



This document addresses known errata and documentation issues for the Nios® II Embedded Design Suite (EDS) version 7.0. Errata are functional defects or errors, which might cause the product to deviate from published specifications. Documentation issues include errors, unclear descriptions, or omissions from current published specifications or product documents. Errata items discovered after the release of Nios II EDS version 7.0 are marked with the date the items were added to this document.

Table of Contents:

Nios II Processor Core	4
Nios II/e core burst support.....	4
Interrupt vector custom instruction in multi-processor systems.....	4
"nios_cpu: Unknown Break Location nios_cpu/jtag_debug_module" error message.....	4
VHDL sensitivity list warnings during Quartus II synthesis.....	4
Double-precision Floating-Point operations with floating-point custom instructions.....	5
Peripherals	5
JTAG UART is unstable after device-wide reset.....	5
Host Platform	5
Windows XP: Internal error when repeatedly stepping into with debugger.....	5
Linux: Debugging with the Nios II ISS target can cause a process leak on Linux.....	5
Linux: The Quartus II stand-alone programmer is not supported on Linux.....	5
Windows: Frisk antivirus software causes SOPC Builder and Nios II Command Shell to be unresponsive.....	6
Device	6
Stratix® II EP2S60 ES devices cannot use MRAM byte enables.....	6
Nios II IDE	6
Building Projects.....	6
Incorrect stack checking when interrupt vector custom instruction is used.....	6
Unable to save files containing copyright symbol.....	6
Illegal project location error when creating new project.....	7
Compilation error "UNDEFINED VARIABLE %STACK_POINTER%" when building a project.....	7
When building a project, the Nios II IDE reports problems, but the build output in the Console doesn't contain any errors.....	7
Makefile reports incorrect number of bytes free for the stack and heap, if the heap is in a different memory region than the stack.....	7
Nios II IDE unnecessarily updates the SOPC Builder system file (.ptf).....	7
Incorrect address assignment for dual-port memory mastered by two different masters.....	7
Build errors after changing component names in SOPC Builder.....	8
Debugging Projects.....	8
MicroC/OS-II "Source not found" error occurs when debugging, after the Nios II IDE stops at a breakpoint.....	8
The Restart command on the Run menu does not work.....	8
Programs that interact with a terminal Console on Windows don't work in the Nios II IDE v6.1 and later, but used to work in v6.0 or earlier.....	9
"Step failed. Target is not responding (timed out)" error message.....	9
Incorrect breakpoint filtering on threads.....	9
Uninitialized Memory Error when executing from ISS.....	9
Nios2-gdb-server fails to terminate after setting a watch point.....	9
Watchpoints do not work when set on variables whose size are not 32-bits.....	9
Debugger cannot step into __sflags, and continues execution instead.....	9
Missing traced load/store instruction and data in the Trace view.....	10

Cannot use watchpoints in the Nios II IDE when the FS2 console is open.....	10
Breakpoints on adjacent lines of assembly fail to halt the processor	10
Navigating Projects.....	10
Internal Error when double-clicking on objdump file.....	10
Nios II IDE doesn't recognize when an elf file is built outside the IDE.....	11
After switching workspaces, the IDE appears to freeze while displaying the Nios II IDE splash screen	11
C/C++ Scanner does not understand certain C/C++ constructs	11
C2H Compiler	11
C2H Compiler out-of-memory error.....	11
C2H Compiler doesn't accelerate subfunctions located in a separate file	11
Incorrect results from logical or conditional operation with side-effects.....	12
Closed system library while working with the C2H Compiler	12
Launch SOPC Builder button in C2H view	12
Build clean causes build failure	12
Multiple clock domains causes hardware accelerator to fail.....	12
Comma operators are not supported	12
Array elements of structs do not copy correctly.....	13
Simulating on ISS is not supported.....	13
Hardware accelerators remain after deleting the software project	13
Changing build configurations produces unexpected results	14
Flash Programmer	14
'No such file or directory' error	14
elf2flash elf size limit	14
Verifying flash with the nios2-flash-programmer command fails	14
Using the Nios II Flash Programmer with an override file stored in <Nios II EDS path>/bin might generate an error:.....	15
When used with a multiprocessor Nios II system, the Nios II Flash Programmer might produce an error:.....	15
Download Cables & Debug Hardware	15
Communication errors during run/debug sessions using older download cables	15
Development Boards	15
Intermittent failures while accessing CompactFlash card	16
Toolchain (gcc, gdb, etc.).....	16
Breakpoints in C++ constructors fail to halt the processor	16
Target Software	16
Lightweight IP (lwIP) failure when using DHCP	16
stdin, stdout, and stderr don't work in MicroC/OS-II applications when using small C library option.....	17
malloc(), realloc() failures with MicroC/OS-II	17
cout from MicroC-OS/II task will not send data to STDOUT	17
Problems using HAL drivers with Toshiba Flash	17
Creating new custom HAL components	17
Legacy SDK	18
SOPC Builder and Quartus II Software	18
Example Designs	18
Hardware Designs.....	18
Incorrect SSRAM PLL phase shift in Stratix II 2s60 RoHS and Cyclone™ II 2C35 standard and full_featured designs	18
Software Designs.....	19
RAM test failure when running Memory Test software template on the ISS.....	19
Networking Examples.....	19
Hardware Simulation	19
Error: "UNC paths are not supported. Defaulting to Windows directory."	19
Simulation failure if reset address is set to EPCS	19

Uninitialized BSS variables in simulation	20
ModelSim fails to load large memory models	20
Documentation Issues	20
GNU assembler does not accept the --defsym flag	20
Nios II Processor Reference Handbook refers erroneously to the data bus.....	20
Contact Information	21
Revision History.....	21



For the most up-to-date errata for this release, refer to the errata sheet on the Altera® website:
http://www.altera.com/literature/es/es_nios2eds_70.pdf

Nios II Processor Core

This section lists any issues related to the Nios II processor cores.

