



Mixed Mode PCIe and 8B10B Protocol Demo for CertusPro-NX

User Guide

FPGA-UG-02192-1.0

August 2023

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk the responsibility entirely of the Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Contents.....	3
Acronyms in This Document.....	7
1. Introduction.....	8
1.1. Learning Objectives.....	8
2. Hardware and Software Requirements.....	9
2.1. Hardware Requirements.....	9
2.2. Software Requirements.....	9
3. Demo Design Overview.....	10
3.1. Theory of Operation.....	10
3.2. Design Overview.....	11
3.2.1. User Interface Application.....	11
3.2.2. Device Drivers.....	11
3.2.3. Device Hardware (FPGA Design).....	12
4. Importing and Building the FPGA Demonstration.....	13
4.1. Hardware Directory Structure.....	13
4.2. Building Lattice Radiant Project.....	13
5. Setting Up the Demo.....	14
5.1. Hardware Setup.....	14
5.1.1. Jumper Configuration.....	14
5.1.2. Programming the FPGA.....	14
5.1.3. Status LED.....	17
5.2. Software Setup.....	17
5.2.1. Software Setup and Installation for Windows.....	17
5.2.2. Software Setup for Linux.....	31
6. Application Overview.....	34
6.1. Running the Mixed Mode Demo Application.....	34
6.2. Using the Mixed Mode Demo Application User Interface.....	34
6.2.1. Functionality Test Tab.....	35
6.2.2. GPIO.....	35
6.2.3. I ² C.....	39
7. Troubleshooting.....	46
7.1. SPI Flash Update.....	46
7.2. Driver Installation and User Interface Launch for Windows.....	46
7.2.1. Problem in Driver Installation.....	46
7.2.2. Problem with Launching User Interface.....	47
7.3. Driver Installation User Interface Launch for Linux.....	47
7.3.1. Problem in Driver Installation.....	47
7.3.2. Problem in Driver Loading.....	48
7.3.3. Problem with User Interface Launching.....	48
References.....	50
Technical Support Assistance.....	51
Revision History.....	52

Figures

Figure 3.1. Relationship between Hardware and Software Components	10
Figure 3.2. Mixed Mode Demo SW Design for PCIe.....	11
Figure 3.3. Mixed Mode Design	12
Figure 5.1. CertusPro-NX Versa Evaluation Board Connection.....	14
Figure 5.2. Creating a New Project from a Scan	15
Figure 5.3. Lattice Radiant Programmer Window	15
Figure 5.4. CertusPro-NX FPGA Device Settings	15
Figure 5.5. Device Properties Window for CertusPro-NX SPI Flash Programming (with Macronix Flash).....	16
Figure 5.6. Programmer Menu Bar	16
Figure 5.7. Programmer Output Window	16
Figure 5.8. CertusPro-NX Programming Done LED	17
Figure 5.9. Running Disable Integrity Checks Command	17
Figure 5.10. Running Test Sign on Command	18
Figure 5.11. Troubleshoot Option.....	18
Figure 5.12. Advanced Options.....	19
Figure 5.13. Select Startup Settings	19
Figure 5.14. Restarting Windows.....	19
Figure 5.15. Windows Installer: Welcome Page	20
Figure 5.16. Windows Installer: Destination Folder Page.....	21
Figure 5.17. Windows Installer: Summary Page	21
Figure 5.18. Windows Installer: Application Installed	22
Figure 5.19. Device Configuration Prompt	22
Figure 5.20. Device Driver Installation Wizard	23
Figure 5.21. Windows Security in Driver Installation.....	23
Figure 5.22. Device Driver Installation Completed	24
Figure 5.23. Device Manager	24
Figure 5.24. Showing Device Properties	25
Figure 5.25. Hardware IDs of CertusPro-NX I ² C Device.....	25
Figure 5.26. Hardware IDs of CertusPro-NX GPIO Device.....	26
Figure 5.27. Update Driver Menu in Device Manager	26
Figure 5.28. Update Driver Options.....	27
Figure 5.29. Browse the Driver for Device	27
Figure 5.30. Windows Security in Device Manager	28
Figure 5.31. I ² C Driver Installation Status Message for CertusPro-NX.....	28
Figure 5.32. I ² C and GPIO Device Drivers for CertusPro-NX in Device Manager	29
Figure 5.33. Select “Uninstall device” that User Want to Uninstall.....	30
Figure 5.34. Select the Check Box and Click on Uninstall Button	30
Figure 5.35. Uninstalled Device	31
Figure 6.1. PCIe Test Application Device Info Tab	34
Figure 6.2. Functionality Test Tab.....	35
Figure 6.3. GPIO Input Switch	35
Figure 6.4. GPIO Output LEDs	36
Figure 6.5. Demo LEDs	36
Figure 6.6. Open Reveal Analyzer	36
Figure 6.7. Configure Reveal Analyzer	37
Figure 6.8. Reveal Analyzer Capture Before Press “Run Counter”	38
Figure 6.9. Reveal Analyzer Capture After Press “Run Counter”	38
Figure 6.10. LED_7 Switch ON To Indicate Pattern Detected with No Bit Errors.....	38
Figure 6.11. I ² C Bit-Rate Selection	39
Figure 6.12. I ² C Master Write	39
Figure 6.13. Single Write to an Arbitrary Address of IMX258-0AQH5 Camera.....	40
Figure 6.14. Sequential Write Starting from an Arbitrary Address of IMX258-0AQH5 Camera	40

Figure 6.15. I ² C Master Write Procedure.....	41
Figure 6.16. I ² C Master Read	41
Figure 6.17. Single Read from the Held Address of IMX258-0AQH5 Camera.....	42
Figure 6.18. Sequential Read Starting from the Held Address of IMX258-0AQH5 Camera.....	42
Figure 6.19. I ² C Master Write to Write the Register Address.....	42
Figure 6.20. I ² C Master Read	43
Figure 6.21. I ² C Master Register Read	43
Figure 6.22. Single Read from an Arbitrary Address of MX258-0AQH5 Camera	43
Figure 6.23. Sequential Read Starting from an Arbitrary Address of MX258-0AQH5 Camera	43
Figure 6.24. I ² C Master Register Read	44
Figure 6.25. I ² C Master Register Read	44
Figure 6.26. I ² C Batch Mode Read/Write.....	45
Figure 7.1. TCK Frequency Setting	46
Figure 7.2. Port Selection.....	46
Figure 7.3. User Interface with No Device Driver	47
Figure 7.4. lspci -vnm for CertusPro-NX Output Image	48
Figure 7.5. Content List of Demonstration/Linux Directory	48
Figure 7.6. Content List of Software/Linux Directory	48

Tables

Table 5.1. Device IDs.....	25
Table 6.1. Master Write Control Description.....	39
Table 6.2. Registers of MX258-0AQH5 CMOS Camera	40
Table 6.3. Master Register Read Control Description.....	43
Table 6.4. Transaction Log Control Description	44
Table 6.5. Batch Mode Control Description.....	45

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
BAR	Base Address Register
FDSOI	Fully Depleted Silicon on Insulator
FPGA	Field-Programmable Gate Array
LED	Light-emitting diode
MIPI	Mobile Industry Processor Interface
PCIe	PCI Express
PHY	Physical Layer
SPI	Serial Peripheral Interface
USB	Universal Serial Bus
I ² C	Inter-integrated Circuit
MDIO	Management Data Input Output
GPIO	General Purpose Input Output

1. Introduction

This guide describes how to set up and run the PCIe Multifunction Demo using devices built on the CertusPro™-NX FPGA device.

The demo is targeted for CertusPro-NX Versa Evaluation Board, which features the CertusPro-NX FPGA in the LFG672 package. The above-mentioned FPGAs are built on the Nexus FPGA platform using low power 28 nm FD-SOI technology.

This guide familiarizes the user with the process of setting up and running a design which incorporates both PCI Express and 8B10B Protocol. It is assumed that the user does not have any associated tools installed on the user's system.

The demo discussed in this document is the Mixed Mode PCI Express and 8B10B protocol Demo.

1.1. Learning Objectives

After completing the steps in this guide, user can perform the following:

- Set up and install all applicable development tools and PCI Express demos.
- Establish communication between the FPGA and the system through the PCI Express link.
- Run the Mixed Mode demo which implements two separate PCI Express functions on a single endpoint device and one 8B10B protocol interface. Each PCIe function allows the user to control a different aspect of the FPGA Board. While the 8B10B protocol incorporates a random pattern generator and checker.
- Use what the demo teaches the user about designing Lattice PCI Express and 8B10B protocol solutions.
- Become familiar with the software development tools and major design flow steps employed in this kit.
- Use other existing documentation in conjunction with this guide.

This document assumes that the user has already installed the Lattice Radiant™ design software. This document covers some of the basic functions of the Lattice Radiant software. If the user would like to learn more about the Lattice Radiant software, refer to the Lattice Radiant software Help system.

2. Hardware and Software Requirements

2.1. Hardware Requirements

To install the kit design and run the demo software, a computer with a PCI Express ×16, ×8, ×4, or ×1 slot is required. The computer must also have a USB port and be able to run the Lattice Radiant software. All other hardware and drivers are included in the kit.

- Mini-USB to USB-A cable for programming the bitstream
- 12 V Power Adapter
- Evaluation Board
 - CertusPro-NX Versa Evaluation Board for CertusPro-NX

Additionally, for CertusPro-NX, a MIPI CSI Camera Module is needed to test I²C functionality; this demo uses the Sony IMX258-0AQH5 Camera Module.

2.2. Software Requirements

The following software are required to obtain the expected results in the procedures described in this guide:

- Lattice Radiant software version 2.2 or later (available at the Lattice website [Design Software and IP](#) page).
- Mixed Mode PCIe and 8B10B Protocol demo for Windows 10 or Linux
- Windows 10 or Ubuntu 18.04.6
- Bit file for the SPI Flash
- LFCPNX_100_Mixed_Mode.bit file for CertusPro-NX

3. Demo Design Overview

3.1. Theory of Operation

The demo runs on a standard x64 PC and accesses the FPGA board installed in a PCIe slot. Figure 3.1 shows the relationship between the hardware and software components of the demo. The PCIe IP present in the Lattice FPGA occupying the Quad 0 Channel 0 acts as a PCIe endpoint occupying certain ranges of PCI memory space. The 8B10B protocol occupies Quad 0 Channel 2 and currently consists of a random pattern generator and checker. The Quad 0 Channel 2 used in the CertusPro-NX Versa Evaluation board is connected to the PCIe Goldfinger; hence a serial loopback from Tx to Rx is used in this demo to showcase the data transmission for 8B10B protocol.

When the PC boots, the BIOS and OS probe the PCI Express and PCI buses and detect the devices present on the buses and assign them in the PCI memory space ranges. The PCI memory space is mapped into the PC's memory space by the BIOS. Once the device driver is installed, the application software can read/write from/to the PCIe device memory (Embedded Block RAM - EBR). Currently the design allows the GPIO function within the PCIe to trigger the start of random pattern generator and checker within the 8B10B protocol. Hence, the application software can write to PCIe device memory which triggers the start of the random pattern generator and checker within the 8B10b protocol. The application software can also read from the PCIe configuration space registers. In the Mixed Mode demo, two separate PCIe functions are implemented within a single endpoint.

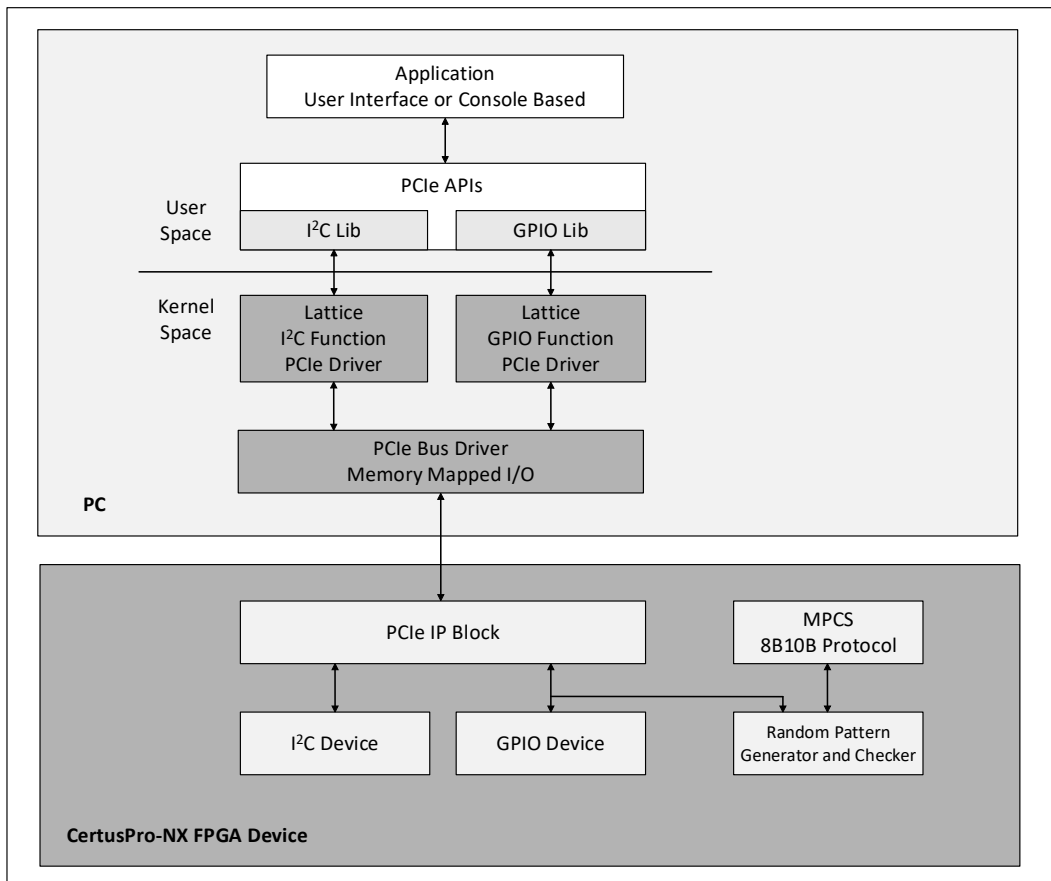


Figure 3.1. Relationship between Hardware and Software Components

3.2. Design Overview

A user interface application is provided for demonstrating the Mixed Mode demo. Application software is developed using a layered architecture consisting of the following layers:

- User Interface Application
- Driver API
- Device Drivers
- Device Hardware (FPGA Design)

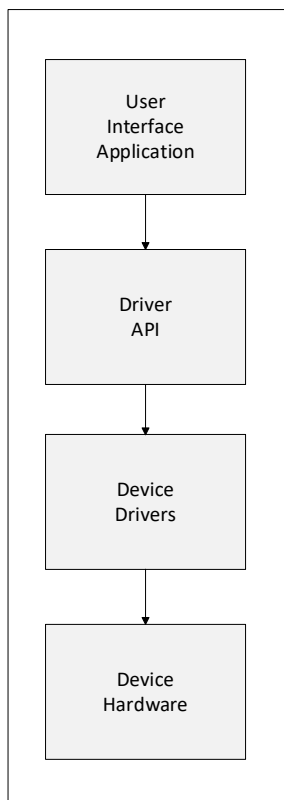


Figure 3.2. Mixed Mode Demo SW Design for PCIe

3.2.1. User Interface Application

The Mixed Mode demo includes a user interface written in Qt and uses the driver API to communicate with the device hardware to send control plane read/writes to registers in the IP. The Driver API is a C++ dynamic library which provides an interface to access the hardware.

