- *<macro-name>*: Macro to be added to the system.h file for the generated BSP
- *<macro-value>*: Value associated with the macro-name to be added to the system.h file for the generated BSP

Description

This adds a system.h assignment or macro during a driver callback execution. The BSP typically uses this during the generate phase depending on the generator. This command is only valid for a callback script. A callback script is set in the driver's `*_sw.tcl` file, using the command `set_sw_property callback_source_file <filename>`.

Example

```
add_class_systemh_line MY_MACRO "Macro_Value";
```

add_module_sw_property

Usage

```
add_module_sw_property <property-name> <property-value>
```

Options

- *<property-name>*: Name of the property to add to the BSP for a module instance
- *<property-value>*: Value of the property to add to the BSP for a module instance

Description

This adds a software property to the BSP driver of this module instance. The BSP typically uses this during the generate phase depending on the generator. This command is only valid for a callback script. A callback script is set in the driver's *_sw.tcl file, using the command `set_sw_property callback_source_file <filename>`.

Example

```
add_module_sw_setting MY_FAVORITE_SETTING String
```

add_module_sw_setting**Usage**

```
add_module_sw_setting <setting-name> <setting-type>
```

Options

- *<setting-name>*: Name of the setting to persist in the BSP settings file. This is prepended with the module name associated with this callback script
- *<setting-type>*: Type of the setting to persist in the BSP settings file.

Description

Creates a BSP setting that is associated with a particular instance of hardware module in a SOPC system. The `set_module_sw_setting_property` command is required to set the values for fields pertaining to a BSP software setting definition. This command is only valid for a callback script. A callback script is set in the driver's *_sw.tcl file, using the command `set_sw_property callback_source_file <filename>`.

Example

```
add_module_sw_setting MY_FAVORITE_SETTING String
```

add_module_systemh_line**Usage**

```
add_module_systemh_line <macro-name> <macro-value>
```

Options

- *<macro-name>*: Macro to be added to the system.h file for the generated BSP
- *<macro-value>*: Value associated with the macro-name to be added to the system.h file for the generated BSP

Description

This adds a system.h assignment or macro during a driver callback execution. The BSP typically uses this during the generate phase depending on the generator. This command is only valid for a callback script. A callback script is set in the driver's *_sw.tcl file, using the command `set_sw_property callback_source_file <filename>`.

Example

```
add_module_systemh_line MY_MACRO "Macro_Value";
```

add_systemh_line

Usage

```
add_systemh_line <sw> <name> <value>
```

- **<sw>**: The software (OS) that the **system.h** text is associated with
- **<name>**: Name of macro to write into **system.h** (left-hand side of #define)
- **<value>**: Name of value to assign to macro in **system.h** (right-hand side of #define)

Description

Adds a line of text to the **system.h** file. The **<sw>** argument is the name of the software type (typically an operating system name) that the **system.h** text applies to. In the context of an operating system Tcl script, the name in the `create_os <name>` command must be used. The text is a name-value pair that creates a macro (#define statement) in the **system.h** file.



This command can only be used by Tcl scripts that are registered to run at BSP generation time by an operating system.

Example

```
add_systemh_line UCOSII OS_TICKS_PER_SEC 100
```

get_class_peripheral

Usage

```
get_class_peripheral <instance-name> <irq-number>
```

Options

- **<instance-name>**: Name of EIC module instance to find connected peripheral for.
- **<irq-number>**: IRQ number to locate connected peripheral device

Description

This command is used on an EIC instance callback to obtain a peripheral slave descriptor connected to a specific IRQ port number. This command is only valid for a callback script.

Example

```
get_class_peripheral eic_1 $irq_2;
```

get_module_assignment

Usage

```
get_module_assignment <assignment-name>
```

Options

- *<assignment-name>*: Name of the module assignment to retrieve the value for, as defined for the module instance in the **.sopcinfo** file

Description

Given a module assignment key, return the assignment value of a module associated with the callback script using this command. The callback script must be set in the ***_sw.tcl** file using the following command:

```
set_sw_property callback_source_file <filename>
```

Example

```
puts [get_module_assignment embeddedsw.configuration.isMemoryDevice]
```

Returns:

```
true
```

get_module_name**Usage**

```
get_module_name
```

Options

None

Description

Returns the name of the module associated with the callback script using this command. The callback script must be set in the ***_sw.tcl** file using the following command:

```
set_sw_property callback_source_file <filename>
```

Example

```
puts [get_module_name]
```

Returns:

```
jtag_uart
```

get_module_peripheral**Usage**

```
get_module_peripheral <irq-number>
```

Options

- *<irq-number>*: IRQ number to locate connected peripheral device

Description

This command is used on an EIC instance callback to obtain a peripheral slave descriptor connected to a specific IRQ port number. This command is only valid for a callback script.