Nios II/e core burst support

The Nios II wizard incorrectly allows you to select burst support for the instruction master in the Nios II/e core. Instruction master burst support in Nios II requires an instruction cache. The Nios II/e core does not include an instruction cache, therefore, burst support can not be enabled.

Workaround: Do not select burst support for Nios II/e cores.

Interrupt vector custom instruction in multi-processor systems

In designs containing multiple Nios II processors, enabling the interrupt vector custom instruction in more than one of the processors causes the following errors during Quartus® II compilation:

```
Error (10430): VHDL Primary Unit Declaration error at
interruptvector_cpu_1.vhd(26): primary unit "interrupt_vector" already exists in
library "work"
Error (10523): Ignored construct interruptvector_cpu_1 at
interruptvector_cpu_1.vhd(64) due to previous errors
Error: Node instance "the_interruptvector_cpu_1" instantiates undefined entity
"interruptvector_cpu_1"
```

Workaround: Apply the patch provided at http://www.altera.com/support/kdb/solutions/rd02052007_136.html, regenerate your SOPC Builder system, and recompile in Quartus II.

"nios_cpu: Unknown Break Location nios_cpu/jtag_debug_module" error message

If you modify the Nios II core in an existing system by disabling the JTAG debug module in the Nios II configuration wizard, you might receive the error message "**nios_cpu: Unknown Break Location nios_cpu/jtag_debug_module**" during SOPC Builder generation.

Workaround: Delete the Nios II core from the system and re-add it, ensuring that the JTAG debug core is disabled in the wizard.

VHDL sensitivity list warnings during Quartus II synthesis

You might receive warnings similar to the following in the Quartus II software when compiling a VHDL Nios II system containing the JTAG debug module.

```
Warning (10492): VHDL Process Statement warning at
cpuname_jtag_debug_module.vhd(254): signal "usr1" is read inside the Process
Statement but isn't in the Process Statement's sensitivity list
Warning (10492): VHDL Process Statement warning at
cpuname_jtag_debug_module.vhd(254): signal "ena" is read inside the Process
Statement but isn't in the Process Statement's sensitivity list
```

These warnings are benign and can be ignored. The warnings are a result of the ena and usr1 signals not being included in the debug module's sensitivity list.

Double-precision Floating-Point operations with floating-point custom instructions

Calls to double-precision floating-point functions in math.h return less-precise results on Nios II processors using the floating-point custom instruction. Floating-point constants are forced to single-precision when the floating-point custom instruction is present, which affects the constants for the double-precision floating-point functions in libm.

Peripherals

This section lists any issues related to the Altera embedded peripherals included in the Quartus II software.

JTAG UART is unstable after device-wide reset

If the DEV_CLRn pin on the FPGA input has been assigned (in Quartus[®] II software) to generate a device-wide reset, and the FPGA is reset while the JTAG UART is active, then the JTAG UART might become unstable.

Workaround: Do not use the DEV_CLRn function on the FPGA. Turn off the Enable device wide reset (DEV_CLRn) setting in Quartus II software.

Host Platform

This section lists any issues related specifically to the host platform.

Windows XP: Internal error when repeatedly stepping into with debugger

If you repeatedly use the Step Into command while debugging on Windows XP, you might receive the internal error, "Retrieving Children: An internal error occurred during: Retrieving Children".

Workaround: End your debug session, then re-open it and resume debugging. Reduce the frequency of using the step into command.

Linux: Debugging with the Nios II ISS target can cause a process leak on Linux

If you try to interrupt or terminate a debug session targeting the Nios II instruction set simulator (ISS), you might see an error message "Interrupt Failed or Terminate Failed". This means that the nios2-iss process failed to terminate. The debug session appears to have terminated in the IDE, but the nios2-iss process still remains alive.

Workaround: Open a command shell and kill the nios2-iss process.

Linux: The Quartus II stand-alone programmer is not supported on Linux

There is no Quartus II stand-alone programmer for Linux. As a result, in the Nios II IDE the Quartus II Programmer command on the Tools menu has no effect. The IDE does not automatically launch the programmer when you attempt to download software to a board that does not match the expected hardware image.

Workaround: Launch the Quartus II software to access the Quartus II Programmer.

Windows: Frisk antivirus software causes SOPC Builder and Nios II Command Shell to be unresponsive

The SOPC Builder and Nios II Command Shell might become unresponsive if run while the Frisk antivirus software is running.