On Windows, the *multifunction_lib.dll* library is a DLL that bridges the user space demo applications to the kernel space driver code and provides routines to control the IP modules.

On Linux, the library is named *libmem_rw.so*.

3.2.2. Device Drivers

The hardware drivers provide access to the FPGA board. On Windows, the *lpcie_gpio.sys* and *lpcie_i2c.sys*, and drivers support the Mixed Mode demo. On Linux, *gpio_main.ko* and *i2c_main.ko* support the demo.

3.2.3. Device Hardware (FPGA Design).

Figure 3.1 shows the top-level architecture of the FPGA design. PCIe hard IP is used on the FPGA side to implement the PCIe endpoint meanwhile MPCS is used to implement 8B10B protocol. The endpoint interfaces with the application logic for each function through an arbiter. The arbiter is also used to trigger the start of random pattern generator and checker used in the 8B10B protocol. LMMI interface is used for initial configuration of the PCIe IP and MPCS IP. A block diagram of the FPGA design is shown in Figure 3.3.

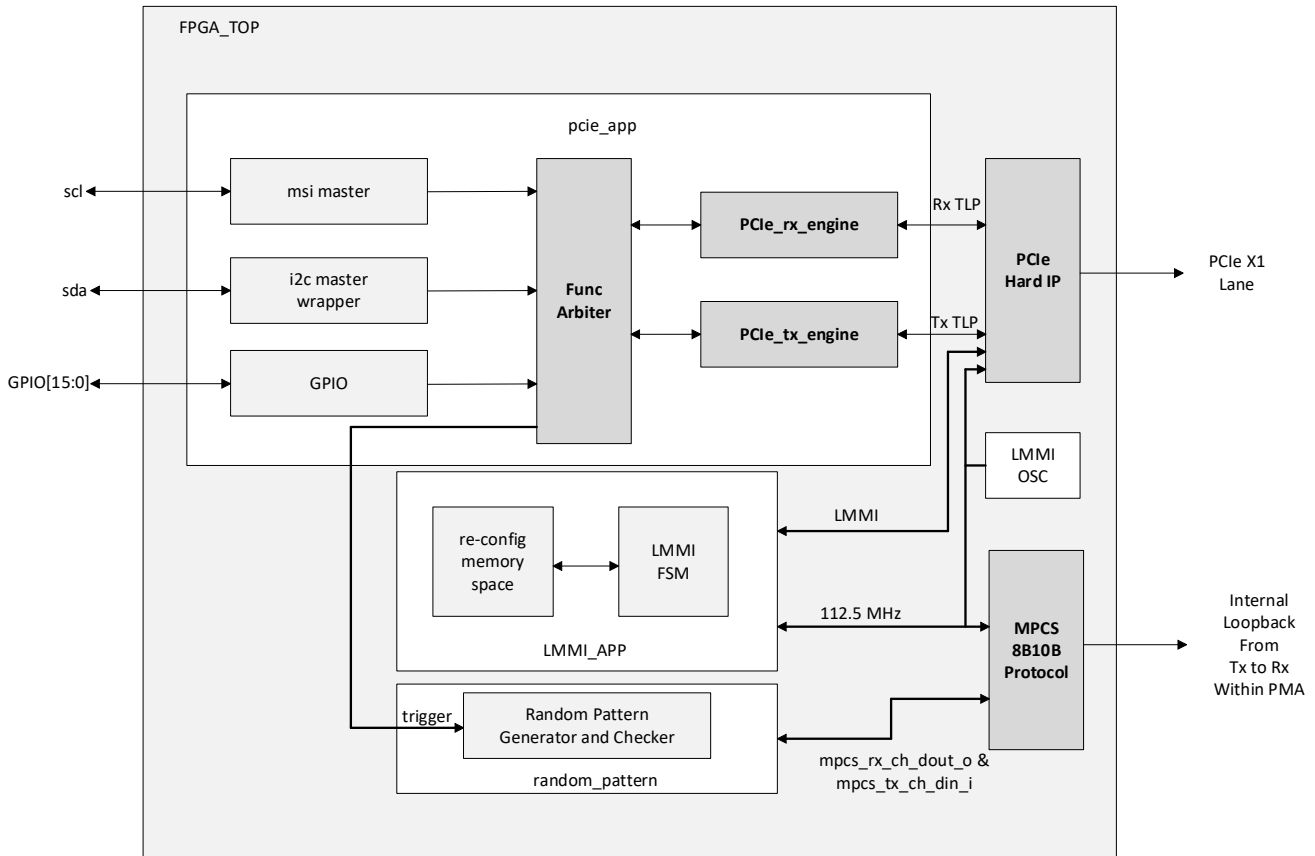


Figure 3.3. Mixed Mode Design

4. Importing and Building the FPGA Demonstration

The package includes the PCIe IP, .bit file, and synthesis projects using Lattice Radiant Software.

4.1. Hardware Directory Structure

The Hardware folder inside the package contains the following subfolders.

./CertusPro_NX_Mixed_Mode

- IP – Contains the pre-generated IPs used in the design. These IPs can be configured by clicking the .ipx file after opening the project in Lattice Radiant.
- Implementation – Contains the Lattice Radiant project (.rdf) file, constraints file (.pdc), and implemented design and bit files.
- Source – Contains RTL files required for the design.

4.2. Building Lattice Radiant Project

To generate the bit stream file:

1. Open the Lattice Radiant software.
2. Click **Open project** and browse to the .rdf file
 - a. The file is LFCPNX_100_Multi_Mode.rdf, which is located in the *Hardware\CertusPro_NX_Mixed_Mode\LFCPNX_100_Multi_Mode* folder.
3. Once the project loads, click **Task Detail View**. This shows a list of actions that Lattice Radiant will perform to build the .bit file.
4. Select the files and reports that the user wants to generate.
Note: The options needed to regenerate the .bit file are selected by default.
5. After selecting user's preferred reports, click **Run All**.
This creates a .bit file with the project's current name in the impl_1 folder.

5. Setting Up the Demo

5.1. Hardware Setup

This section covers the steps in programming the demo to the SPI memory of the FPGA Board.

5.1.1. Jumper Configuration

For CertusPro-NX, there are no jumpers to configure beyond the default. Refer to the [CertusPro-NX Versa Board User Guide \(FPGA-EB-02053\)](#) for a detailed list of jumpers on the board. When the board is plugged into the PCIe slot, external power is provided by the system, and SW6 should be in the up position to receive power from the PCIe slot. Connect the board to the PC running the Lattice Radiant software with the Mini USB Type A Cable as shown in [Figure 5.1](#).

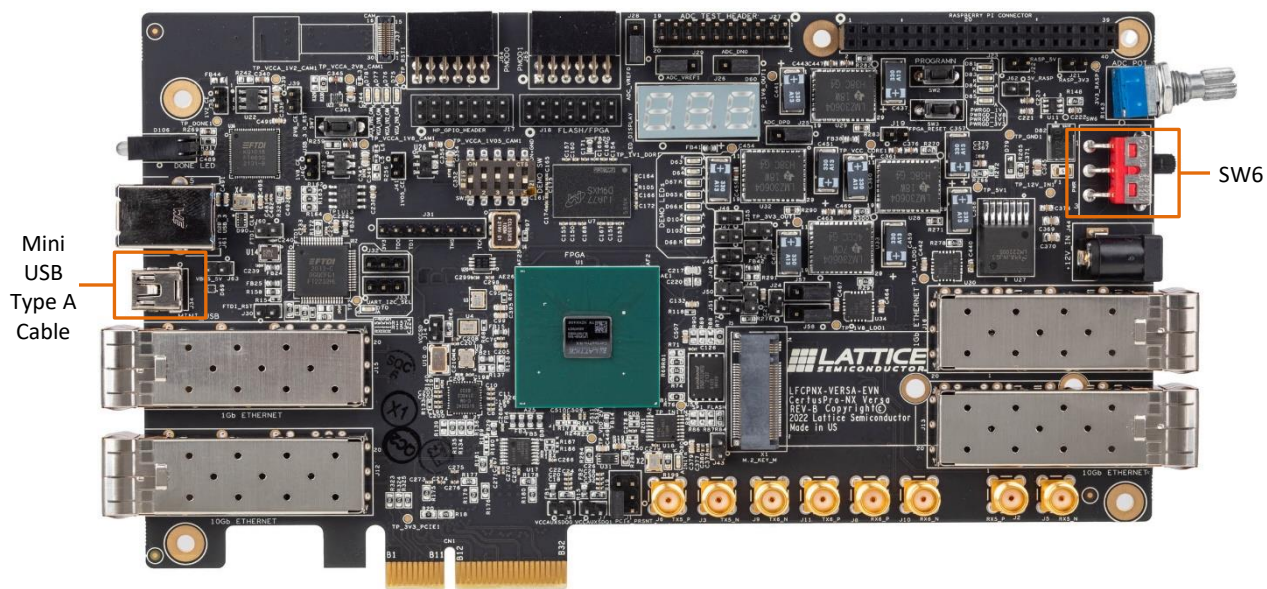


Figure 5.1. CertusPro-NX Versa Evaluation Board Connection

5.1.2. Programming the FPGA

To program the FPGA device:

1. Create a new project using the Lattice Radiant Programmer software. In the **Getting Started** dialog box, indicate **Project Name** and **Location** as shown in [Figure 5.2](#).
2. Select **Create a new project from scan**. Values are indicated in the **Cable**, **Port**, and **TCK Divider Setting (0-30x)** fields.
3. Click **OK**.

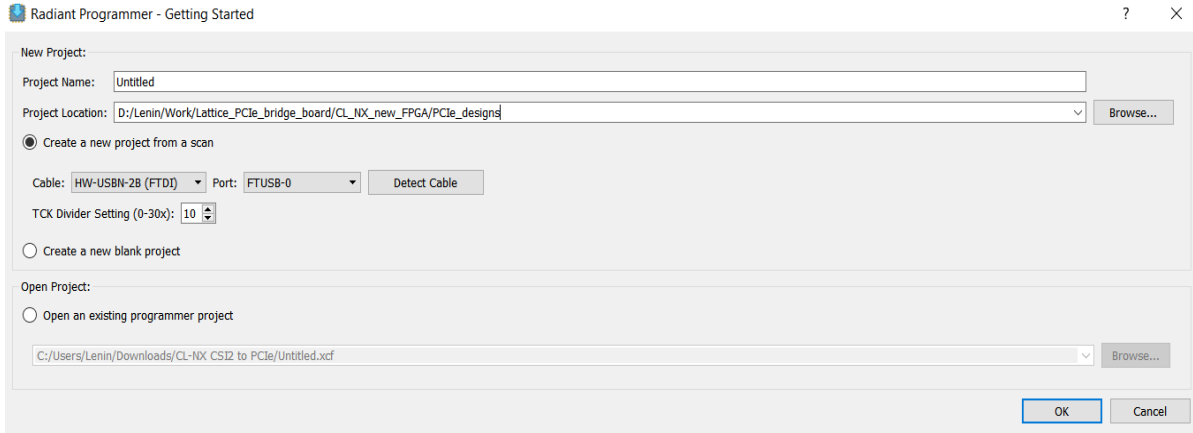


Figure 5.2. Creating a New Project from a Scan

4. The main interface opens as shown in Figure 5.3.

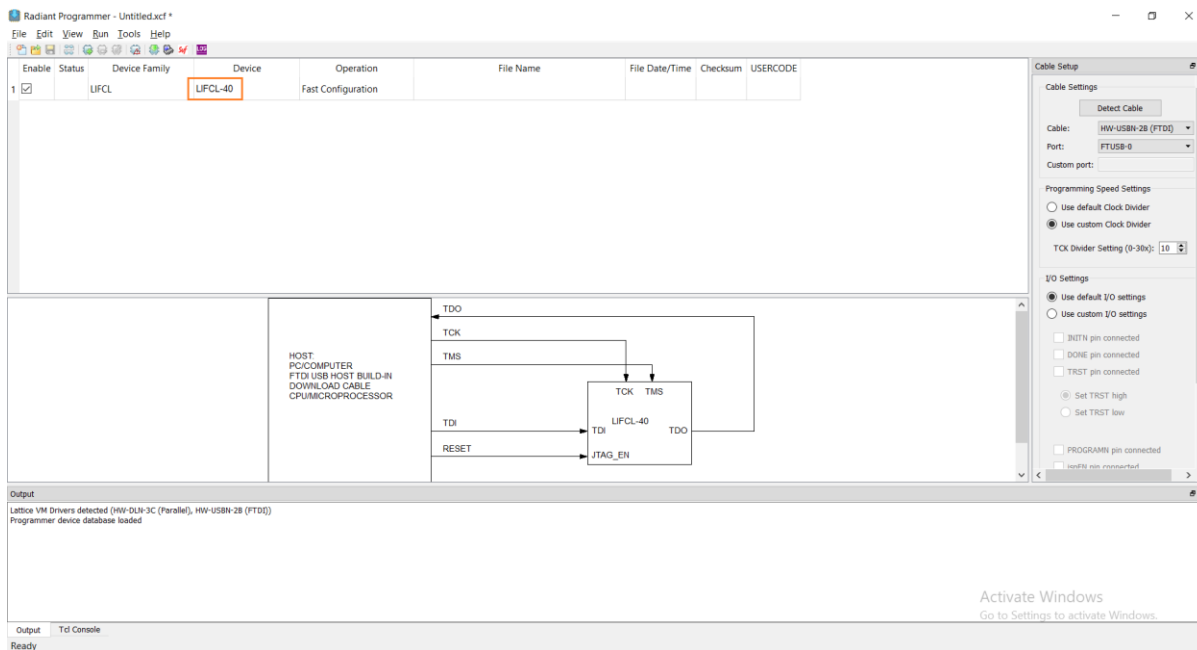


Figure 5.3. Lattice Radiant Programmer Window

5. If the Programmer settings do not match the settings shown in Figure 5.7 for CertusPro-NX, select these settings manually from drop down menu as shown in Figure 5.4.