Example

```
get_module_peripheral 2;
```

get_module_sw_setting_value

Usage

```
get_module_sw_setting_value <setting-name>
```

Options

- *<setting-name>*: Name of the module software setting to retrieve the value for, as defined by the `add_module_sw_setting` command.

Description

Given a module software setting name, return the setting value. The callback script using this command must be set in the `*_sw.tcl` file using the following command:

```
set_sw_property callback_source_file <filename>
```

You can use this command in a generation or validation callback to retrieve the current value of a setting created in an initialization callback.

Example

```
puts [get_module_sw_setting_value MY_SETTING]
```

Returns:

```
"My setting value"
```

get_peripheral_property

Usage

```
get_peripheral_property <slave-descriptor> <property-name>
```

Options

- *<slave-descriptor>*: Slave descriptor of a connected peripheral device
- *<property-name>*: Property name to query from the connected peripheral device

Description

This command is used on an EIC instance callback to obtain a connected peripheral property value. This command is only valid for a callback script. A callback script is set in the driver's `*_sw.tcl` file, using the command `set_sw_property callback_source_file <filename>`.

Example

```
get_peripheral_property jtag_uart supports_preemption;
```

remove_class_systemh_line

Usage

```
remove_class_systemh_line <macro-name>
```

Options

- *<macro-name>*: Macro to be removed to the system.h file for the generated BSP

Description

This removes a system.h assignment or macro during a driver callback execution. The BSP typically uses this during the generate phase depending on the generator. This command is only valid for a callback script. A callback script is set in the driver's *_sw.tcl file, using the command `set_sw_property callback_source_file <filename>`.

Example

```
remove_class_systemh_line MY_MACRO;
```

remove_module_systemh_line

Usage

```
remove_module_systemh_line <macro-name>
```

Options

- *<macro-name>*: Macro to be removed to the system.h file for the generated BSP

Description

This removes a system.h assignment or macro during a driver callback execution. The BSP typically uses this during the generate phase depending on the generator. This command is only valid for a callback script. A callback script is set in the driver's *_sw.tcl file, using the command `set_sw_property callback_source_file <filename>`.

Example

```
remove_module_systemh_line MY_MACRO;
```

set_class_sw_setting_property

Usage

```
set_class_sw_setting_property <setting-name> <property> <value>
```

Options

- *<setting-name>*: Name of the setting to persist in the BSP settings file associated with the driver class of this callback script
- *<property>*: Name of the setting property to update
- *<value>*: Value of the setting property to update

Description

Update a driver class software setting property. The setting must be added using the `add_class_sw_setting` command before calling this method. This command is only valid for a callback script. A callback script is set in the driver's *_sw.tcl file, using the command `set_sw_property callback_source_file <filename>`.

You can set the following setting properties:

- destination
- identifier
- value
- default_value
- description
- restrictions
- group

Example

```
set_class_sw_setting_property MY_FAVORITE_SETTING default-value '42'
```

set_module_sw_setting_property

Usage

```
set_module_sw_setting_property <setting-name> <property> <value>
```

Options

- *<setting-name>*: Name of the setting to persist in the BSP settings file associated with the SOPC module of this callback script
- *<property>*: Name of the setting property to update
- *<value>*: Value of the setting property to update

Description

Update a module's software setting property. The setting must be added using the `add_module_sw_setting` command before calling this method. This command is only valid for a callback script. A callback script is set in the driver's `*_sw.tcl` file, using the command `set_sw_property callback_source_file <filename>`.

