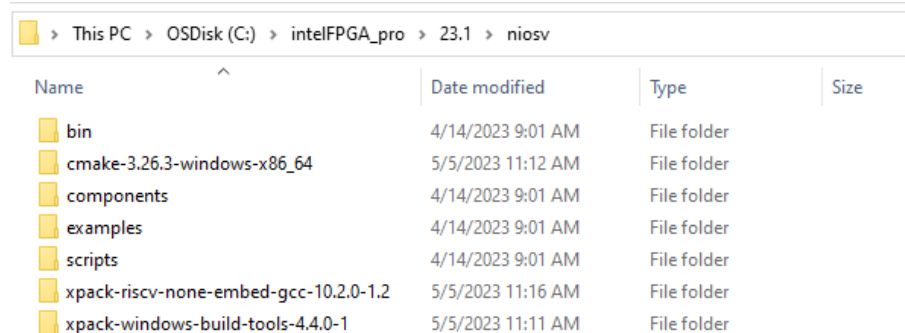# Nios V Read Write Example Instructions

1. Quartus Prime Pro version 21.3 or newer is required for Nios V

2. Install required software
   a. GNU RISC-V Embedded GCC
      i. https://github.com/xpack-dev-tools/riscv-none-embed-gcc-xpack/releases/
      ii. Download file: xpack-riscv-none-embed-gcc-10.2.0-1.2-win32-x64.zip
      iii. Extract file in Quartus install directory
         1. C:\intelFPGA_pro\21.4\niosv
   b. CMake packages for binary distributes
      i. https://cmake.org/download/
      ii. Download file: cmake-3.26.3-windows-x86_64.zip
      iii. Extract file in Quartus install directory
         1. C:\intelFPGA_pro\21.4\niosv
   c. xPack Windows Build Tools
      i. https://github.com/xpack-dev-tools/windows-build-tools-xpack/releases/
      ii. Download file: xpack-windows-build-tools-4.4.0-1-win32-x64.zip
      iii. Extract file in Quartus install directory
         1. C:\intelFPGA_pro\21.4\niosv
   d. Upon completion of steps a-c your niosv/ directory should look like the following.

3. Edit Windows PATH environment variable.
   a. Open the Start Search, type in "env", and choose "Edit the system environment variables":
   b. Click the "Environment Variables..." button near the bottom.
   c. In the "User variables for <user>" frame highlight "Path" and then press "Edit..."
   d. Use the "New" or "Edit" button to ensure the following paths have been added.

4. Verify correctness of PATH environment variable
    a. Double click the open_cmd_prompt.bat file to open a Windows Command Prompt in the present working directory.
    b. From the Windows Command Prompt run path_env_check.bat. The output should look something like the following:

    c.


```
C:\work\Intel_GIT\Software_Read_Write_Example\Software_Read_Write_Example\NiosV_Read_Write_Example_C>path_env_check.bat

This script will check to ensure your PATH environment variable
is set correctly.

Testing xpack-riscv-none-embed-gcc path...
C:\intelFPGA_pro\23.1\niosv\xpack-riscv-none-embed-gcc-10.2.0-1.2\bin\riscv-none-embed-ar.exe

Testing xpack-windows-build-tools path...
C:\intelFPGA_pro\23.1\niosv\xpack-windows-build-tools-4.4.0-1\bin\make.exe

Testing cmake path...
C:\intelFPGA_pro\23.1\niosv\cmake-3.26.3-windows-x86_64\bin\cmake.exe

Testing sopc_builder path...
C:\intelFPGA_pro\23.1\quartus\sopc_builder\bin\qsys-edit.exe

Testing niosv path...
C:\intelFPGA_pro\23.1\niosv\bin\niosv-app.exe

Testing quartus path...
C:\intelFPGA_pro\23.1\quartus\bin64\quartus_sh.exe

C:\work\Intel_GIT\Software_Read_Write_Example\Software_Read_Write_Example\NiosV_Read_Write_Example_C>
```

If you see a "Could not find files…" message you will need to correct your PATH environment variable prior to proceeding.

```
Testing quartus path...
INFO: Could not find files for the given pattern(s).
```

5. Unzip NiosV_Read_Write_Example Design.zip file
    a. Below is a description of the files included in the NiosV_Read_Write_Example_C directory.