Figure 5.4. CertusPro-NX FPGA Device Settings

To select the programming settings:

1. Browse and select the Programming file from the *Demonstration\Bitstream* folder.
 - a. Select *LFCPNX_100_Multi_Mode.bit*.
2. Click **OK**.
3. Double-click under **Operation** to open the **Device Properties** dialog box.

- Select the settings as shown in [Figure 5.5](#) for CertusPro-NX.

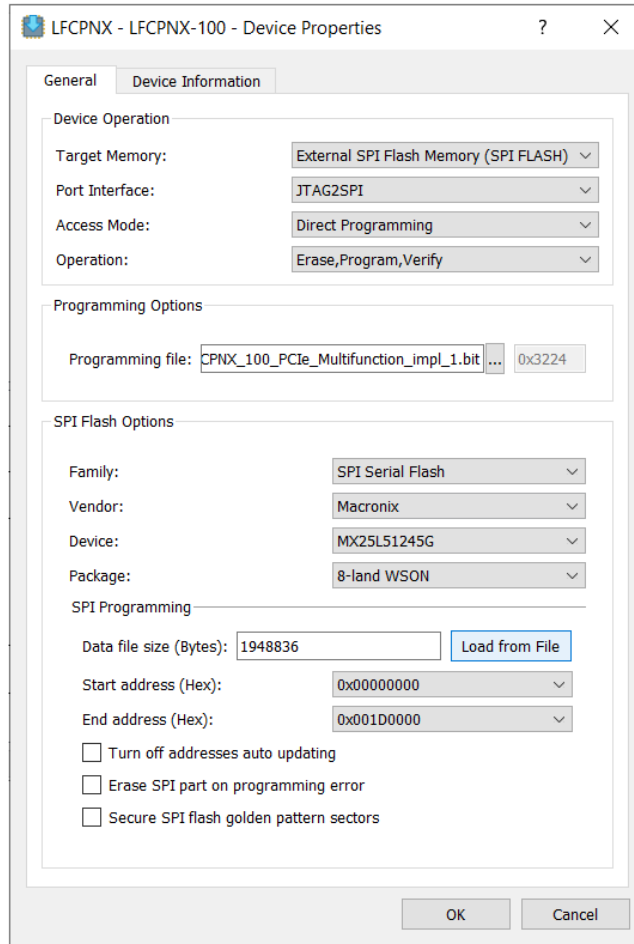


Figure 5.5. Device Properties Window for CertusPro-NX SPI Flash Programming (with Macronix Flash)

- Click the **Programming** button from the menu bar shown in [Figure 5.6](#) to start programming.

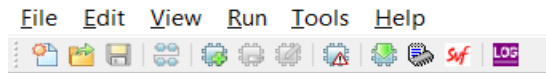


Figure 5.6. Programmer Menu Bar

- When the FPGA programming is successful, the output console displays an **Operation: successful** message as shown in [Figure 5.7](#).

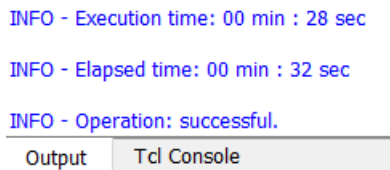


Figure 5.7. Programmer Output Window

- If the programming operation is unsuccessful, refer to the [Troubleshooting](#) section of this document.

- After programming, power cycle the board and check the status LEDs on the board. The LED status will be discussed in the next section.

5.1.3. Status LED

For CertusPro-NX, if the programming done LED lights up in green, then the configuration is successful. The LED is located as shown in [Figure 5.8](#).

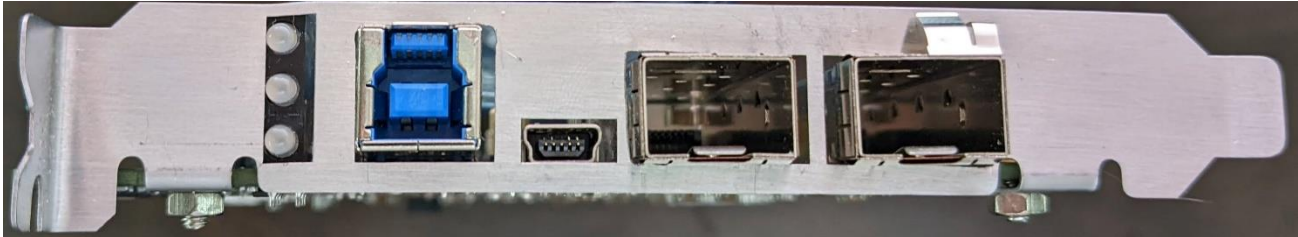


Figure 5.8. CertusPro-NX Programming Done LED

5.2. Software Setup

This section provides the procedure for installing software onto the host machine.

5.2.1. Software Setup and Installation for Windows

Before installing the driver, the Driver Signature Enforcement feature should be disabled.

5.2.1.1. Disabling Driver Signature Enforcement Permanently

To permanently disable Driver Signature Enforcement:

- Start the Command Prompt as administrator.
- Enter the following lines and press **Enter**.

```
bcdedit.exe -set loadoptions DISABLE_INTEGRITY_CHECKS
```

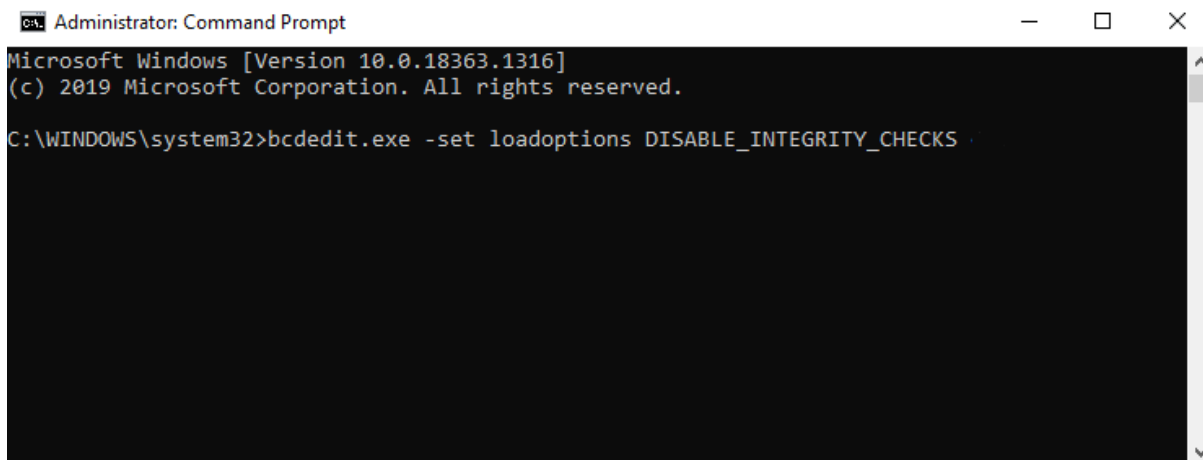


Figure 5.9. Running Disable Integrity Checks Command

```
bcdedit.exe -set TESTSIGNING ON
```

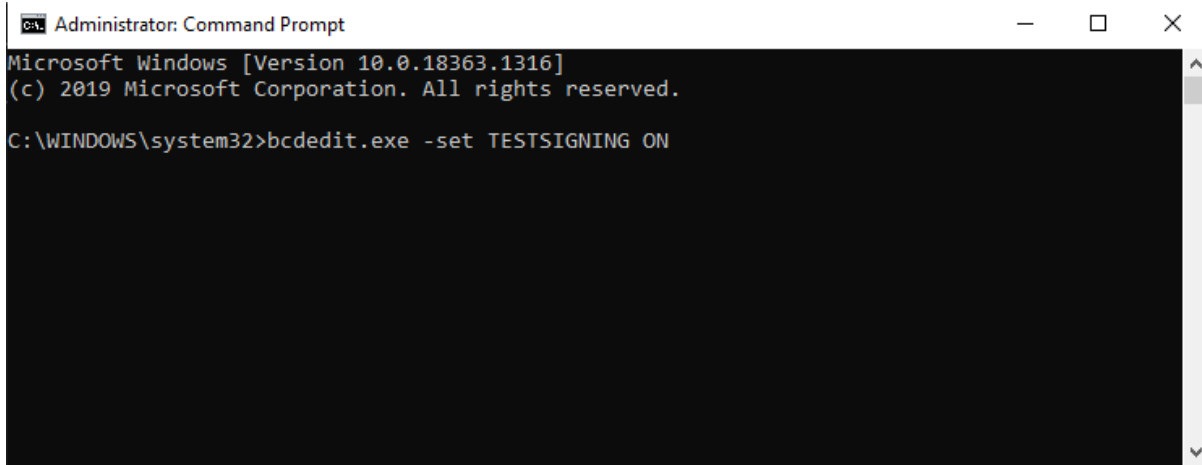


Figure 5.10. Running Test Sign on Command

3. Close the Command Prompt and restart the PC.

5.2.1.2. Disabling Driver Signature Enforcement Temporarily

Note: If Driver Signature Enforcement is already disabled, skip this section and proceed to the [Driver Installation](#) section.

To disable Driver Signature Enforcement temporarily on Windows 10:

1. Press the Windows key and click the **power** button.
2. Press and hold the Shift key and click **Restart**.
3. When the **Choose an option** screen appears as shown in [Figure 5.11](#), release the Shift key.
4. Click **Troubleshoot**.

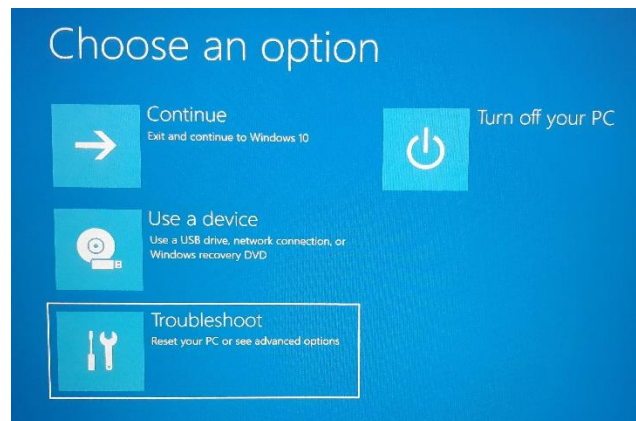


Figure 5.11. Troubleshoot Option

5. Select **Advanced options** and press **Enter**.



Figure 5.12. Advanced Options

6. Select **Startup Settings** and press **Enter**.

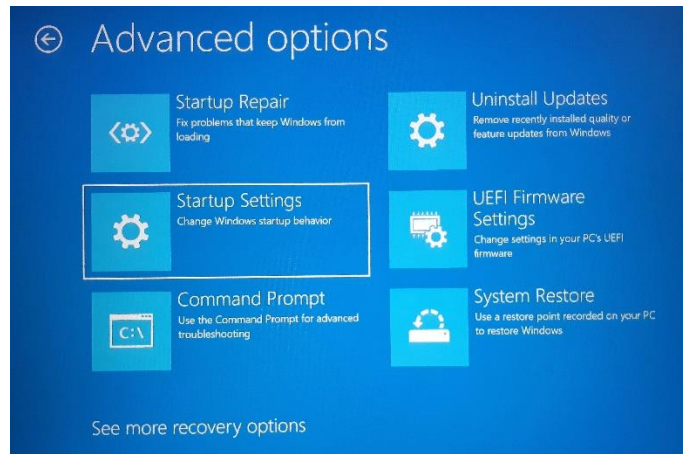


Figure 5.13. Select Startup Settings

7. Press **Enter** to restart.

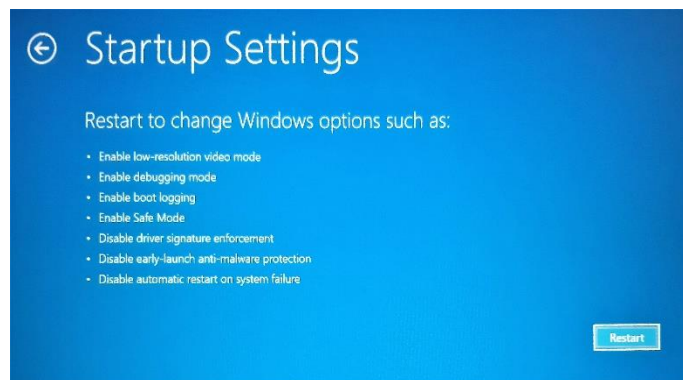


Figure 5.14. Restarting Windows

8. After restarting, select **Option 7** to disable driver signature verification.

5.2.1.3. Driver Installation

There are two ways to install the device driver:

- Install through a user interface installer (.exe) that is included in the demo package.
- Install through Device Manager manually.

