

**PLATFORM DESIGNER CUSTOM COMPONENT WITH PULSE WIDTH MODULATION (PWM)**

© Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty but reserves the right to make changes to any products and services at any time without notice. Intel 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 Intel. Intel 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. Other names and brands may be claimed as the property of others.

Contents

[1 Introduction: Test Your Knowledge: Knight Rider ROM 3](#_Toc59100928)

[2 Lab Steps **Error! Bookmark not defined.**](#_Toc59100929)

[3 Programming on to the DE10-Lite board 8](#_Toc59100930)

# Introduction: Test Your Knowledge: Platform Designer Customer Component with PWM

This test your knowledge workshops assumes you have completed the [Embedded Nios course](https://www.intel.com/content/www/us/en/programmable/support/training/course/ouwnios.html).

The resulting embedded system from this lab will be used as the starting point for your lab.

# Making Platform Designer Components

The basis of this section is the following tutorial:

<https://ftp.intel.com/Public/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/Making_Qsys_Components.pdf>

You will need to complete a few steps prior to starting the tutorial.

Since copying and pasting text from files to the Quartus editor can introduce hidden characters download these 3 files from the github site:

[reg32\_avalon\_interface.v](https://github.com/intel/FPGA-Devcloud/blob/master/main/QuickStartGuides/EmbeddedNios_DE_board_Quickstart/CustomComponents/reg32_avalon_interface.v)

[PWM\_generator.v](https://github.com/intel/FPGA-Devcloud/blob/master/main/QuickStartGuides/EmbeddedNios_DE_board_Quickstart/CustomComponents/PWM_generator.v)

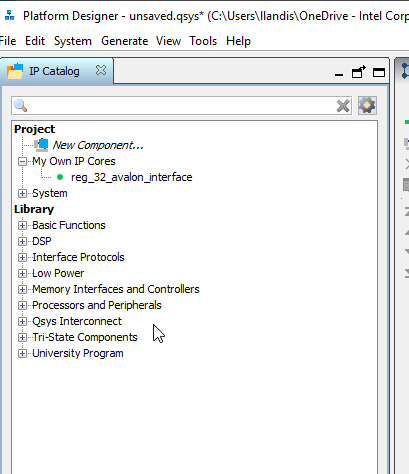
[DE\_hello\_world\_PWM.c](https://github.com/intel/FPGA-Devcloud/blob/master/main/QuickStartGuides/EmbeddedNios_DE_board_Quickstart/CustomComponents/DE_hello_world_PWM.c)

Restore your completed project Embedded Nios project in Quartus. This project needs to compile hardware properly and demonstrate that you can download the .sof file to the board, export the .sopcinfo file to Eclipse, write C code, compile it and download the .elf file to the DE10-Lite board. If these steps are not complete, revisit the Embedded Nios course.

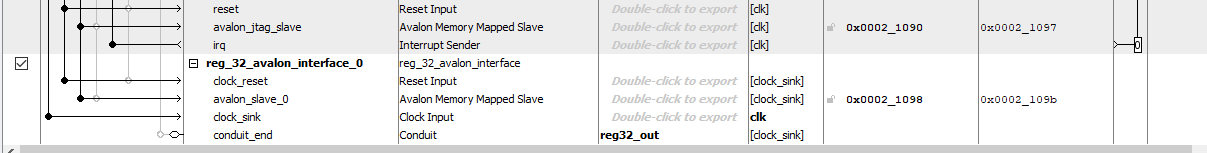
Open the Making Qsys Components pdf file. Follow the steps through page 25.

# Integrating the reg32\_avalon\_interface module into your existing Platform Designer project

In Platform Designer, add your new component as new slave device.



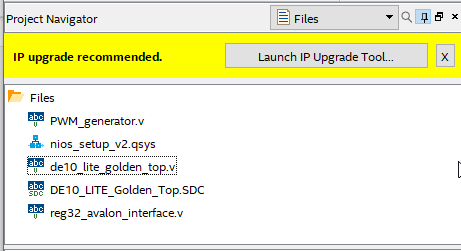
Double click on your new component. Add the component to your design and make connections to clock\_reset, clock\_sink, and Avalon Memory Mapped Slave. No connection is required to the conduit end. Name components as shown in the figure below.



Use the System 🡪 Assign Base addresses, generate your nios\_setup\_v2 system.