You can set the following setting properties:

- destination
- identifier
- value
- default_value
- description
- restrictions
- group

Example

```
set_module_sw_setting_property MY_FAVORITE_SETTING default-value '42'
```

Tcl Commands for Drivers and Packages

This section describes the tools that you use to specify and manipulate the settings and characteristics of a custom software package or driver. Typically, when creating a custom software package or device driver, or importing a package or driver from another development environment, you need these more powerful tools. To manipulate settings on existing software packages and device drivers, refer to [“Settings Managed by the Software Build Tools” on page 15-37](#) and [“Tcl Commands for BSP Settings” on page 15-79](#).

A device driver and a software package are both collections of source files added to the BSP. A device driver is associated with a particular component class (for example, `altera_avalon_jtag_uart`). A software package is not associated with any particular component class, but implements a functionality such as TCP/IP.

To define a device driver or software package, you create a Tcl script defining its characteristics. This section describes the Tcl commands available to define device drivers and software packages.



For more information about creating Tcl scripts, refer to [“Tcl Scripts for BSP Settings”](#) in the *Nios II Software Build Tools* chapter of the *Nios II Software Developer's Handbook*.

The following commands are available for device driver and software package creation:

- [“add_sw_property” on page 15-116](#)
- [“add_sw_setting” on page 15-118](#)
- [“create_driver” on page 15-120](#)
- [“create_os” on page 15-121](#)
- [“create_sw_package” on page 15-121](#)
- [“set_sw_property” on page 15-122](#)

add_sw_property

Usage

```
add_sw_property <property> <value>
```

Options

- `<property>`: Name of property.
- `<value>`: Value assigned, or appended to the current value.

Description

This command defines a property for a device driver or software package. A property is a list of values (for example, a list of file names). The `add_sw_property` command defines a property if it is not already defined. The command appends a new value to the list of values if the property is already defined.

In the case of a property consisting of a file name or directory name, use a relative path. Specify the path relative to the directory containing the Tcl script.

This command supports the following properties:

- **asm_source**—Adds a Nios II assembly language source file (**.s** or **.S**) to BSPs containing your package. **nios2-bsp-generate-files** copies assembly source files into a BSP and adds them to the source build list in the BSP makefile. This property is optional.
- **c_source**—Adds a C source file (**.c**) to BSPs containing your package. **nios2-bsp-generate-files** copies C source files into a BSP and adds them to the source build list in the BSP makefile. This property is optional.
- **cpp_source**—Adds a C++ source file (**.cpp**, **.cc**, or **.cxx**) to BSPs containing your package. **nios2-bsp-generate-files** copies the C++ source files into a BSP and adds them to the source build list in the BSP makefile. This property is optional.
- **include_source**—Adds an include file (typically **.h**) to BSPs containing your package. **nios2-bsp-generate-files** copies include files into a BSP, but does not add them to the generated makefile. This property is optional.
- **include_directory**—Adds a directory to the **ALT_INCLUDE_DIRS** variable in the BSP's **public.mk** file. Adding a directory to **ALT_INCLUDE_DIRS** allows all source files to find include files in this directory. **add_sw_property** adds the path to the generated public makefile shared by the BSP and applications or libraries referencing it. **add_sw_property** compiles all files with the include directory listed in the compiler arguments.
This property is optional.
- **lib_source**—Adds a precompiled library file (typically **.a**) to each BSP containing the driver or package. **nios2-bsp-generate-files** copies the precompiled library file into the BSP directory and adds both the library file name and the path (required to locate the library file) into to the BSP's **public.mk** file. Applications using the BSP link with the library file.
The library file name must conform to the following pattern:
lib<name>.a
where **<name>** is a nonempty string.
Example:
`add_sw_property lib_source HAL/lib/libcomponent.a`
This property is optional.
- **specific_compatible_hw_version**—Specifies that the device driver only supports the specified component hardware version. See the **version** property of the **set_sw_property** command for information about version strings. This property applies only to device drivers (see the **create_driver** command), not to software packages. If your driver supports all versions of a peripheral after a specific release, use the **set_property min_compatible_hw_version** command instead.
This property is optional.
This property is only available for device drivers.
- **supported_bsp_type**—Adds a specific BSP type (operating system) to the list of supported operating systems that the driver or software package supports. Specify **HAL** if the software supports the Altera HAL, or operating systems that extend it. If your software is operating system-neutral and works on multiple HAL-based operating systems, state **HAL** only. If your software or driver contains code that depends on a particular operating system, state compatibility with that operating system only, but not **HAL**.
The name of another operating system to support must match the name of the operating system exactly. This operating system name string is the same as that

used to create a BSP with the `nios2-bsp-*` commands, as well as in the `.tcl` script that describes the operating system, in its `create_os` command.