Installing Multifunction Demo Device Driver through the User Interface Installer

The Multifunction Demo device driver can also be installed during the installation of the user interface as described in the following section.

The Installer provides a standard packaging format for applications and a standard method for customizing the applications. The installer helps to install the Multifunction Demo application in the user's system.

The Framework supported version is Windows 10 using WDF 1.25 or earlier.

To install the Multifunction Demo device driver through user interface:

1. In the *Demonstration\Windows10\Application* folder, double click *setup.exe*.
2. The welcome page appears. Click **Next**.

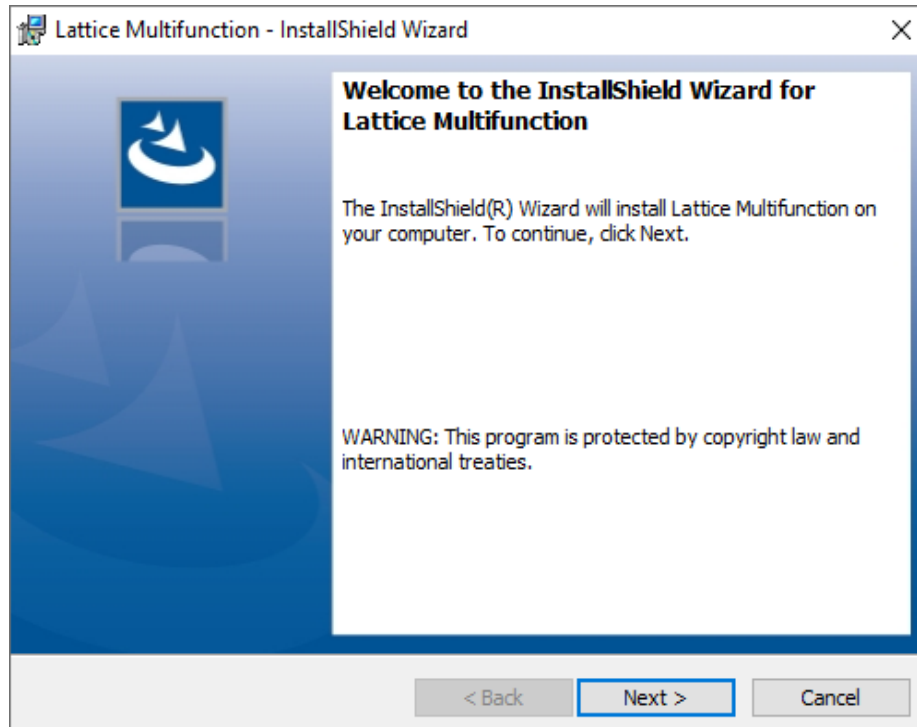


Figure 5.15. Windows Installer: Welcome Page

3. Provide the location where the user want to install the application. Click **Next**.

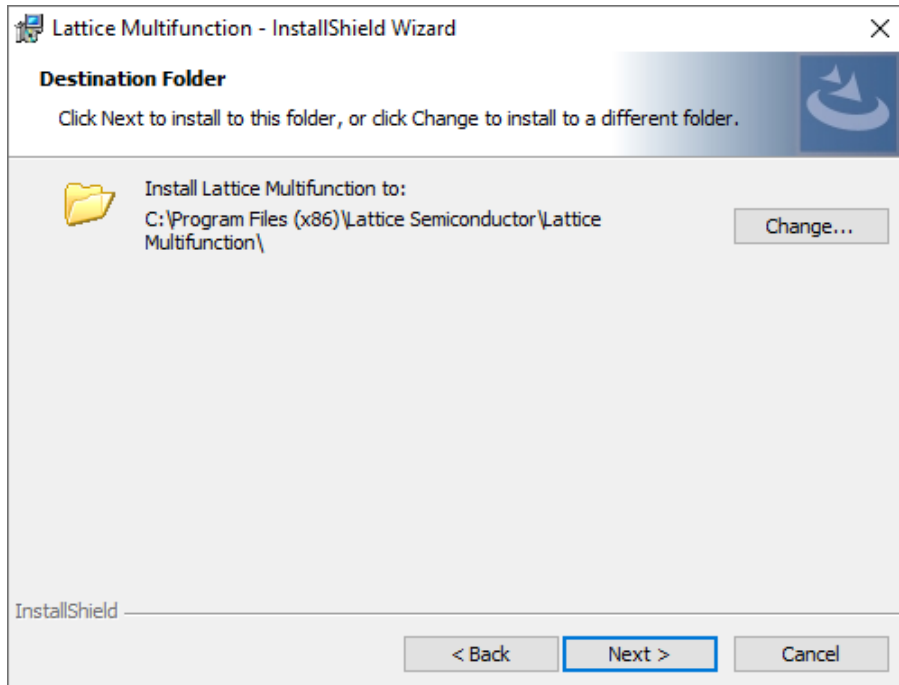


Figure 5.16. Windows Installer: Destination Folder Page

4. The installation summary page is shown in Figure 5.17. Click **Install**.

Note: Administrative access is required to run this command.

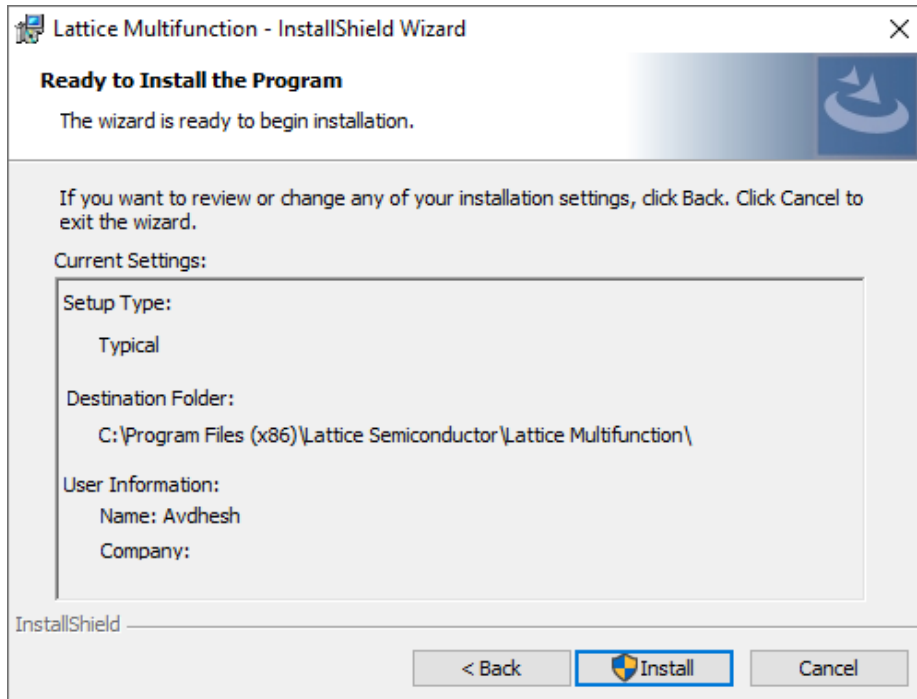


Figure 5.17. Windows Installer: Summary Page

5. The installation of the Multifunction Demo application starts. When the installation of the software is complete, the drivers are installed.

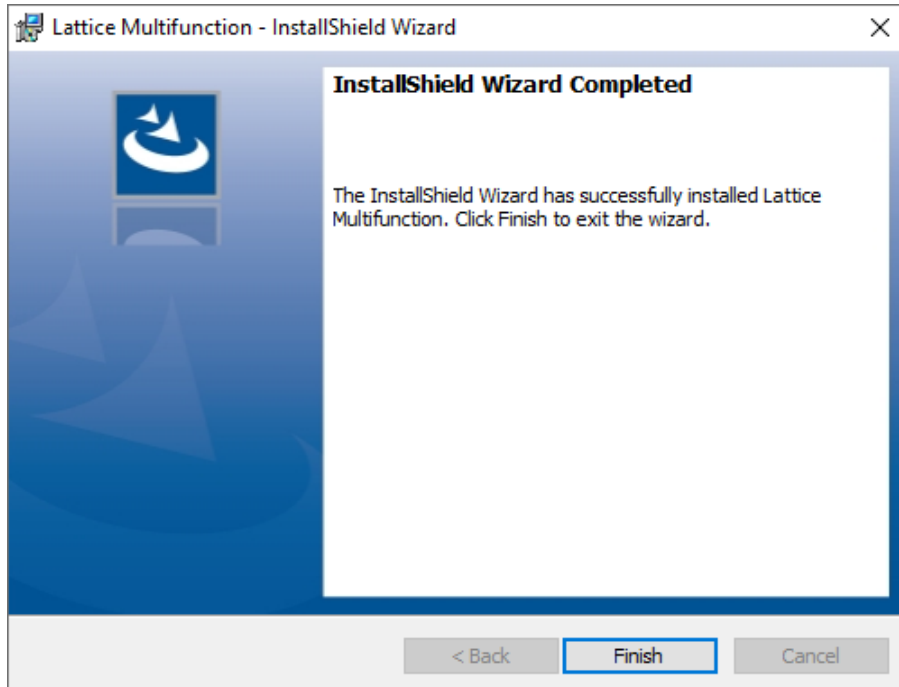


Figure 5.18. Windows Installer: Application Installed

6. After that, a message box appears. Click **Yes**.

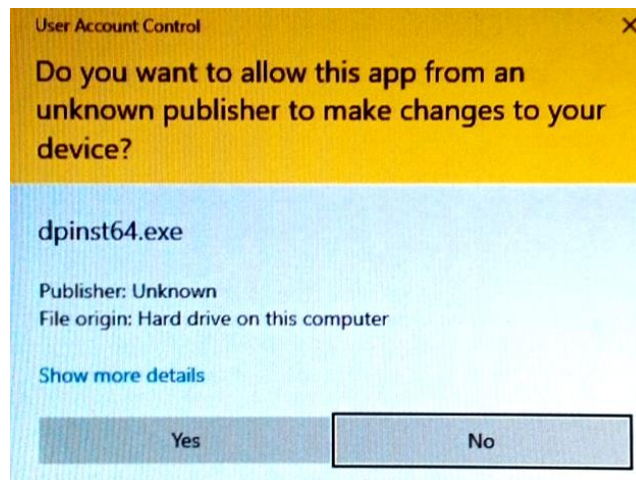


Figure 5.19. Device Configuration Prompt

7. The device driver installation wizard opens. Click **Next**.

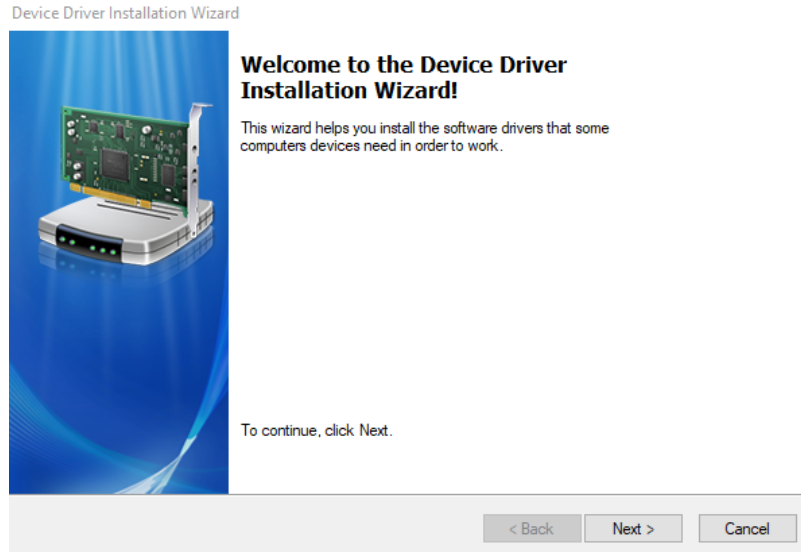


Figure 5.20. Device Driver Installation Wizard

8. If the user receive a Windows Security prompt, select Install this driver software anyway as shown in Figure 5.21.

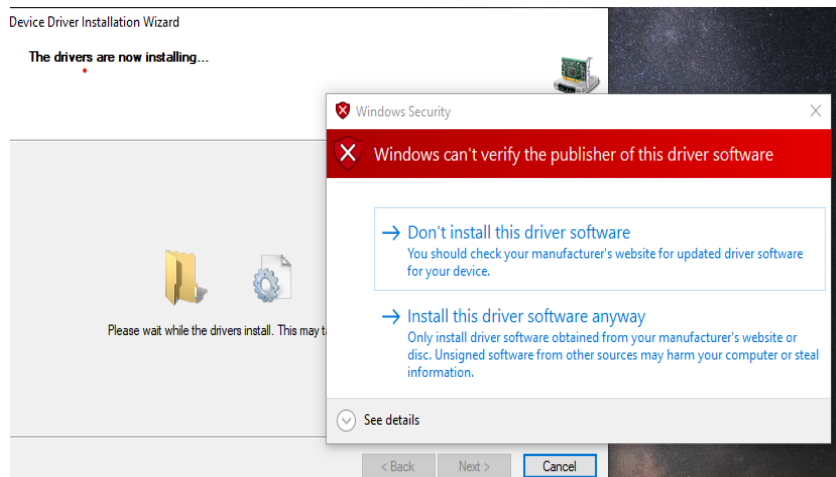


Figure 5.21. Windows Security in Driver Installation

9. If the driver is installed successfully, a message is displayed as shown in Figure 5.22.

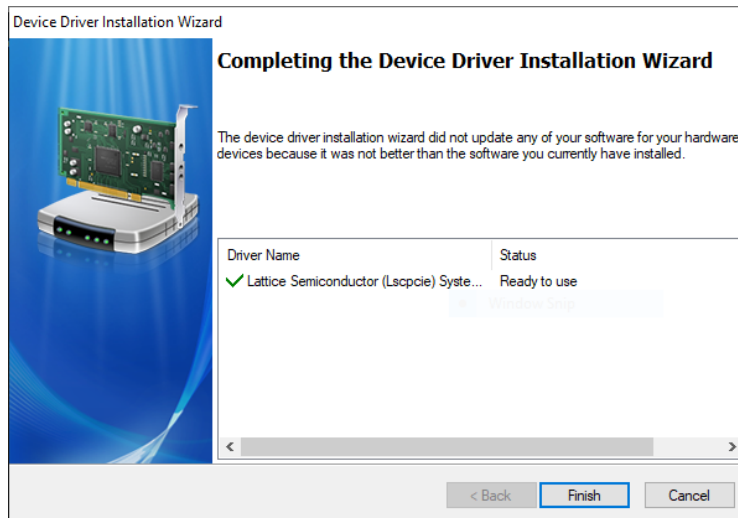


Figure 5.22. Device Driver Installation Completed

Installing Multifunction Demo Device Driver Manually

The drivers for the Multifunction Demo can be found in the *Demonstration\Windows10\Driver* folder.