Filename	Description
Makefile	Makefile to build and manage FPGA and software application projects.
NiosV_Read_Write_Example_Guilde.pdf	This document.
open_cmd_prompt.bat	Opens a Windows Command Prompt in the present working directory.
path_env_check.bat	Window batch file to help check correctness of PATH environment variable.
program_fpga.tcl	Tcl script to assist in the programming of the FPGA.
read_write_example.c	Software application file written in C.
toggle_issp.tcl	Tcl script to toggle the internal reset register using In System Sources and Probes (ISSP).

top_21_4_0_67.qar	Quartus project archive file. Including Platform Designer system and top-level System Verilog file.

6. Restore and View Quartus project
    a. From the command line type "make restore"
    b. Open the Quartus project and the Platform Designer system (sys.qsys). The system consists of the following:
        i. Clock and reset inputs
        ii. Nios V CPU
        iii. data_instruction_ram - This RAM is used to hold the application code which the Nios V CPU will execute. During FPGA compilation the ./data_instruction_ram.hex file is located into this RAM.
        iv. jtag_uart - Used to capture STDIO from our application code.
        v. pio_8bit - 8-bit Parallel I/O. Only bit 0 is used and connected to an LED on a development board. The application code blinks this LED.
        vi. ram_32bit - 32-bit slave RAM connected to the Nios V CPU. The application code demonstrates how to read and write to this RAM.

Use	Co...	Name	Description
✔		⊞ ⊡ clock_in	Clock Bridge Intel FPGA IP
✔		⊞ ⊡ reset_in	Reset Bridge Intel FPGA IP
✔		⊞ ⊡ cpu	Nios V/m Processor Intel FPGA IP
✔		⊞ ⊡ data_instruction_ram	On-Chip Memory (RAM or ROM) Intel FPGA...
✔		⊞ ⊡ jtag_uart	JTAG UART Intel FPGA IP
✔		⊞ ⊡ pio_8bit	PIO (Parallel I/O) Intel FPGA IP
✔		⊞ ⊡ ram_32bit	On-Chip Memory (RAM or ROM) Intel FPGA...

7. Compile Project
    a. The project is currently targeted towards an Arria 10 development board. You will need to change the target to match your board and update the location of the "clk" pin to match your board. Optionally you can pin out the "led" pin to match your board in order to view the blinking LED.
    b. To compile the FPGA, from the command line run "make fpga". This will:
        i. Generate the Board Support Package
        ii. Generate the software application (create the data_instruction_ram.hex file)
        iii. Compile the FPGA

8. Program the FPGA
    a. Connect your computer to your board and run "make program" from the command line. This script will program the FPGA, toggle the built-in reset register (using In System Sources and Probes) and start the UART terminal. You should see the following output.

```
Capture stdout...
juart-terminal
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "USB-BlasterII [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

Read 0x01234567 at address 0x00A00000
Read 0x89ABCDEF at address 0x00A00004

Ctrl-C to exit.
```

9. Update Application Code Without Recompiling the FPGA
   a. Open the read_write_example.c file and change the last printf statement to include the word "Press":
      i. printf("\nPress Ctrl-C to exit.\n");
   b. Run "make app". This script will regenerate the application code, creating an updated .elf file.
   c. run "make update". The .elf file will be downloaded to the FPGA. The built-in reset register will be toggled, and the application code will execute. You should see the following output.

```
Capture stdout...
juart-terminal
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "USB-BlasterII [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

Read 0x01234567 at address 0x00A00000
Read 0x89ABCDEF at address 0x00A00004

Press Ctrl-C to exit.
```

10. Application Code
    a. The application file read_write_example.c includes io.h. The io.h library provides the IOWR and IORD functions used to read and write the slave/agent RAMs attached to the Nios V.
    b. The included ./software/cpu_hal_bsb/system.h file is generated as part of the board support package. This file defines parameters associated with the components in the Platform Designer system including the base address location of the slave/agent RAM and PIO block which are used in the read_write_example.c file.