Workaround: Turn off the Dynamic Virus Checking feature of the Frisk software before running SOPC Builder or the Nios II Command Shell.

Device

This section lists any device-related issues.

Stratix® II EP2S60 ES devices cannot use MRAM byte enables

Early shipments of the Nios II Development Kit, Stratix II Edition include an EP2S60 engineering sample (ES) device. Stratix II EP2S60 ES devices have a silicon problem that prevents the use of byte enables on MRAM blocks.

Workaround: Refer to the Stratix II FPGA Family Errata Sheet for details.

Nios II IDE

This section lists any issues relating to the Nios II IDE.

Building Projects

Incorrect stack checking when interrupt vector custom instruction is used

If you enable run-time stack checking in systems that include the interrupt vector custom instruction, the Nios II IDE might erroneously report a stack overflow or might fail to report an actual stack overflow.

Workaround: Do not enable the run-time stack checking and interrupt vector custom instruction features in the same application.

Unable to save files containing copyright symbol

On computers using character encoding other than Cp1252, the copyright symbol included in the board_diag.c file is not recognized. Because of this, the file can not be saved using the Nios II IDE.

Workaround: Change the default text file encoding setting in the Nios II IDE to Cp1252 by performing the following steps:

- In the Nios II IDE, select Preferences from the Window menu
- In the General settings, select Workspace
- Select Default (Cp 1252) in the Text File Encoding section and click OK.

Illegal project location error when creating new project

You might receive an "Illegal project location" error message in the IDE if you use the default project name when creating a project in a new workspace. If the project name exists in another workspace, the IDE might not account for that in the new workspace.

Workaround: Change the project name to a name other than the default.

Compilation error "UNDEFINED VARIABLE %STACK_POINTER%" when building a project

This error occurs if the **Use a separate exception stack option** is turned on, and the exception stack is larger than the memory available for it.

Workaround: On the system library properties page for the project, turn off the separate exception stack or reduce the **Maximum exception stack size** setting.

When building a project, the Nios II IDE reports problems, but the build output in the Console doesn't contain any errors

Some linker warnings are incorrectly reported as errors in the Nios II IDE. The Dhystone software example exhibits this behavior, and recompiling the project again makes the issue go away.

Workaround: If the Console output doesn't contain errors, then the project actually built fine. On subsequent builds, the linker step is skipped and the errors will not appear, because the project built without error previously.

Makefile reports incorrect number of bytes free for the stack and heap, if the heap is in a different memory region than the stack.

Workaround: Don't trust the heap and stack memory report from the makefile if you have placed the heap and stack in different memory regions.

Nios II IDE unnecessarily updates the SOPC Builder system file (.ptf)

The Nios II IDE opens the SOPC Builder system file (**.ptf**) by invoking SOPC Builder during certain operations, which might cause SOPC Builder to change the date stamp of the file even though the system was not modified. This might cause problems if you are using a version control system.

Workaround: If you are not using the Nios II C2H Compiler, you can change the PTF file properties to read-only to prevent the IDE from changing the file.

Incorrect address assignment for dual-port memory mastered by two different masters

If you have a dual-port memory in your Nios II system, and only the second slave port is mastered by the CPU, you might see an overlapping section error during the linking stage of building your software.

Workaround: In SOPC Builder, ensure that the first slave port of the dual-port memory is mastered by the Nios II CPU. The second port does not have to be mastered by the Nios II CPU.

Build errors after changing component names in SOPC Builder

If you rename components in the SOPC Builder system and then regenerate the SOPC Builder system, Nios II IDE system library projects based on that system will have build errors.

Workaround: After regenerating the SOPC Builder system, create a new system library project for the SOPC Builder system. Alternately, you can delete the system library project from the workspace without deleting the contents from the file system, and then re-import the project, selecting the appropriate SOPC Builder system.

Debugging Projects

MicroC/OS-II "Source not found" error occurs when debugging, after the Nios II IDE stops at a breakpoint.

You might receive a 'Source not found' error message if you step into MicroC/OS-II code or break on a program entry point while debugging your application.

Workaround: To eliminate this error, you must specify the proper source path mapping in the Nios II IDE by performing the following steps.

1. On the Windows menu, click **Preferences**. The Preferences dialog box appears.
2. Expand the **C/C++** group, then expand the **Debug** group, then click **Common Source Lookup Path**.
3. Click **Add...** .The Add Source dialog box appears.
4. Select **Path Mapping**, then click **OK**.
5. In the Preferences dialog box, select **Path Mapping: New Mapping** and click **Edit**. The Path Mappings dialog box appears.
6. Enter `cygdrive` in the **Name** field.
7. Click **Add...** . A dialog box appears, prompting you to specify the path mapping.
8. Enter `/cygdrive/c` in the **Compilation path** field and `c:/` in the **Local file system path** field.
9. Click **OK** in all open dialog boxes to return to the IDE workbench.

This process sets up the source path mapping for the entire workspace, which is recommended. You can also create a similar mapping for individual projects by clicking **Edit Source Lookup Path** that appears below the **Source not found** error message and following steps 4 through 8.