To install the Multifunction Demo device driver manually:

1. Open **Device Manager**, which shows the connected PCIe devices, as shown in [Figure 5.23](#).

Note: If the **Device Manager** does not show the connected PCIe device, see the [Troubleshooting](#) section.

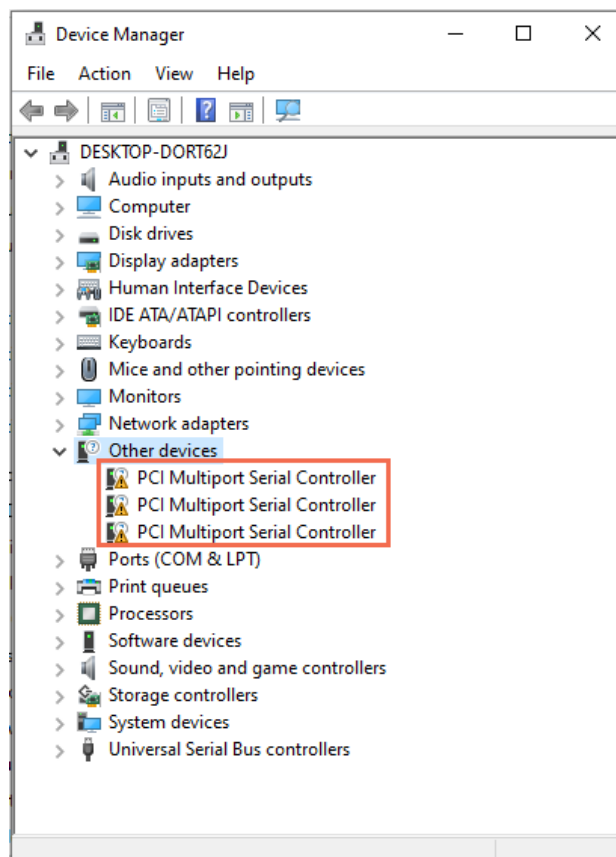


Figure 5.23. Device Manager

- Right-click on each device and select **Properties** as shown in [Figure 5.24](#).

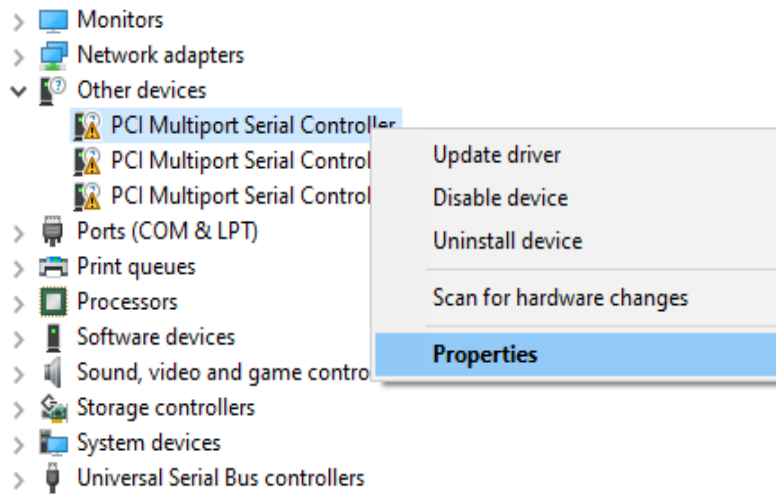


Figure 5.24. Showing Device Properties

- The information in Hardware IDs is needed to install the correct driver for the corresponding device. The Hardware IDs of the I²C, and GPIO devices are shown in [Table 5.1](#).

Table 5.1. Device IDs

Sl. No	Certus-NX
I ² C	PCI\VEN_1204&DEV_9C2F&SUBSYS_E00419AA
GPIO	PCI\VEN_1204&DEV_9C30&SUBSYS_E00419AA

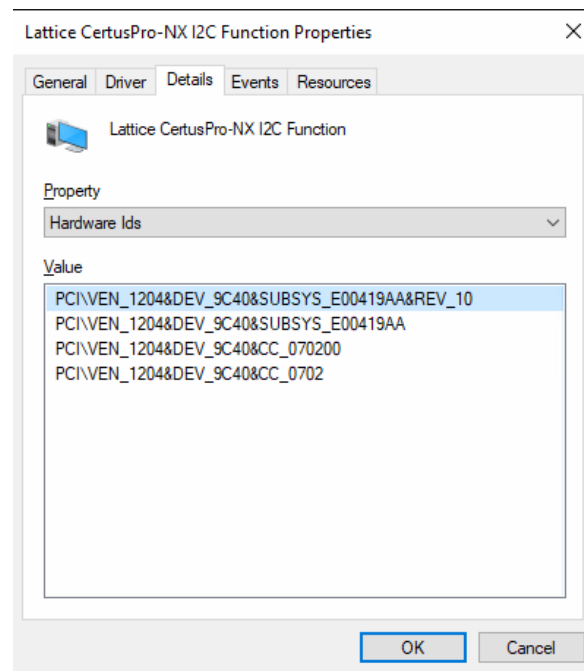


Figure 5.25. Hardware IDs of CertusPro-NX I²C Device

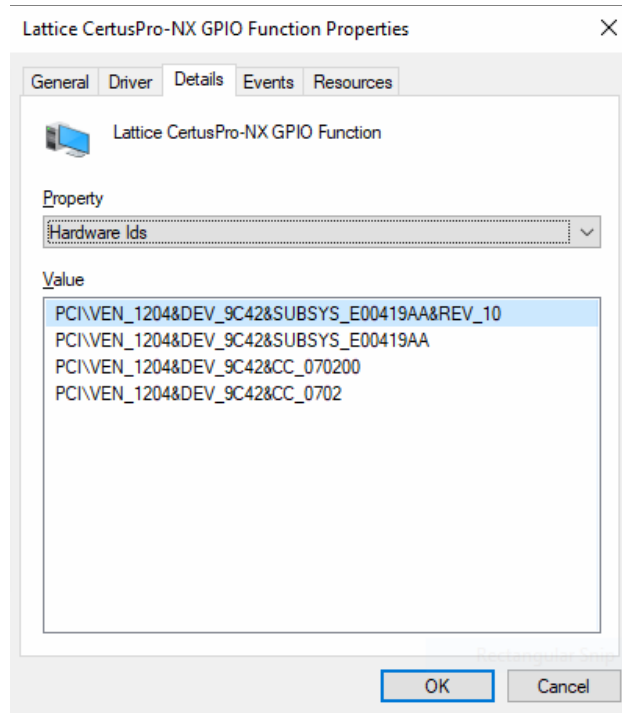


Figure 5.26. Hardware IDs of CertusPro-NX GPIO Device

4. Right click on the device and select **Update driver**.

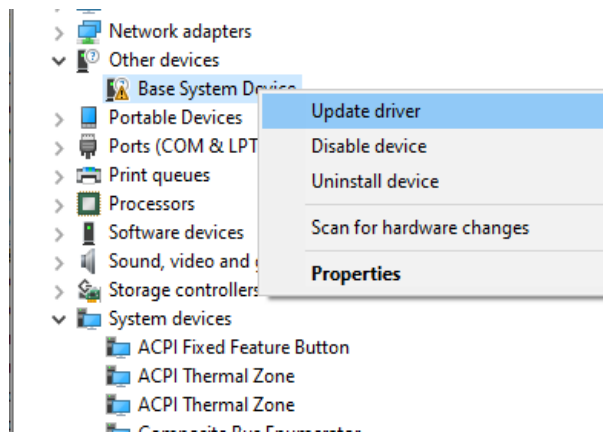


Figure 5.27. Update Driver Menu in Device Manager

5. Select the **Browse my computer for driver software** option as shown in [Figure 5.28](#).

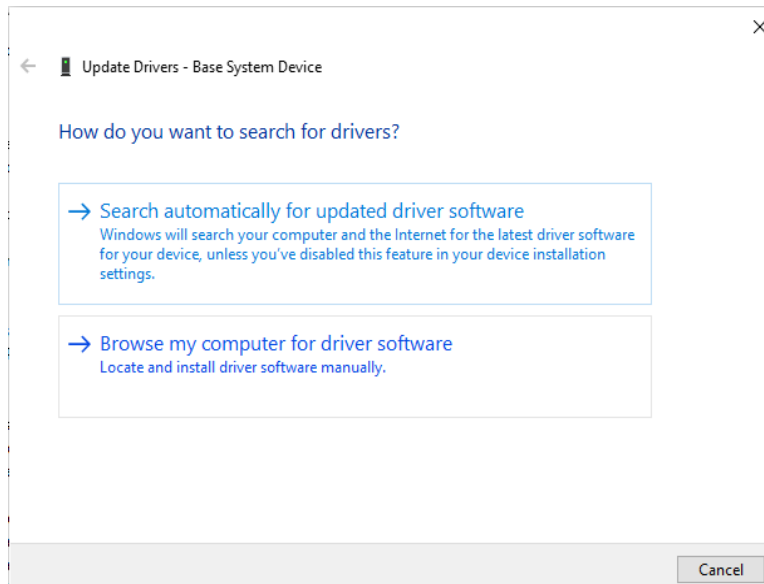


Figure 5.28. Update Driver Options

6. Browse for I²C or GPIO driver.

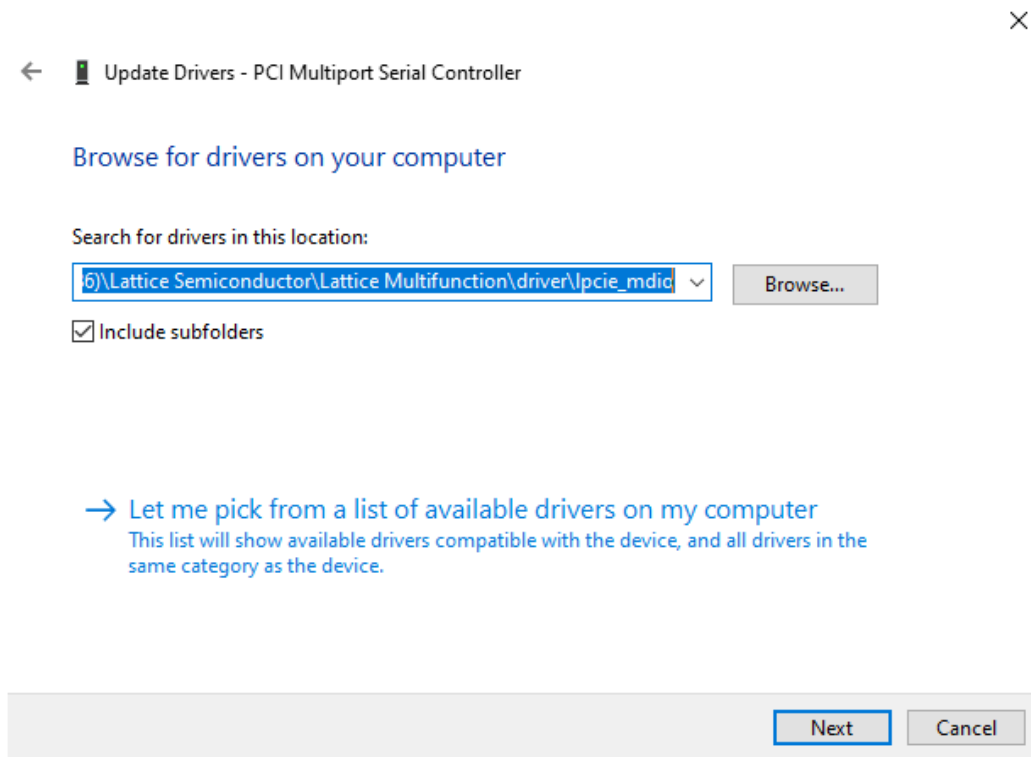


Figure 5.29. Browse the Driver for Device

7. If the user receives a Windows Security prompt, select **Install this driver software anyway** as shown in Figure 5.30.