# Connections from nios\_setup\_v2 to the top level wrapper

We provided you the PWM\_generator.v file. Place that file in the same directory as de10\_lite\_golden\_top.v. If you aren’t familiar with PWM (pulse width modulation), this technique is commonly used for control of LED lighting and motors. Click on this [link](https://www.kompulsa.com/introduction-pwm-pulse-width-modulation-works/#:~:text=PWM%20works%20by%20pulsating%20DC,device%20such%20as%20an%20LED.&text=PWM%20signals%20are%20typically%20square,with%20a%2050%25%20duty%20cycle.) for more details. Add the PWM\_generator.v file to your project: Project 🡪 Add/Remove Files in Project. Follow the steps as you have done in past projects to add the PWM\_generator.v file. If reg32\_avalon\_interface.v isn’t part of your project, add that as well. The list of files should look like this:



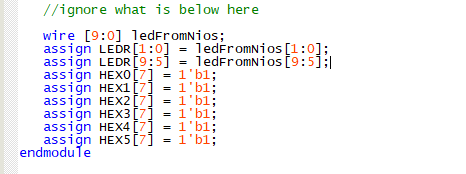
The .SDC file is optional.

Edit de10\_lite\_golden\_top.v file. Because you added a new port called reg32\_out you need to change the instantiation of that line of code. Note lines 74, 88 and 92 are changed lines from your original version. The line numbers might be slightly different.



Note on line 92, that we have used LEDR[4] as the LED that will get the PWM controlled signal that will control the brightness. Because LEDR[4] is already assigned, you will need to change a bit of code to avoid a conflict.

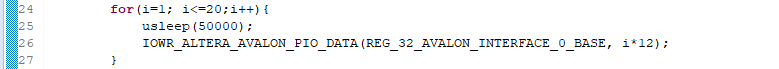
See line 103 below, LEDR[4] is removed from the list.



Compile your design and look for errors and fix as necessary.

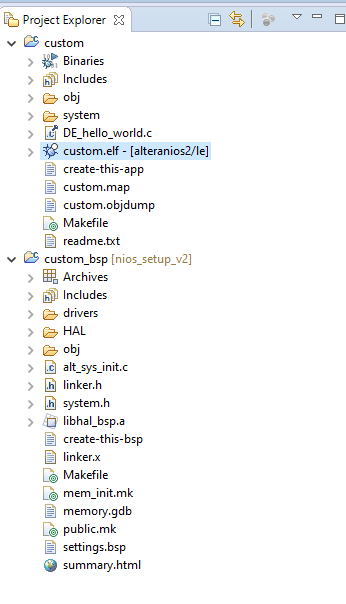
# Eclipse Project

Launch the Eclipse project that you used for your Nios Embedded project, we will reuse it. Delete the user .c code from the project, and add the code you downloaded called DE\_hello\_world\_PWM.c . Open the code in Eclipse and observe how it works. The main piece of code that controls the PWM function is shown below:



The code will cycle through 20 increments of brightness from 12 to 240 (max is 255). Note the use of the usleep command which indicates a delay of 50,000 microseconds. There is a long list of Hardware Adaption Layer (HAL) commands you can use here: [Nios® II Software Developer Handbook](https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2sw_nii5v2gen2.pdf).

Note that the REG\_32\_AVALON\_INTERFACE\_0\_BASE code is referenced in system.h. Find the file in the list and open it (its under the BSP).

The system.h has the memory map showing variables assigned to the addresses in the memory map. Make sure the line 26 maps appropriate variable for the reg32. If compile fails, refer to the system.h file. Build your project fix errors and complete.

# Programming on to the DE10-Lite board

Bring up your Quartus Project and launch the programmer. Select your SOF file and program. If you have problems programming, make sure that Eclipse isn’t running a program or the JTAG programmer will conflict.

# Downloading the .elf file

Download the .elf file to the board and check out the functionality of the PWM running LED[4].

# Follow on challenges

1. Tune the C code and make the brightness increment up to full brightness and then reverse the PWM to decrease the brightness back to the minimum value.
2. Merge the PWM\_generator.v file as a submodule to reg32\_avalon\_interface.v. recompile and try again. Note the change of bit widths – you now will have a 1 bit PWM as the exported interface.
3. Add additional reg32+PWM modules and write a Knight Rider type sequencer using varying levels of brightness per LED.

Document Revision History

List the revision history for the application note.

|  |  |  |
| --- | --- | --- |
| Name | Date | Changes |
| Larry Landis | 12/18/2020 | Initial Release of lab manual |