The Restart command on the Run menu does not work

Workaround: Stop the program, then debug it again. If the debugger is hung in an endless loop, use the following bash alias to break the target, then stop it:

```
alias break="kill -2 `ps -a | grep nios2-elf-gdb | cut -f6 -d' '\`"
```

Programs that interact with a terminal Console on Windows don't work in the Nios II IDE v6.1 and later, but used to work in v6.0 or earlier

The Eclipse platform in v6.1 and later of the IDE (on Windows only) sends the string '\r\n' instead of just '\n' when running or debugging using the Terminal. This behavior can break existing software designs, and it is inconsistent with nios2-terminal, which still just sends '\n'.

Workaround: Change the software running on the Nios II processor to parse for \r as well as \n.

"Step failed. Target is not responding (timed out)" error message

The Nios II IDE debugger might hang and report this message if your code contains large arrays declared as local variables on the stack.

Workaround: Place the array and any other large buffers on the heap rather than on the stack.

Incorrect breakpoint filtering on threads

If you enable breakpoint filtering for a thread and later turn off filtering for the thread, the debugger might incorrectly continue to filter the thread.

Uninitialized Memory Error when executing from ISS

Under some cases the ISS does not ignore uninitialized memory reads, even when Uninitialized memory reads is set to Ignore on the ISS Settings tab of the run configuration. There is no known workaround.

Nios2-gdb-server fails to terminate after setting a watch point

You might be unable to terminate nios2-gdb-server after setting a watchpoint in the Nios II IDE debugger and resuming execution past the end of main. You will see an error "Terminate failed". You will not be able to start the debugger again; you will see a message reading "Another application is using the target processor..." in the Console view.

Workaround: Terminate the nios2-gdb-server.exe process manually using the Windows Task Manager.

Watchpoints do not work when set on variables whose size are not 32-bits

Workaround: Change the type of global and static local variables to int, long, or unsigned long before setting watchpoints on them.

Debugger cannot step into __sflags, and continues execution instead

The Nios II IDE debugger is unable to step into some low-level C library functions, such as `__sflags()` which is called from `_fopen_r()`. (`_fopen_r()` is called from `fopen()`.) If you try to step into such a function, execution will proceed as if you had indicated the debugger should resume execution.

Workaround: Step over such functions. Or, if execution continues after trying to step in, click **Suspend** on the Run menu.

Missing traced load/store instruction and data in the Trace view

If the trace options **Include load addresses**, **Include store addresses** or **Include data values** are enabled during debug, the load and store address and data will not appear at the first breakpoint after starting debugging. They will appear at successive breakpoints.

Workaround: To see load or store addresses and data in the instruction trace prior to main, turn on **Break at alt_main()** located on the **Debugger** tab for your debug configuration.

Cannot use watchpoints in the Nios II IDE when the FS2 console is open

Watchpoints do not work in the Nios II IDE when the **Use FS2 console window for trace and watchpoint support** setting is turned on in the **Debugger** tab of the Debug configuration. You will see an error message "The execution of program is suspended because of error." with details indicating that hardware watchpoints could not be inserted and deleted.

Workaround: If the FS2 console is open, you must use it to control watchpoints. For details, see the FS2 documentation.

Breakpoints on adjacent lines of assembly fail to halt the processor

Setting breakpoints on adjacent lines of assembly code might cause the Nios II processor to stop responding to the debugger.

Workaround: When debugging in mixed mode or disassembly view, separate breakpoints by at least one assembly instruction. This issue does not affect Nios II cores that do not have hardware breakpoints enabled in the JTAG debug module.

Navigating Projects

Internal Error when double-clicking on objdump file

On a Windows PC when opening a large objdump file in the Nios II IDE, you might get the following error message: "Unable to create this part due to an internal error. Reason for the failure: Editor could not be initialized."

Workaround: Adjust the Windows launch arguments for the Nios II IDE editor. Perform the following steps:

10. On the Windows Start menu, browse to the **Nios II EDS 7.0** program icon, right click it, then click **Properties**. A Windows Properties dialog box appears.
11. In the **Target** field, add `vmargs -Xmx1024m` to the end of the path to the Nios II IDE executable. For example:

```
C:\altera\70\nios2eds\bin\eclipse\nios2-ide.exe -vmargs -Xmx1024m
```

Nios II IDE doesn't recognize when an elf file is built outside the IDE

If you use a build process external to the Nios II IDE, the IDE might not recognize when you have generated or re-generated the elf file for your project. In this case, the IDE debugger might report an error that the elf file doesn't exist.

Workaround: Right click on the project name and click **Refresh** to allow the IDE to recognize the elf file.

After switching workspaces, the IDE appears to freeze while displaying the Nios II IDE splash screen

After clicking **Switch Workspace** on the File menu on a Windows machine, a Nios II IDE splash screen appears. Unfortunately, this splash screen obscures a dialog box which asks you to specify the new workspace.

Workaround: Press Alt-Tab to switch applications. You will see two relevant application icons: an Eclipse icon associated with the splash screen and a Nios II IDE icon associated with the workspace dialog box. Select the Nios II icon to bring the dialog box to the foreground.