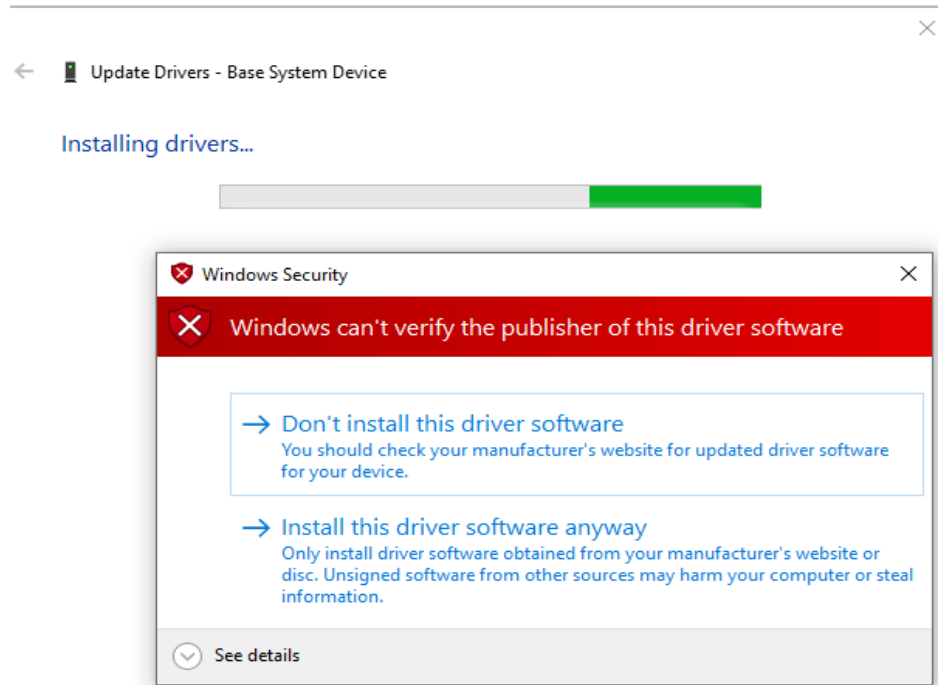


Figure 5.30. Windows Security in Device Manager

8. If the driver is installed successfully, a message is displayed as shown in [Figure 5.31](#).

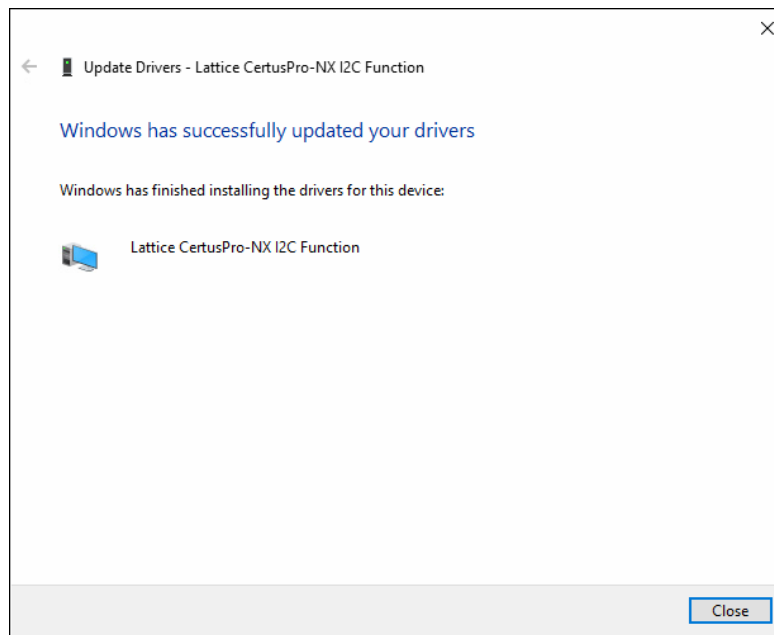


Figure 5.31. I²C Driver Installation Status Message for CertusPro-NX

9. Follow steps 3 through 8 for GPIO devices. After successful installation, all the two devices should be displayed in Device Manager as shown in [Figure 5.32](#).

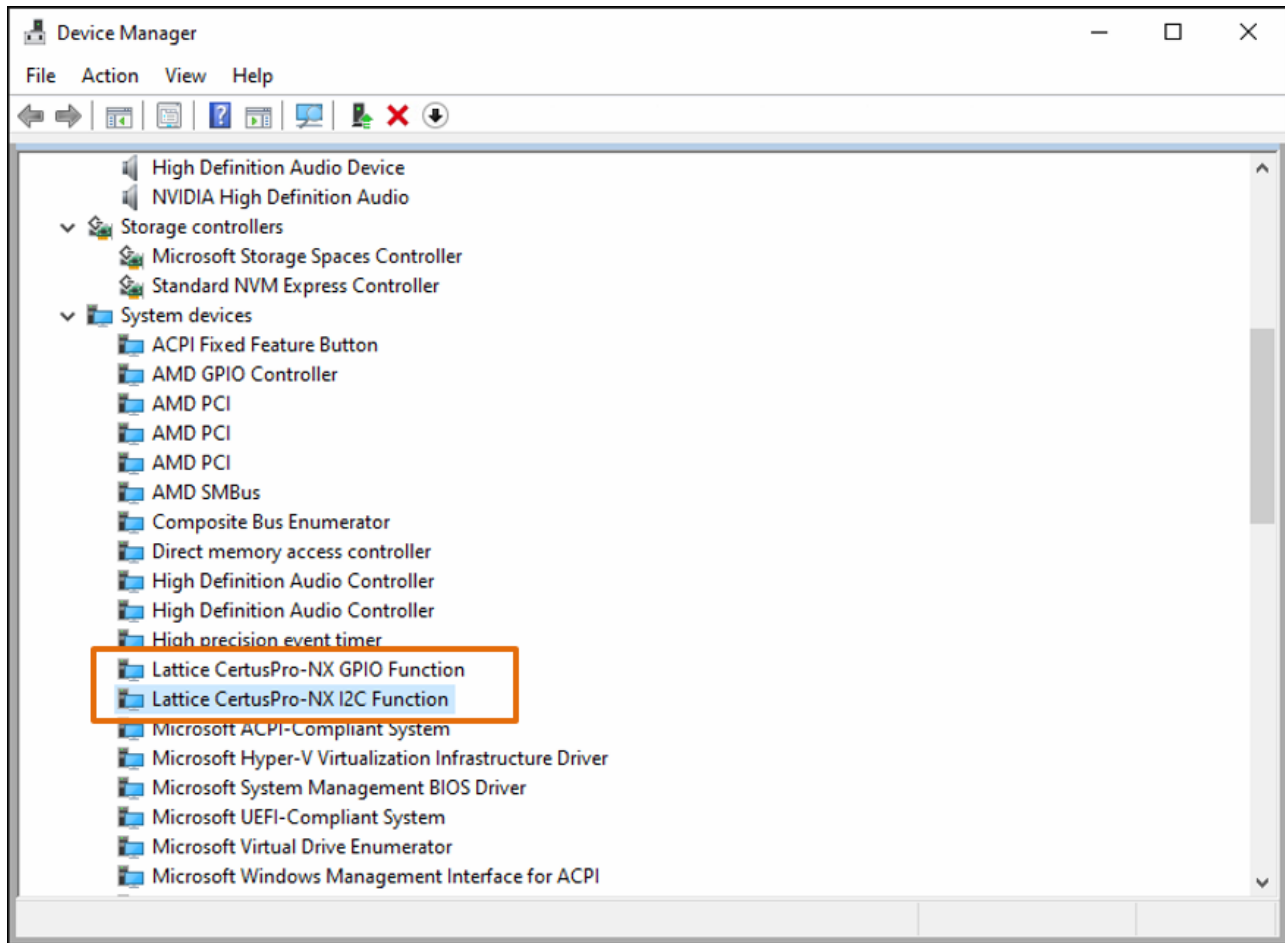


Figure 5.32. I²C and GPIO Device Drivers for CertusPro-NX in Device Manager

5.2.1.4. Driver Uninstall Manually

Follow the steps below to uninstall the device driver.

1. Open **Device Manager** window. Look for the device that user wants to uninstall.
2. Right click on the **device driver** and select **Uninstall device** as shown in [Figure 5.33](#).

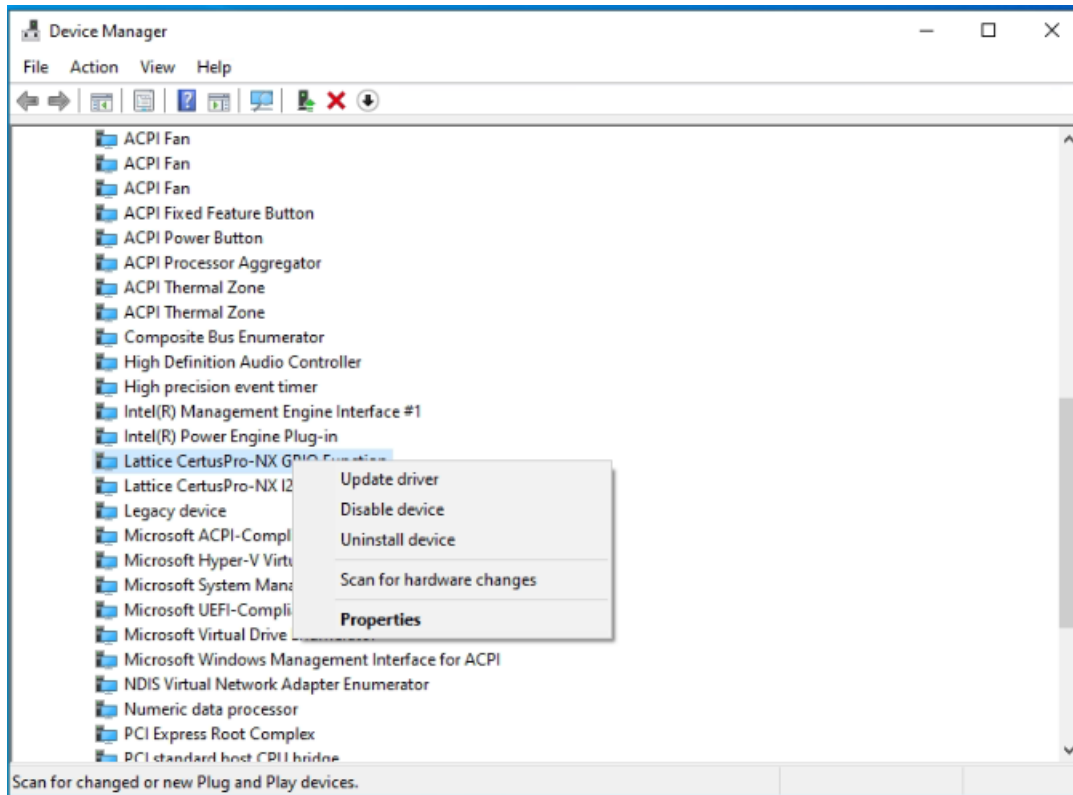


Figure 5.33. Select “Uninstall device” that User Want to Uninstall

3. Checked the option **Delete the driver software for this device** and click on **Uninstall** as shown in Figure 5.34.

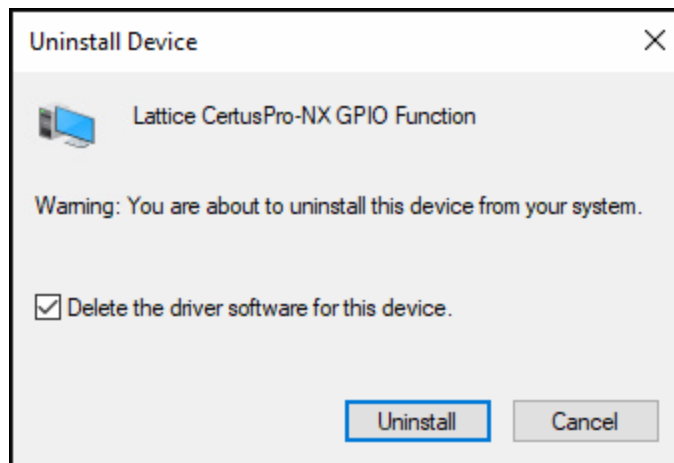


Figure 5.34. Select the Check Box and Click on Uninstall Button

4. In the Device Manager, Click **Action menu** and select **Scan** for hardware changes. Make sure that the uninstalled device is not listed in the Device Manager.