When you create a BSP with an operating system that extends HAL, such as UCOSII, and the BSP tools select a driver for a particular hardware module, precedence is given to drivers which state compatibility with a that specific operating system (OS) before a more generic driver stating HAL compatibility. This property is only available for device drivers and software packages. This property must be set to at least one operating system.

- `alt_cppflags_addition`—Adds a line of arbitrary text to the `ALT_CPPFLAGS` variable in the BSP **public.mk** file. This technique can be useful if you wish to have a static compilation flag or definition that all BSP, application, and library files receive during software build. This property is optional.
- `excluded_hal_source`—Specifies a file to exclude from the a BSP generated with an operating system that extends HAL. The value is the path to a BSP file to exclude, with respect to the BSP root. This property is optional.
- `systemh_generation_script`—Specifies a `.tcl` script to execute during generation of the BSP **system.h** file. This script runs with the tcl commands available to other BSP settings tcl scripts, and allow you to influence the contents of the **system.h** file. This property is available only to operating systems, created with the `create_os` command. This property is optional.

add_sw_setting

Usage

```
add_sw_setting <type> <destination> <displayName>
               <identifier> <value> <description>
```

Options

- `<type>`: Setting type - Boolean, QuotedString, UnquotedString.
- `<destination>`: The destination BSP file associated with the setting, or the module generator that processes this setting.
- `<displayName>`: Setting name.
- `<identifier>`: Name of the macro created for a generated destination file.
- `<value>`: Default value of the setting.
- `<description>`: Setting description.

Description

This command creates a BSP setting associated with a software package or device driver. The setting is available whenever the software package or device driver is present in the BSP. **nios2-bsp-generate-files** converts the setting and its value into either a C preprocessor macro or BSP makefile variable. `add_sw_setting` passes macro definitions to the compiler using the `-D` command-line option, or adds them to the **system.h** file as `#define` statements.

The setting only exists once even if there are multiple instances of a software package. Set or get the setting with the `--set` and `--get` command-line options of the **nios2-bsp**, **nios2-bsp-create-settings**, **nios2-bsp-query-settings**, and **nios2-bsp-update-settings** commands. You can also use the BSP Tcl commands `set_setting` and `get_setting` to set or get the setting. The value of the setting persists in the BSP settings file.

To create a setting, you must define each of the following parameters:

- **type**—This parameter formats the setting value during BSP generation. The following supported types and usage restrictions apply:
 - **boolean_define_only**—Defines a macro if the setting's value is 1 or true. Example: `#define LCD_PRESENT`. No macro is defined if the setting's value is 0 or false. This setting type supports the `system_h_define` and `public_mk_define` destinations, defined below.
 - **boolean**—Defines a macro or makefile variable to 1 (if the value is 1 or true) or 0 (if the value is 0 or false). Example: `#define LCD_PRESENT 1`. This type supports all destinations.
 - **character**—Defines a macro with a single character with single quotes around the character. Example: `#define DELIMITER ':'`. This type supports the `system_h_define` destination, defined below.
 - **decimal_number**—Decimal numbers define a macro or makefile variable with an unquoted decimal (integer) number. Example: `#define NUM_COPROCESSORS 3`. This type supports all destinations.
 - **double**—Double numbers have a macro name and setting value in the destination file including decimal point. Example: `#define PI 3.1416`. This type supports the `system_h_define` destination, defined below.
 - **float**—Float numbers have a macro name and setting value in the destination file including decimal point and `f` character. Example: `#define PI 3.1416f`. This type supports the `system_h_define` destination, defined below.
 - **hex_number**—Hex numbers have a macro name and setting value in the destination file with `0x` prepended to the value. Example: `#define LCD_SIZE 0x1000`. This type supports the `system_h_define` destination, defined below.
 - **quoted_string**—Quoted strings always have the macro name and setting value added to the destination files. In the destination, the setting value is enclosed in quotation marks. Example:
`#define DFLT_ERR "General error"`
 If the setting value contains white space, you must also place quotation marks around the value string in the Tcl script.
 This type supports the `system_h_define` destination, defined below.
 - **unquoted_string**—Unquoted strings define a macro or makefile variable with setting name and value in the destination file. In the destination file, the setting value is not enclosed in quotation marks. Example:
`#define DFLT_ERROR Error`
 This type supports all destinations.