C/C++ Scanner does not understand certain C/C++ constructs

The C/C++ scanner is used for C/C++ Search, navigation, open declaration and parts of content assist. Due to limitations of the C/C++ Scanner, these features will not work with the following code constructs:

- Kernighan & Ritchie-style C
- Functions that take a function-pointer as an argument

Workaround: If the C/C++ Search fails, use the File Search facility.

C2H Compiler

This section lists any issues related to the Nios II C-to-Hardware Acceleration (C2H) Compiler.

C2H Compiler out-of-memory error

The C2H Compiler might generate an out-of-memory error at compile time for code that initializes a large local array.

Workaround: Initialize the memory with the Nios II processor and access it through pointers in the accelerator.

C2H Compiler doesn't accelerate subfunctions located in a separate file

When accelerating a function in a file, the C2H Compiler cannot link subfunctions that are defined in a different file.

Workaround: Include all subfunctions called by the accelerated function within the same source code file.

Incorrect results from logical or conditional operation with side-effects

The C2H Compiler always evaluates both operands of logical (&&, ||) and conditional (?:) operators. This is different from expected ANSI C behavior, for which operands are evaluated left-to-right, and unnecessary operands are skipped. For example, in the expression `(i-- && j--)`, if the value of `i` is zero, the right-hand-side (RHS) expression should not evaluate (i.e., `j` should not be decremented). However, this C2H Compiler erroneously evaluates both sides unconditionally, causing `j` to be decremented. The following example expressions could suffer from the same issue: `(i-- || j--)`, `(cond ? i-- : j--)`

Workaround: Use logical and conditional operations whose operators have no side effects. Side effects include pre-/post-fix increment operations (`++`, `--`), memory operations (`*`, `[]`, `.`, `->`), and function calls.

Closed system library while working with the C2H Compiler

The C2H Compiler requires the system library to obtain important details about the system, and cannot function if the system library is closed.

Workaround: Ensure that the system library project in the Nios II IDE is open prior to building an application project that contains a hardware accelerator.

Launch SOPC Builder button in C2H view

When the Nios II IDE workspace contains multiple projects with multiple system libraries, the incorrect SOPC Builder system might open when you click the Launch SOPC Builder in the C2H view.

Workaround: Keep only one system library project open at a time while using the C2H Compiler.

Build clean causes build failure

Performing a clean build on a Nios II IDE project that contains a hardware accelerator can cause the next build to fail in the IDE, because the clean build erroneously deletes a file required by the C2H Compiler.

Workaround: Do not perform a clean build on projects that use hardware accelerators. If you have already performed a clean build, recompile with option **Build software, generate SOPC Builder system, and run Quartus II compilation** to regenerate the necessary files.

Multiple clock domains causes hardware accelerator to fail

If a hardware accelerator and the components connected to its master ports are in different clock domains, the accelerator might behave incorrectly.

Workaround: Assign a single clock to a hardware accelerator and all the slave ports it connects to. It is acceptable for the system to contain multiple clock domains.

Comma operators are not supported

The C2H Compiler does not support comma operators, such as the following example:

```
for(i = 0, j = 3; i < 10; i++, j++)
{
```

```

    /* statements */
}

```

Workaround: You can manually duplicate the same functionality, such as:

```

j = 3;
for(i = 0; i < 10; i++)
{
    /* statements */
    j++;
}

```

Array elements of structs do not copy correctly

C2H hardware accelerators do not correctly copy array elements of structs. For example:

```

typedef struct my_struct {
    int a;
    int b;
    int buf[BUF_SIZE];
}MY_STRUCT;

MY_STRUCT struct_a = {1, 2, {3, 3, 3, 3}};
MY_STRUCT struct_b = {9, 8, {7, 7, 7, 7}};

struct_a = struct_b;

```

In this example, the `a` and `b` elements of the struct will copy correctly, but the `buf` element will not. After this assignment, `struct_a` will equal `{9, 8, {3, 3, 3, 3}}`.

Workaround: Copy the array elements explicitly, as follows:

```

{
    int i=0;
    do
    {
        struct_a.buf[i] = struct_b.buf[i];
        i++;
    } while (i<LENGTH_OF_BUF_ELEMENT)
}

```

Simulating on ISS is not supported

The Nios II instruction set simulator (ISS) does not support custom SOPC Builder components, and therefore cannot simulate systems that use hardware accelerators. You might get the following internal error if attempting to simulate using the ISS:

```

Internal Error (unhandled exception) in file cosim_main.cpp

```

Workaround: Run the program on a hardware system that includes the hardware accelerator.

Hardware accelerators remain after deleting the software project

If a system contains C2H hardware accelerators, deleting the software project that defines the accelerators does not remove the accelerators from the hardware system, and the accelerator logic remains in the SOPC Builder system.

Workaround: To remove an accelerator from a system, delete the accelerator from the C2H view in the Nios II IDE first, and then recompile the software project. The C2H Compiler then removes the accelerator from the

SOPC Builder system. Once the compilation is complete then the software application can be deleted from the workspace.

Changing build configurations produces unexpected results

The C2H Compiler does not support multiple build configurations (e.g. Release or Debug) in the Nios II IDE. After creating one or more accelerators in a particular configuration, the C2H Compiler will produce undefined results if you switch to a different build configurations and create more accelerators.