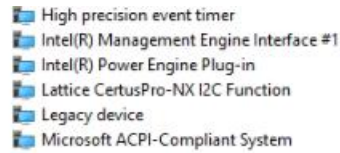


Figure 5.35. Uninstalled Device

5.2.2. Software Setup for Linux

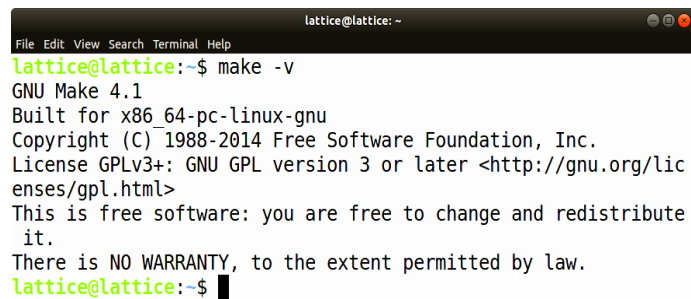
5.2.2.1. Supported Operating System

- Distribution: Ubuntu
- Description: Ubuntu 18.04.6 LTS
- Release: 18.04.6
- OS Type: 64 bit
- Codename: bionic

5.2.2.2. Required Packages

To check whether the required packages are installed or not, run the following commands on a terminal as shown below.

```
make -v
```



```
lattice@lattice: ~  
File Edit View Search Terminal Help  
lattice@lattice:~$ make -v  
GNU Make 4.1  
Built for x86_64-pc-linux-gnu  
Copyright (C) 1988-2014 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute  
it.  
There is NO WARRANTY, to the extent permitted by law.  
lattice@lattice:~$
```

```
gcc -v
```

```
lattice@lattice: ~  
File Edit View Search Terminal Help  
lattice@lattice:~$ gcc -v  
Using built-in specs.  
COLLECT_GCC=gcc  
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wr  
apper  
OFFLOAD_TARGET_NAMES=nvptx-none  
OFFLOAD_TARGET_DEFAULT=1  
Target: x86_64-linux-gnu  
Configured with: ../src/configure -v --with-pkgversion='Ubu  
ntu 7.5.0-3ubuntu1~18.04' --with-bugurl=file:///usr/share/d  
oc/gcc-7/README.Bugs --enable-languages=c,ada,c++,go,brig,d  
fortran,objc,obj-c++ --prefix=/usr --with-gcc-major-versio  
n-only --program-suffix=-7 --program-prefix=x86_64-linux-gn  
u- --enable-shared --enable-linker-build-id --libexecdir=/u  
sr/lib --without-included-gettext --enable-threads=posix --  
libdir=/usr/lib --enable-nls --enable-bootstrap --enable-cl  
ocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=  
yes --with-default-libstdcxx-abi=new --enable-gnu-unique-ob  
ject --disable-vtable-verify --enable-libmpx --enable-plugi  
n --enable-default-pie --with-system-zlib --with-target-sys  
tem-zlib --enable-objc-gc=auto --enable-multiarch --disabl  
e-werror --with-arch-32=i686 --with-abi=m64 --with-multilib  
-list=m32,m64,mx32 --enable-multilib --with-tune=generic --e  
nable-offload-targets=nvptx-none --without-cuda-driver --en  
able-checking=release --build=x86_64-linux-gnu --host=x86_6  
4-linux-gnu --target=x86_64-linux-gnu  
Thread model: posix  
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)  
lattice@lattice:~$
```

g++ -v

```
lattice@lattice: ~  
File Edit View Search Terminal Help  
lattice@lattice:~$ g++ -v  
Using built-in specs.  
COLLECT_GCC=g++  
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/7/lto-wr  
apper  
OFFLOAD_TARGET_NAMES=nvptx-none  
OFFLOAD_TARGET_DEFAULT=1  
Target: x86_64-linux-gnu  
Configured with: ../src/configure -v --with-pkgversion='Ubu  
ntu 7.5.0-3ubuntu1~18.04' --with-bugurl=file:///usr/share/R  
EADME.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,o  
bj-c++ --prefix=/usr --with-gcc-major-version-only --program-suf  
fix=-7 --program-prefix=x86_64-linux-gnu- --enable-shared --enab  
le-linker-build-id --libexecdir=/usr/lib --without-included-gett  
ext --enable-threads=posix --libdir=/usr/lib --enable-nls --enab  
le-bootstrap --enable-clocale=gnu --enable-libstdcxx-debug --ena  
ble-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable  
-gnu-unique-object --disable-vtable-verify --enable-libmpx --ena  
ble-plugin --enable-default-pie --with-system-zlib --with-target  
-system-zlib --enable-objc-gc=auto --enable-multiarch --disabl  
e-werror --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m  
32,m64,mx32 --enable-multilib --with-tune=generic --enable-offlo  
ad-targets=nvptx-none --without-cuda-driver --enable-checking=re  
lease --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=  
x86_64-linux-gnu  
Thread model: posix  
gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)  
lattice@lattice:~$
```

5.2.2.3. Packages Installation Steps

Run the following commands on a terminal to install the required packages.

```
sudo apt update  
sudo apt install build-essential  
sudo apt install flex
```


5.2.2.4. Automatic Setup and Installation

To setup the demo in automatic mode:

1. Go to the *Demonstration/Linux* directory.
2. Change the permission of the *install.sh* file by running *chmod* command.

```
sudo chmod 777 install.sh
```
3. Run the command below, which builds the driver and API library, install the driver, and launch the QT use interface application.

```
sudo ./install.sh
```
4. To uninstall the driver, go to the *Demonstration/Linux* directory and run the command below.

```
sudo ./uninstall.sh
```

5.2.2.5. Manual Setup and Installation

Note: User may skip this section if user have installed the driver successfully in the previous section.

Before installing the driver, the driver must be built.

To build the driver:

1. Go to the *Demonstration/Linux* directory.
2. Run the following command on terminal.

```
sudo chmod 777 -R Source_Code
```
3. Go to in *Source_Code* directory.
4. Run the following commands.

```
sudo make clean  
sudo make
```

This builds the driver and API library.

5. Make sure that the driver is not installed.
6. To remove an existing driver, run one or all the following command in a terminal window:

```
sudo rmmod gpio_main.ko  
sudo rmmod i2c_main.ko  
sudo rmmod mdio_main.ko
```

To install the drivers:

1. Go to *drv_src/gpio_drv/* directory.
2. Run the command below.

```
sudo insmod gpio_main.ko
```

 - a. Repeat the same procedure for the *i2c_drv* and *mdio_drv* directories. *mdio_drv* is not applicable for CertusPro-NX.
3. To launch the user interface application, go to the *app_src/gui/deploy/* directory and run the command below.

```
sudo ./MFDemo.sh
```

6. Application Overview

The Mixed Mode application uses the PCIe Multifunction Application to demonstrate the multifunction capabilities of the FPGA PCIe IP and 8B10B Protocol. The application software allows the user to access GPIO and I²C registers for PCIe and provides real time interaction with the FPGA hardware to demonstrate a functional PCI Express communications path between the software and the FPGA IP. The application software also allows user to trigger the random pattern generator and checker used in the 8B10B protocol.

6.1. Running the Mixed Mode Demo Application

On Linux, the demo application can be launched by running the `./MFDemo.sh` script, which is located in the `Demonstration/Linux/Source_Code/app_src/gui/deploy` directory. On Windows, the application can be launched by clicking Lattice Multifunction shortcut present on the desktop.

The graphical user interface opens the PCI Express Test Application software with the **Device Info** tab selected, as shown in [Figure 6.1](#). Note that some of the information may vary between different FPGA devices (such as the Device ID).

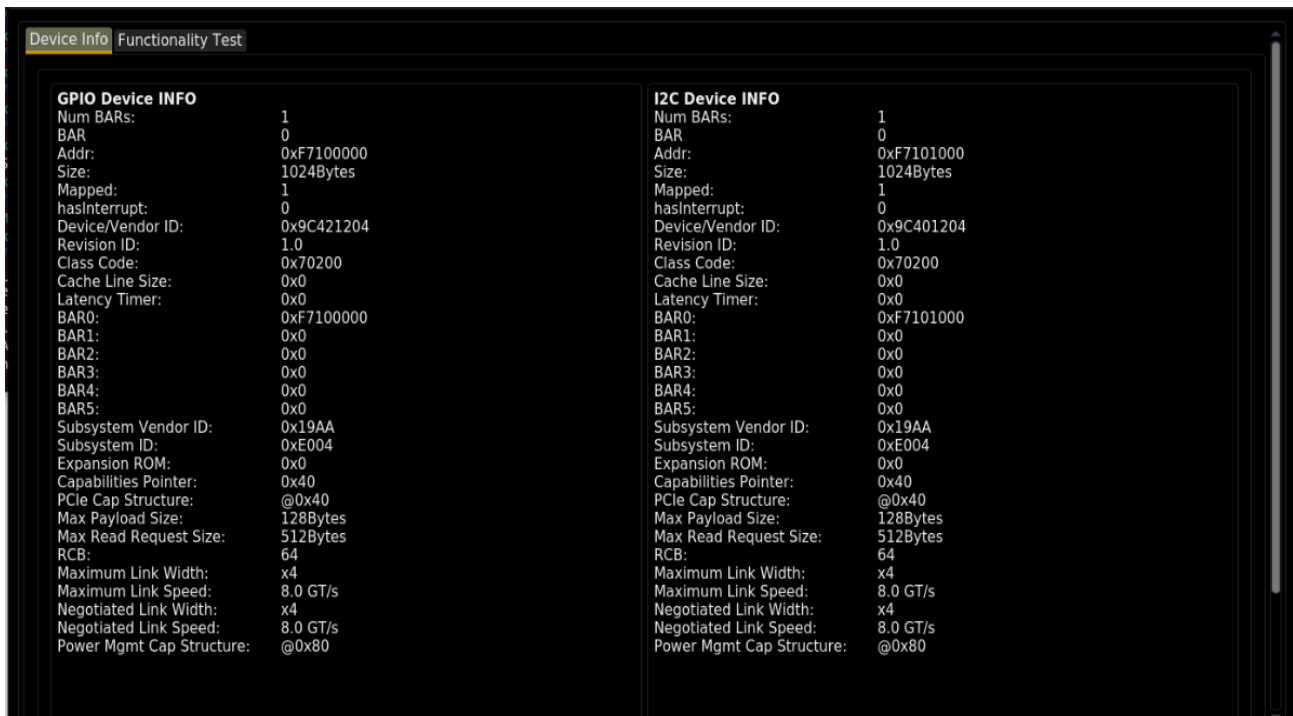


Figure 6.1. PCIe Test Application Device Info Tab

6.2. Using the Mixed Mode Demo Application User Interface

The PCI Express Device Info page displays information about the device driver and device's PCI configuration registers. The data displayed is read-only and cannot be edited. If there is a problem loading the driver or accessing the device, an error message appears on this page. The **"Run Counter"** button in the Functionality tab is used to trigger of start of random pattern generator and checker used in 8B10B protocol.

6.2.1. Functionality Test Tab

The user can read or write from or to the device using controls present on this tab. [Figure 6.2](#) shows the Functionality Test tab.



Figure 6.2. Functionality Test Tab

6.2.2. GPIO

This section is used to interact with the GPIO endpoint function.

6.2.2.1. Input Switch

There are five input LEDs present in the GPIO input box. These LEDs change status when the user change the position of the input switches on the board. This mapped to SW1 on the board, and there is one LED per input switch. To start reading input from the switches, click the **“Read Input”** button. An example of the LEDs is shown in [Figure 6.3](#).

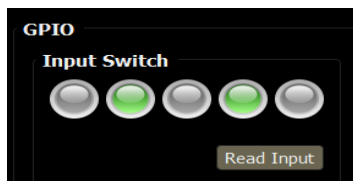


Figure 6.3. GPIO Input Switch

6.2.2.2. Output LED

The Output LED section shows the status of eight output LEDs. These LEDs represent the five bits of a counter, which begin to increment when the user click the “Run Counter” button on the user interface, as shown in Figure 6.4.

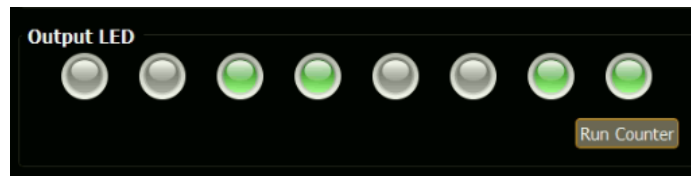


Figure 6.4. GPIO Output LEDs

Run Counter

The “Run Counter” switches the Demo LED within the CertusPro-NX Versa Evaluation board to be turned ON and OFF following the GPIO Output LED as shown in Figure 6.5.

7 out of 8 (LED_0 till LED_6) of the Demo LED follows the GPIO Output LED.



Figure 6.5. Demo LEDs

The “Run Counter” trigger starts the Random Pattern generator and checker used in the 8B10B protocol. The data transmitted through the 8B10B protocol is looped back from Tx to Rx at the PMA using the serial loopback feature. Random pattern generator and checker signals are shown in Reveal Analyzer.

Reveal Analyzer

Reveal Analyzer can be opened using Radiant software version 2022. Reveal Analyzer can be opened by clicking the highlighted button as shown in Figure 6.6.



Figure 6.6. Open Reveal Analyzer

Configure Reveal Analyzer

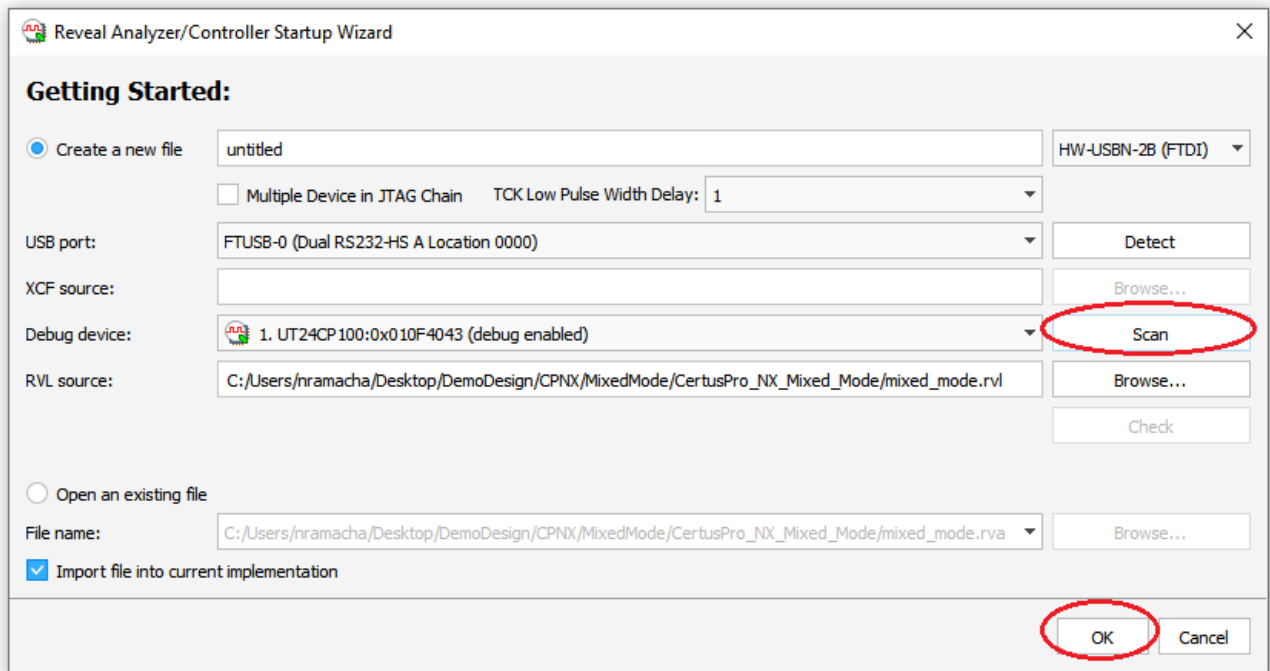


Figure 6.7. Configure Reveal Analyzer

Reveal Analyzer can be configured as shown in Figure 6.7. The Reveal Analyzer source file is `mixed_mode.rvl`, which is located in the `Hardware\CertusPro_NX_Mixed_Mode\LFPCNX_100_Mixed_Mode` folder. Select “Scan” and “Ok” to configure the Reveal Analyzer to be ready to capture.