- **destination**—The destination parameter specifies where `add_sw_setting` puts the setting in the generated BSP. `add_sw_settings` supports the following destinations:
 - **system_h_define**—With this destination, `add_sw_settings` formats settings as `#define <setting name> [<setting value>]` macros in the **system.h** file
 - **public_mk_define**—With this destination, `add_sw_settings` formats settings as `-D<setting name>[=<setting value>]` additions to the `ALT_CPPFLAGS` variable in the BSP **public.mk** file. **public.mk** passes the flag to the C preprocessor for each source file in the BSP, and in applications and libraries using the BSP.
 - **makefile_variable**—With this destination, `add_sw_settings` formats settings as makefile variable additions to the BSP makefile. The variable name must be unique in the makefile.
- **displayName**—The name of the setting. Settings exist in a hierarchical namespace. A period separates levels of the hierarchy. Settings created in your Tcl script are located in the hierarchy under the driver or software package name you specified in the `create_driver` or `create_sw_package` command. Example: `my_driver.my_setting`. The Nios II SBT adds the hierarchical prefix to the setting name.
- **identifier**—The name of the macro or makefile variable being defined. In a setting added to the **system.h** file at generation time, this parameter corresponds to the text immediately following the `#define` statement.
- **value**—The default value associated with the setting. If you do not assign a value to the option, its value is this default value. Valid initial values are `true`, `1`, `false`, and `0` for boolean and `boolean_define_only` setting types, a single character for the `character` type, integer numbers for the `decimal_number` setting type, integer numbers with or without a `0x` prefix for the `hex_number` type, numbers with decimals for `float_number` and `double_number` types, or an arbitrary string of text for quoted and unquoted string setting types. For string types, if the value contains any white space, you must enclose it in quotation marks.
- **description**—Descriptive text that is inserted along with the setting value and name in the **summary.html** file. You must enclose the description in quotation marks if it contains any spaces. If the description includes any special characters (such as quotation marks), you must escape them with the backslash (`\`) character. The description field is mandatory, but can be an empty string (`" "`).

create_driver

Usage

```
create_driver <name>
```

Options

- **<name>**: Name of device driver.

Description

This command creates a new device driver instance available for the Nios II SBT. This command must precede all others that describe the device driver in its Tcl script. You can only have one `create_driver` command in each Tcl script. If the `create_driver` command appears in the Tcl script, the `create_sw_package` and `create_os` commands cannot appear.

The name argument is usually distinct from all other device drivers and software packages that the SBT might locate. You can specify driver name identical to another driver if the driver you are describing has a unique version number assignment.

If your driver differs for different operating systems, you need to provide a unique name for each BSP type.

This command is required, unless you use the `create_sw_package` or `create_os` commands, as appropriate.

create_os

Usage

```
create_os <name>
```

Options

- `<name>`: Name of operating system (BSP type).

Description

This command creates a new operating system (OS) instance (also known as a BSP type) available for the Nios II BSP tools. This command must precede all others that describe the OS in its Tcl script. You can only have one `create_os` command in each Tcl script. If the `create_os` command appears in the Tcl script, the `create_driver` or `create_sw_package` commands cannot appear.

The name argument is usually distinct from all other operating systems that the SBT might locate. You can specify an OS name identical to OS if the OS you are describing has a unique version number assignment.

This command is required, unless you use the `create_driver` or `create_sw_package` commands, as appropriate.

create_sw_package

Usage

```
create_sw_package <name>
```

Options

- `<name>`: Name of the software package.

Description

This command creates a new software package instance available for the Nios II SBT. This command must precede all others that describe the software package in its Tcl script. You can only have one `create_sw_package` command in each Tcl script. If the `create_sw_package` command appears in the Tcl script, the `create_driver` or `create_os` commands cannot appear.

The name argument is usually distinct from all other device drivers and software packages that the SBT might locate. You can specify a name identical to another software package if the software package you are describing has a unique version number assignment.

If your software package differs for different operating systems, you need to provide a unique name for each BSP type.