Workaround: For a specific SOPC Builder system and Nios II IDE project, specify C2H accelerators in only one build configuration. Note that you can still use multiple build configurations, as long as only one configuration specifies C2H Compiler settings.

Flash Programmer

This section lists any issues relating to the Nios II IDE flash programmer.

'No such file or directory' error

You might get this error when programming flash for a project stored in a path containing spaces. The flash programmer does not correctly handle spaces in the directory path. However, this error is benign, because flash programming will complete successfully.

Workaround: None required.

elf2flash elf size limit

The elf2flash utility supports .elf files up to approximately 24 MBytes in size. The elf2flash utility might fail with error "java.lang.OutOfMemoryError" on files larger than 24 MBytes.

Workaround: You can either lower the number of symbols in your elf file by turning off debug symbols, or specify less initialized data in the application.

Verifying flash with the nios2-flash-programmer command fails

Using the **nios2-flash-programmer** command line utility to verify flash contents using the `--verify` argument might result in a verify failure even though flash contents are correct. The failure message will be similar to the following:

```
Verifying 00000000 ( 0%)Failed to verify at around 00000000 Verify failed
```

Workaround: To work around this issue, avoid using the `--verify` argument with the **nios2-flash-programmer** command-line utility. A verification of flash contents can be done by reading the flash contents into a file using the `--read` argument, then comparing the file to the input file used to program flash.

Using the Nios II Flash Programmer with an override file stored in <Nios II EDS path>/bin might generate an error:

```
4 [main] nios2-flash-programmer 4440 _cygtls::handle_exceptions: Error while
dumping state (probably corrupted stack) Segmentation fault (core dumped)
```

Workaround: To workaround this issue, move the override file from the <Nios II EDS path>/bin directory to the working directory of your Nios II IDE project. Then, specify the override file to **nios2-flash-programmer** using the `--override=` parameter. In Nios II IDE, you can do this in the **Additional nios2-flash-programmer arguments** field in the Flash Programmer dialog box. In command-line mode, the override file can be specified by adding the `--override=` argument to the command line, as follows:

```
nios2-flash-programmer --base-0x0 --override=my_fp_override.txt standard.flash
```

When used with a multiprocessor Nios II system, the Nios II Flash Programmer might produce an error:

```
There are two or more Nios II processors available which match the values
specified. Please use the configuration dialog to pick one, or specify the --
device and/or --instance parameters on the command line.
```

Workaround: In the Nios II IDE on the Flash Programmer dialog box, specify an appropriate `--instance=<correct instance value>` argument in **Additional nios2-flash-programmer arguments**. If using command line mode, add `--instance=<correct instance value>` to the command line. The correct instance value can be found in the project's system library project in the file **generated.sh**.

Download Cables & Debug Hardware

This section lists any issues related to download cables and other debug hardware.

Communication errors during run/debug sessions using older download cables

Debugging with the following Altera download cables might fail, due to electrical noise-related JTAG communication failures: USB-Blaster™ Rev A, ByteBlaster™, ByteBlasterMV™, ByteBlaster II, and MasterBlaster™ cables.

Currently, the only fully supported cable for downloading, debugging, or communicating with Nios II systems is the USB-Blaster Rev B cable or later. Revision B cables are clearly labeled as Revision B. (Revision A cables have no revision label.)

Workaround: Use a USB-Blaster Rev B cable. Older cables can be used, but they might encounter JTAG communication failures.

Development Boards

This section lists any issues related to Altera development boards.

Intermittent failures while accessing CompactFlash card

The Nios II Development Kit version 5.0 and higher includes a CompactFlash controller peripheral suitable for interfacing to CompactFlash cards in True IDE mode on Nios development boards. In order for True IDE mode to operate, CompactFlash cards require that the ATASEL_N input be driven to ground during power-up.

The CompactFlash controller peripheral includes a configurable power register used to power-cycle CompactFlash cards in Nios II software through a MOSFET on the Nios development boards. However, in certain development boards, power to the CompactFlash card will not turn off completely during this power cycle operation. Because of this, the CompactFlash might not sample the ATASEL_N pin during the power-cycle operation after FPGA configuration when this pin is driven to ground. Instead, the CompactFlash card might sample the ATASEL_N pin when power is first applied to the development board, when I/O are not yet driven by the FPGA (before FPGA configuration).

Workaround: If you encounter errors with CompactFlash when using the Nios development boards, try one of the following:

- Use a different CompactFlash card. Certain cards are more susceptible to the power-cycling issue than others.
- Modify the Nios development board. This is recommended for users who are familiar and comfortable with board-level modifications. Disconnect pin 9 (ATASEL_N) on the CompactFlash socket on your Nios development board and tie this pin to ground. Note that the CompactFlash socket uses a staggered numbering on the pins (starting from pin 1: 1, 26, 2, 27, ...); refer to the CompactFlash Association specification for right-angle surface-mount connectors for exact specifications on this connector. This modification will permanently enable True-IDE mode operation.

Toolchain (gcc, gdb, etc.)

This section lists any issues related to the Nios II compiler toolchain, such as gcc and gdb.

Breakpoints in C++ constructors fail to halt the processor

Breakpoints set in a C++ constructor might not halt the processor due to a widespread GNU GCC, GDB issue. This is not a Nios II IDE-specific issue.