This command is required, unless you use the `create_driver` or `create_os` commands, as appropriate.

set_sw_property

Usage

```
set_sw_property <property> <value>
```

Options

- `<property>`: Type of software property being set.
- `<value>`: Value assigned to the property.

Description

Sets the specified value to the specified property. The properties this command supports can only hold a single value. This command overwrites the existing (or default) contents of a particular property with the specified value. This command applies to device drivers and software packages.

This command supports the following properties:

- `hw_class_name`—The name of the hardware class which your device driver supports. The hardware class name is also the **Component Name** shown in the Component Editor. Example: `altera_avalon_uart`. This property is only available for device drivers. This property is required for all drivers.
- `version`—The version number of this package. `set_sw_property` uses version numbers to determine compatibility between hardware (peripherals) and their software (drivers), as well as to choose the most recent software or driver if multiple compatible versions are available. A version can be any alphanumeric string, but is usually a major and one or more minor revision integers. The dot (.) character separates major and minor revision numbers. Examples: `9.0`, `5.0sp1`, `3.2.11`. This property is optional, but recommended. If you do not specify a version, the newest version of the package is used.

- **min_compatible_hw_version**—Specifies that the device driver supports the specified hardware version, or all greater versions. This property is only available for device drivers. If your device driver supports only one or more specific versions of a hardware class, use the `add_sw_property specific_compatible_hw_version` command instead. See the `version` property documentation for information about version strings. This property is optional. This property is only available for device drivers.
- **auto_initialize**—Boolean value that specifies **alt_sys_init.c** needs to initialize your package. If enabled, you must provide a header file containing `INSTANCE` and `INIT` macros per the instructions in the *Nios II Software Developer's Handbook*. This property is optional; if unspecified, **alt_sys_init.c** does not contain references to your driver or software. This property is only available for device drivers and software packages.
- **bsp_subdirectory**—Specifies the top-level directory where **nios2-bsp-generate-files** copies all source files for this package. This property is a path relative to the top-level BSP directory. This property is optional; if unspecified, **nios2-bsp-generate-files** copies the driver or software package into the **drivers** subdirectory of any BSP including this software.
- **alt_sys_init_priority**—This property assigns a priority to the software package or device driver. The value of this property must be a positive integer. Use this property to customize the order of macro calls in the BSP **alt_sys_init.c** file. Specifying the priority is useful if your software or driver must be initialized before or after other software in the system. For example, your driver might depend on another driver already being initialized. This property is optional. The default priority is 1000. This property is only available for device drivers and software packages.
- **display_name**—This property is used for user interfaces and other tools that wish to show a human-readable name to identify the software being described in the `.tcl` script. `display_name` is set to a few words of text (in quotes) that name your software. For example: Altera Nios II driver. This property is optional. If not set, tools that attempt to use the display name use the package name created with the appropriate `create_` command.
- **extends_bsp_type**—This property specifies which BSP type that an operating system (created with the `create_os` command) extends (if any). Currently, only the Altera HAL (HAL) is supported. This command is required for all operating systems that wish to use HAL-compatible generators in the Nios II BSP tools. It is also required for operating systems that require the Altera HAL, device driver, or software package source files that are HAL compatible in BSPs created with that operating system. An operating system that extends HAL is presumed to be compatible with device drivers that support HAL. This command is only available for operating systems.
- **callback_source_file**—This property specifies a Tcl source file containing callback functions.

- `initialization_callback`—This property specifies the name of a Tcl callback function which is intended to run in the following environment:
 - Run time: initialization
 - Scope: component instance
 - Function argument(s): component instance name
- `validation_callback`—This property specifies the name of a Tcl callback function which is intended to run in the following environment:
 - Run time: validation
 - Scope: component instance
 - Function argument(s): component instance name
- `generation_callback`—This property specifies the name of a callback function which is intended to run in the following environment:
 - Run time: generation
 - Scope: component instance
 - Function argument(s): component instance name, BSP generate target directory, driver BSP subdirectory
- `class_initialization_callback`—This property specifies the name of a callback function which is intended to run in the following environment:
 - Run time: initialization
 - Scope: component instance
 - Function argument(s): driver class name
- `class_validation_callback`—This property specifies the name of a callback function which is intended to run in the following environment:
 - Run time: validation
 - Scope: component instance
 - Function argument(s): driver class name
- `class_generation_callback`—This property specifies the name of a callback function which is intended to run in the following environment:
 - Run time: generation
 - Scope: component instance
 - Function argument(s): driver class name, BSP generate target directory, driver BSP subdirectory
- `supported_interrupt_apis`—Specifies the interrupt API that the device driver supports. Specify `legacy_interrupt_api` if the device driver supports the legacy API only or `enhanced_interrupt_api` if the device driver supports the enhanced API only. Specify both using a quoted list if the device driver supports both APIs.

If you do not specify which API your device driver supports, the Nios II SBT assumes that only the legacy interrupt API is supported. The Nios II SBT analyzes this property for each driver in the system to determine the appropriate API to be used in the system.



This property is only available for device drivers.



For more information about the legacy and enhanced APIs, refer to “Exception Handling” in the *Nios II Software Build Tools* chapter of the *Nios II Software Developer’s Handbook*.

- `isr_preemption_supported`—Specify true if your device driver ISR can be preempted by a higher priority ISR. If you do not specify whether ISR preemption is supported, the Nios II SBT assumes that your device driver does not support preemption. If your driver does not have an ISR, but the associated device has an interrupt port, you can set this property to true.



This property is valid for operating systems and device drivers.

Software Build Tools Path Names

There are some restrictions on how you can specify file paths when working with the Nios II SBT. The tools are designed for the maximum possible compatibility with a variety of computing environments. By following the restrictions in this section, you can ensure that the build tools work smoothly with other tools in your tool chain.

Command Arguments

Many Nios II software build tool commands take file name and directory path arguments. You can provide these arguments in any of several supported cross-platform formats. The Nios II SBT supports the following path name formats:

- **Quoted Windows**—A drive letter followed by a colon, followed by directory names delimited with backslashes, surrounded by double quotes. Example of a quoted Windows absolute path:

```
"c:\altera\72\nios2eds\examples\verilog\niosII_cyclone_1c20\standard"
```

Quoted Windows relative paths omit the drive letter, and begin with two periods followed by a backslash. Example:

```
"..\niosII_cyclone_1c20\standard"
```

- **Escaped Windows**—The same as quoted Windows, except that each backslash is replaced by a double backslash, and the double quotes are omitted. Examples:

```
c:\\altera\\72\\nios2eds\\examples\\verilog\\niosII_cyclone_1c20\\standard  
..\\niosII_cyclone_1c20\\standard
```

- **Linux**—An optional forward slash, followed by directory names delimited with forward slashes. Examples:

```
/altera/72/nios2eds/examples/verilog/niosII_cyclone_1c20/standard  
verilog/niosII_cyclone_1c20/standard
```

Linux relative paths begin with two periods followed by a forward slash.
Example:

```
../niosII_cyclone_1c20/standard
```

- **Mixed**—The same as quoted Windows, except that each backslash is replaced by a forward slash, and the double quotes are omitted. Examples:

```
c:/altera/72/nios2eds/examples/verilog/niosII_cyclone_1c20/standard
../niosII_cyclone_1c20/standard
```

- **Cygwin**—An absolute Cygwin path consists of the pseudo-directory name "/cygdrive/", followed by the lower case Windows drive name, followed by directory names delimited with forward slashes. Example:

```
/cygdrive/c/altera/72/nios2eds/examples/verilog/niosII_cyclone_1c20/standard
```

Cygwin relative paths are the same as Linux relative paths. Example:

```
../niosII_cyclone_1c20/standard
```

The Nios II SBT accepts both relative and absolute path names.

Table 15-8 shows the supported path name formats for each platform, for Nios II SBT utilities and makefiles.

Table 15-8. Path Name Format Support

Context	Formats supported on Linux (1)	Formats supported on Windows with Cygwin
Utilities and scripts	Linux	<ul style="list-style-type: none"> ■ Quoted Windows (2) ■ Mixed (2) ■ Escaped Windows (2) ■ Cygwin
Makefiles	Linux	<ul style="list-style-type: none"> ■ Mixed (3) ■ Cygwin (3)
Notes to Table 15-8: <p>(1) These rules apply to any Unix-like platform.</p> <p>(2) These rules apply to other Unix-like shells running on Windows. The Nios II Command Shell, provided with the Nios II EDS, is based on Cygwin. Examples in this chapter are designed for the Nios II Command Shell.</p> <p>(3) The build tools automatically convert path names to Cygwin format</p>		

Object File Directory Tree

The makefile created by the Nios II SBT creates a new directory tree for generated object files. To the extent possible, the object file directory tree retains the structure of the corresponding source directory.