Workaround: You can work around this issue by moving all of your constructor source code into another class method, called `init`. Then invoke this method from within the constructor.

Target Software

This section lists any issues related to software or drivers that target the Nios II processor.

Lightweight IP (lwIP) failure when using DHCP

The Lightweight IP stack might experience intermittent failures when using DHCP to acquire an IP address. When the failure occurs, the following is printed to stdout:

```
Assertion "dhcp_create_request: dhcp->p_out == NULL" failed at line 1283 in
/cygdrive/c/altera/61b169/nios2eds/components/altera_lwip/UCOSII/src/downloads/lwi
p-1.1.0/src/core/dhcp.c
```

Workaround: If possible, Altera recommends switching to the NicheStack TCP/IP Stack - Nios II Edition introduced in Nios II 6.1. If not, the following workarounds can be employed.

- Power-cycle the target board and re-attempt DHCP negotiation, which usually results in correct acquisition of an address from DHCP, assuming that the DHCP server is able to allocate IP addresses.
- Use a static IP address and disable DHCP.

stdin, stdout, and stderr don't work in MicroC/OS-II applications when using small C library option

MicroC/OS-II applications using `stdin`, `stdout`, and `stderr` fail to operate when using the small C library.

Workaround: Disable the small C library option.

malloc(), realloc() failures with MicroC/OS-II

When using the MicroC/OS-II RTOS, calls to `malloc()` and `realloc()` might fail if successive calls to `malloc()` or `realloc()` within a MicroC/OS-II task occur after changing the task priority of the task in which a memory block was originally allocated.

Workarounds:

- Allocate and/or reallocate memory blocks outside of MicroC/OS-II tasks, before task switching starts. Changing thread priorities at runtime is now possible.
- Allocate fixed areas of memory using arrays (rather than using `malloc()`) before task switching starts. Changing thread priorities at runtime is now possible.
- Allocate memory using `malloc()` or `realloc()` from a MicroC/OS-II task. You can change task priorities at runtime, but only for tasks that have not used `malloc()` or `realloc()`.

cout from MicroC-OS/II task will not send data to STDOUT

If neither `printf()` or `cout` is used from `main()` before tasks are started, `cout` will not work from a task.

Workaround: Add the following C++ code to the beginning of `main()`:

```
std::ios_base::sync_with_stdio(false);
```

Problems using HAL drivers with Toshiba Flash

The HAL CFI Flash driver might not work for Toshiba flash memory that claims to be CFI compliant.

Workaround: In the `altera_avalon_cfi_flash_table.c` file, change the `#define READ_ARRAY_MODE` from `(alt_u8) 0xFF` to `(alt_u8) 0xF0` and rebuild the project.

Creating new custom HAL components

When you first create a component's inc directory or HAL header file, you might first need to perform a clean build (i.e., rebuild) of existing system library projects for the new files to be detected.

Legacy SDK

Support for the Legacy SDK mode was removed in version 6.0 of the Nios II Embedded Design Suite.

SOPC Builder and Quartus II Software

This section lists any issues related to the Quartus II software or SOPC Builder that specifically affect Nios II designers.



For further information on the Quartus II software, refer to the latest Quartus II release notes on the Altera web site:

<http://www.altera.com/literature/lit-qts.jsp>

Example Designs

This section lists any issues related to the example designs included with the Nios II Embedded Design Suite.

Hardware Designs

Incorrect SSRAM PLL phase shift in Stratix II 2s60 RoHS and Cyclone™ II 2C35 standard and full_featured designs

The PLL phase shift used to implement the clock to SSRAM in the standard and full_featured designs targeting the Nios II Cyclone II 2C35 and Nios II Stratix II 2S60 RoHS development boards is incorrect, and may cause timing violations when reading from or writing to SSRAM if you increase the system clock speed above that of the example designs (85MHz), while using two clock-cycles of read latency in the SSRAM interface.

While most Altera example designs targeting these boards run at 85Mhz, the "standard" design targeting the Stratix II 2S60 RoHS board runs at 100MHz. Although no SSRAM failures have been observed with this board/example design in its current configuration, the design should be re-generated for reliable SSRAM operation.

Note that no action is required if you wish to continue using the example designs at their initial clock speed of 85MHz. However, if you wish to create your own custom design that uses SSRAM and runs above 85MHz, or if you wish to increase clock speed of an Altera example design above 85MHz, Altera recommends that the PLL phase shift for generating the SSRAM clock on Nios II Cyclone II 2C35 and Nios II Stratix II 2S60 RoHS edition boards be set to -3.38ns, in "normal" PLL mode. Altera's example designs targeting these boards currently have a phase shift of -4.8ns, with the PLL in "normal" mode.

To change the PLL setting, perform the following steps:

12. Open the desired example design in SOPC Builder
13. Double click the "pll" instance existing in the design and launch the PLL MegaWizard
14. In the PLL MegaWizard, click the "Output Clocks" tab and then clock "c2", used to generate the SSRAM clock.

15. Change the PLL phase shift from -4.8ns to -3.38ns
16. Click Finish to save changes and exit the PLL MegaWizard
17. Click Finish to save settings to the PLL instance in SOPC Builder
18. Re-generate the system in SOPC Builder and re-compile in Quartus II.