For example, if you specify the path to a source file as

```
src/util/special/tools.c
```

the makefile places the corresponding object code in

```
obj/util/special/tools.o
```



The object file directory structure is illustrated in “Nios II Embedded Software Projects” in the *Nios II Software Build Tools* chapter of the *Nios II Software Developer’s Handbook*.

The makefile does not create object directories outside the project directory root. If the source file path you specify is a relative path beginning with “..”, the Nios II SBT flattens the path name prior to creating the object directory structure.

For example, if you specify the path to a source file as

```
../special/tools.c
```

the makefile places the corresponding object code in

```
obj/tools.o
```

If you specify an absolute path to source files under Cygwin, the Nios II SBT creates the obj directory structure as if you had used the Cygwin form of the path name. For example, if you specify the path to a source file as

```
c:/dev/app/special/tools.c
```

the Nios II SBT places the corresponding object code in

```
obj/cygdrive/c/dev/app/special/tools.o
```

Document Revision History

Table 15-9 shows the revision history for this document.

Table 15-9. Document Revision History (Part 1 of 2)

Date	Version	Changes
May 2011	11.0.0	<ul style="list-style-type: none"> ■ Introduction of Qsys system integration tool ■ New Tcl commands added ■ Recommend leaving <code>hal.enable_sopc_sysid_check</code> enabled
February 2011	10.1.0	<ul style="list-style-type: none"> ■ Correction to <code>add_memory_device</code> Tcl command arguments. ■ New functionality in <code>nios2-bsp-create-settings</code> command. ■ Removed “Referenced Documents” section.
July 2010	10.0.0	<ul style="list-style-type: none"> ■ Update documentation of <code>hal.enable_small_c_library</code> setting. ■ Describe new BSP Tcl commands: <ul style="list-style-type: none"> ■ <code>add_memory_device</code> ■ <code>set_ignore_file</code> ■ Correct missing properties in <code>set_sw_property</code> Tcl command: <ul style="list-style-type: none"> ■ <code>supported_interrupt_apis</code> ■ <code>isr_preemption_supported</code>

Table 15–9. Document Revision History (Part 2 of 2)

Date	Version	Changes
November 2009	9.1.0	<ul style="list-style-type: none"> ■ Support for external interrupt controller. ■ Add documentation for the following utilities: <ul style="list-style-type: none"> ■ nios2-lib-update-makefile ■ nios2-app-update-makefile ■ nios2-convert-ide2sbt ■ nios2-example-sw-create ■ nios2-elf-insert ■ nios2-elf-query ■ nios2-flash-programmer ■ Add documentation for <code>--jdi</code> command-line argument. ■ Add documentation for example design scripts. ■ Add documentation for hal.log_flags setting. ■ Add documentation for interrupt stack settings. ■ Clarify information about default settings for MicroC/OS-II. ■ Clarify information about default settings for host file system. ■ Add documentation for makefile variables. ■ Add documentation for the following BSP Tcl commands: <ul style="list-style-type: none"> ■ <code>add_systemh_line</code> ■ <code>get_assignment</code> ■ <code>get_cpu_name</code> ■ <code>get_interrupt_controller_id</code> ■ <code>get_irq_interrupt_controller_id</code> ■ <code>is_connected_interrupt_controller_device</code> ■ Describe Tcl callback functions.
March 2009	9.0.0	<ul style="list-style-type: none"> ■ Reorganized and updated information and terminology to clarify role of Nios II Software Build Tools. ■ Described usage of custom Tcl scripts. ■ Corrected minor typographical errors.
May 2008	8.0.0	<ul style="list-style-type: none"> ■ Advanced exceptions added to Nios II core. ■ Instruction-related exception handling added to HAL. ■ Describe new BSP setting <code>hal.enable_instruction_related_exceptions_api</code>.
October 2007	7.2.0	Initial release. Reference material moved here from former <i>Nios II Software Build Tools</i> chapter.