For further reference, please refer to the SSRAM Interface readme document, located in the <Quartus II installation directory>/sopc_builder/documents folder. This document discusses the SSRAM timing analysis methodology in detail. Additionally, *AN 411: Understanding PLL Timing for Stratix II Devices* discusses clock phase shift calculations and assignments.

Software Designs

RAM test failure when running Memory Test software template on the ISS

An issue in the instruction set simulator (ISS) model of the JTAG UART can cause a console communication error during the RAM test when running the Memory Test software template on the ISS.

Networking Examples

If you are running a networking example design and you are asked for a 9-digit number after the letters 'ASJ', and your Nios II development board does not have a sticker with a 9-digit number after the letters 'ASJ', please enter a unique 9-digit number when prompted. Ensure that this number is unique to each Nios board connected to your network to avoid network address conflicts.

Hardware Simulation

This section lists issues related to simulating Nios II processor systems on an RTL simulator, such as the ModelSim[®] simulator.

Error: "UNC paths are not supported. Defaulting to Windows directory."

If you launch ModelSim from a working directory that is mapped via a UNC path (a path that starts with // instead of drive letter), you will get this error message in SOPC Builder. This error occurs because ModelSim is calling a cmd shell, which does not support UNC paths.

Workaround: Map the UNC path to a drive letter and use the drive letter to reference the working directory in the launching shell.

Simulation failure if reset address is set to EPCS

Running ModelSim RTL simulation of a Nios II system fails if the reset address of the Nios II processor is set to an EPCS Serial Flash Controller.

Workaround: To simulate your system, temporarily set the Reset Address of the Nios II CPU to the memory that your application code will reside (for example, SDRAM), then re-generate the system in SOPC Builder and run RTL simulation again. Before booting the Nios II CPU from EPCS flash on your target board, change the Nios II

Reset Address back to the EPCS Controller peripheral and re-generate the system in SOPC Builder and re-compile in the Quartus II software to produce an updated FPGA configuration file with the Nios II CPU booting from EPCS flash.

Uninitialized BSS variables in simulation

If your program reads the value of an uninitialized BSS variable during HDL simulation when the HAL system library has been compiled with the **ModelSim only, no hardware support** property enabled in Nios II IDE, a warning will be produced about unfiltered data being 'x'. This occurs because when this property is enabled, the code that clears the BSS memory region is omitted to speed up HDL simulation so this memory region is uninitialized. The BSS region contains global and static local variables that are not initialized by the application so they default to a value of zero. When the Nios II CPU reads uninitialized variables, it displays a warning and converts any of the bits of the uninitialized data to zero which correctly mimics the effect of the missing BSS clearing code. The HAL code that executes before and after main() may use BSS variables so these warnings might be generated even if your application doesn't use the BSS.

ModelSim fails to load large memory models

The ModelSim tool might fail to load simulation models for memory arrays larger than 128M bytes, halfwords or words in size. If the sum of the following parameters is greater than 27, the ModelSim tool will fail to load:

- address_bits (i.e. 14)
- column bits (i.e. 11)
- $\log_2(\text{number of banks})$ (number of banks is usually 4, so this term is usually 2)
- $\log_2(\text{chipselects})$ (number of chipselects is usually 1, so this term is usually 0)

Workaround: Simulate using a smaller SDRAM than the SDRAM implemented in hardware. This is possible if the entire memory space doesn't need to be simulated.

Documentation Issues

This section lists errors, unclear descriptions, or omissions from current published specifications or product documents.

GNU assembler does not accept the --defsym flag

According the GNU documentation, you can set an assembler definition by using the `--defsym` flag, but it does not work in the following form: `--defsym MY_VAR=1`

Nios II Processor Reference Handbook refers erroneously to the data bus

In the *Implementing the Nios II Processor in SOPC Builder* chapter of the *Nios II Processor Reference Handbook*, under *Caches & Tightly Coupled Memories Tab*, the following text appears under *Instruction Settings*: "Usually you enable bursts on the processor's data bus when processor data is stored in DRAM, and disable bursts when processor data is stored in SRAM." This sentence should read "Usually you enable bursts on the processor's instruction master when instructions are stored in DRAM, and disable bursts when instructions are stored in SRAM."

Contact Information

For more information, contact Altera's mySupport website at www.altera.com/mysupport. Click **Create New Service Request**, and choose the **Product Related Request** form.

Revision History

Table 1 shows the revision history for the Nios II Embedded Design Suite v7.0 Errata Sheet.

Table 1. Nios II Embedded Design Suite v7.0 Errata Sheet Revision History

Version	Date	Errata Summary
1.0	March 2007	First release
1.1	March 2007	<ul style="list-style-type: none">■ Interrupt vector custom instruction in multi-processor systems■ Incorrect stack checking when interrupt vector custom instruction is used■ Nios II/e core burst support■ Duplicate errata combined:<ul style="list-style-type: none">■ MicroC/OS-II source code not found while debugging■ "Source not found" error occurs when debugging, after the Nios II IDE stops at a breakpoint.



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
literature@altera.com

© 2007 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



Revision History