Reveal Analyzer Capture

The “Run Counter” will trigger start the Random Pattern Generator and Checker. The Reveal Analyzer in Figure 6.8 shows the behavior before the Random Pattern Generator and Checker started. Before the Random Pattern Generator starts, the Mixed Mode demo designs transmits word alignment pattern “BC” through the 8B10B Protocol to the Tx. The Mixed Mode demo design uses serial loopback from Tx to Rx, hence the Rx will receive BC on the first byte and attempt to determine the word boundary. Word boundary determined correctly by the Rx is indicated by assertion of signal `mpcs_get_Isync` as shown in Figure 6.8. Once the word boundary is determined, then the Mixed Mode demo design will check if the random “Run Counter” has triggered start.

When the “Run Counter” is triggered start, Random Pattern Generator will start transmitting random patterns, this can be observed through `mpcs_tx_ch_din_byte0`, `mpcs_tx_ch_din_byte1`, `mpcs_tx_ch_din_byte2`, and `mpcs_tx_ch_din_byte3` signals. The signal received at the Rx is sent over to `mpcs_rx_ch_dout_byte0`, `mpcs_rx_ch_dout_byte1`, `mpcs_rx_ch_dout_byte2`, and `mpcs_rx_ch_dout_byte3` signals which sends the signal to Random Pattern Checker. If the Random Pattern Checker receives 32 bytes of data correctly, the `random_pattern_detected` signal will be asserted indicating that data has been transmitted and received with no bit errors using 8B10B Protocol as shown in Figure 6.9.

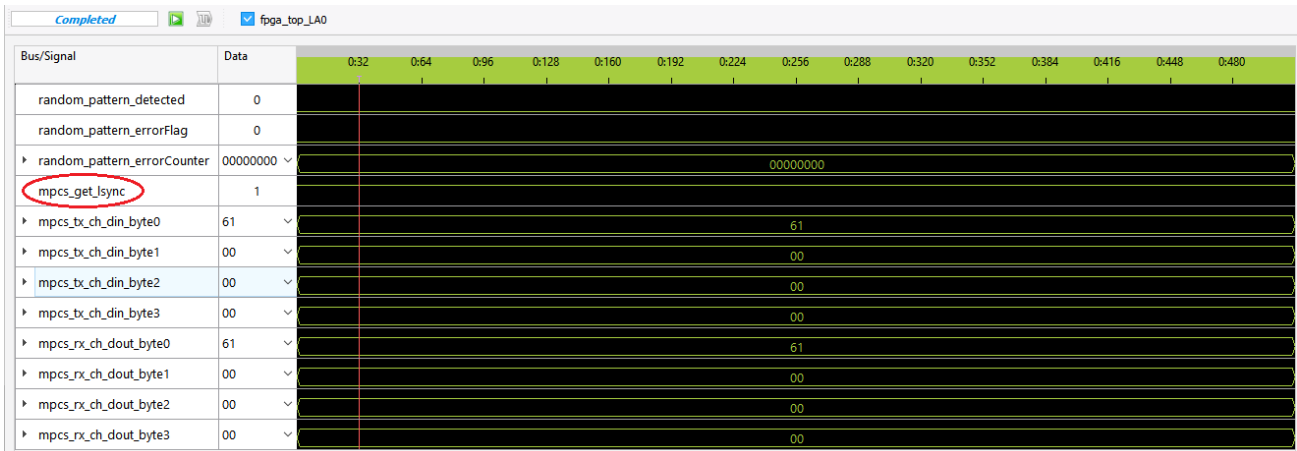


Figure 6.8. Reveal Analyzer Capture Before Press “Run Counter”

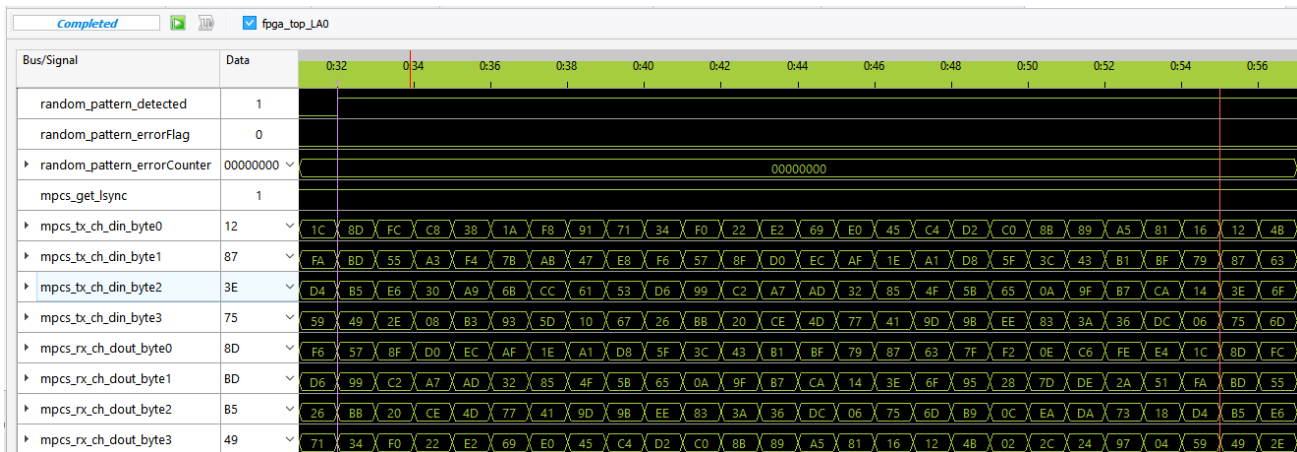


Figure 6.9. Reveal Analyzer Capture After Press “Run Counter”

Pattern Detected

The Demo LED (LED_7) will switch ON indicating the transmitted random pattern has been detected with no bit errors using 8B10B Protocol. This is also indicated by *random_pattern_detected* signal assertion.



Figure 6.10. LED_7 Switch ON To Indicate Pattern Detected with No Bit Errors

6.2.3. I²C

This section describes how to use the GUI to read and write the I²C device.

6.2.3.1. Test Mode

This control has two options for testing the I²C interface: Normal Mode and Batch Mode. In Normal Mode, the user enters each I²C command manually. In Batch Mode, the I²C commands are loaded from a script.

6.2.3.2. Settings

This control is used to configure the I²C bit-rate. Two bit-rates, 100 kHz and 400 kHz, are supported and can be selected from the Bitrate drop down box as shown in Figure 6.11. After selecting the desired speed from the drop-down box, the user must click the Set button to apply the settings to the device.



Figure 6.11. I²C Bit-Rate Selection

6.2.3.3. Master Write

The Master write option is used to write the data to the I²C device. The user can either manually type the address and/or the data information into the Message box, or user can load the data from a hex file by clicking on the **Load** button. Clicking on the Master Write button initiates the write operation. The interface is shown in Figure 6.12.

The user can clear or save the message box contents into a file. To do so, click the **Save** button, provide the desired path, enter the name of the file, and click **OK**. This saves the data into the specified hex file. The I²C interface is spot-checked using the camera: Sony IMX258-0AQH5. For CertusPro-NX, the camera module must be obtained separately.

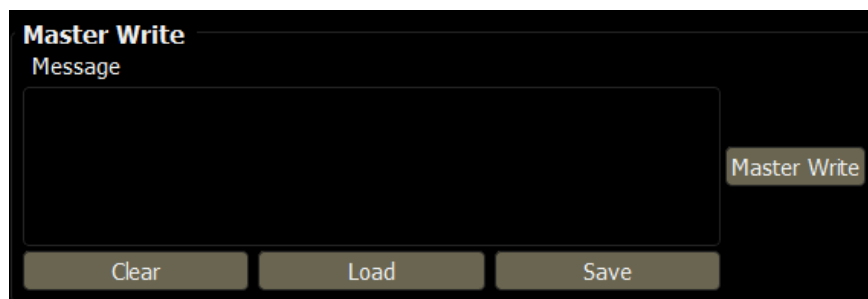


Figure 6.12. I²C Master Write

Table 6.1. Master Write Control Description

Controls	Description
Clear	Clears the input box
Load	Loads the file configuration hex file into the input box
Save	Saves the input box data into the hex file
Master Write	Writes the input box data on the I ² C device

6.2.3.4. IMX258-0AQH5 Camera Write Protocol

Figure 6.13 shows the communication protocol format for the IMX258 camera for a single write to an arbitrary address. Figure 6.14 shows the format for a sequential write starting from an arbitrary address.

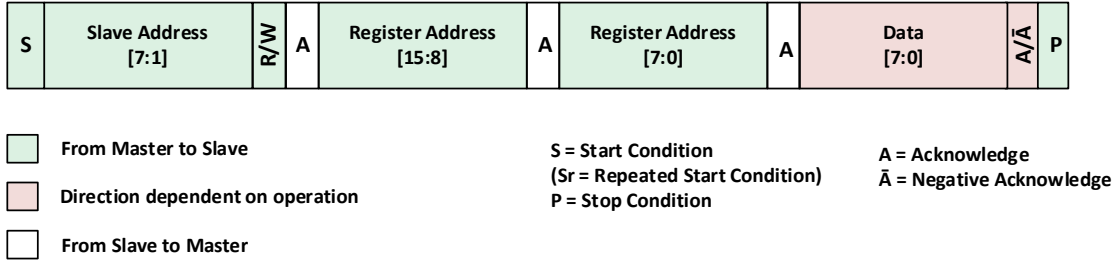


Figure 6.13. Single Write to an Arbitrary Address of IMX258-0AQH5 Camera

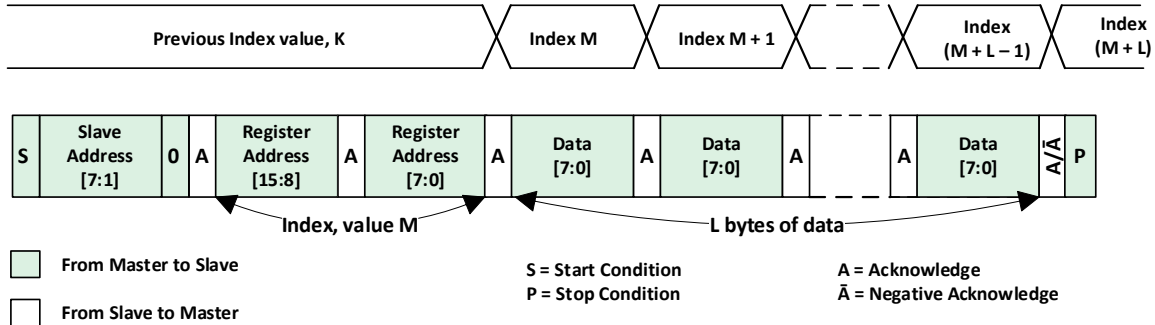


Figure 6.14. Sequential Write Starting from an Arbitrary Address of IMX258-0AQH5 Camera

To test the I²C interface, data can be written to the register 0x3093 of the camera, as shown in Figure 6.15. Additional register information can be obtained through the IMX258-0AQH5 camera data sheet available through Sony. An example of how to execute this transaction is discussed in the next section.

Table 6.2. Registers of MX258-0AQH5 CMOS Camera

Address	Bit	Name	Description
0x3092	[1:0]	MANUAL_DATA PEDESTAL_VALUE [9:8]	Manual setting value for Data Pedestal Default setting value: 0x40
0x3093	[7:0]	MANUAL_DATA PEDESTAL_VALUE [7:0]	
0x3090	[0]	MANUAL_DATA PEDESTAL_ENABLE	Output black level is controlled with Manual_DTA_PEDESTAL_VALUE register value 0x0 : pedestal = 0x40 (fixed value) 0x1 : pedestal = MANUAL_DATA_PEDESTAL_VALUE Else : inhibited value
0x0340	[7:0]	FRM_LENGTH_LINES [15:8]	The length of frame Unit: lines Format: 16-bit unsigned integer *set value for given capture mode ≤ 65525d
0x0341	[7:0]	FRM_LENGTH_LINES [7:0]	
0x0342	[7:0]	LINE_LENGTH_PCK [15:8]	
0x0343	[7:0]	LINE_LENGTH_PCK [7:0]	

Master Write Procedure

To perform a write operation to the camera's register address 0x3093:

1. Select **Normal Mode** from the **Test Mode** drop down box.
2. Select **100 kHz** bit-rate from the Bitrate drop down box and click the **Set** button.
3. Enter **0x1A** in the **Slave addr** box.
4. Select the **7-bit mode** radio button.
5. Enter the data and address into the message box – two bytes of address and one byte of data as shown in [Figure 6.15](#). For example, to write address 0x3093 with 0xFF data, then enter “**30 93 FF**” into the message box.
6. To initiate the transaction, click the **Master Write** button. To verify that the write operation is successful, follow the instructions on how to perform a Master Read operation.

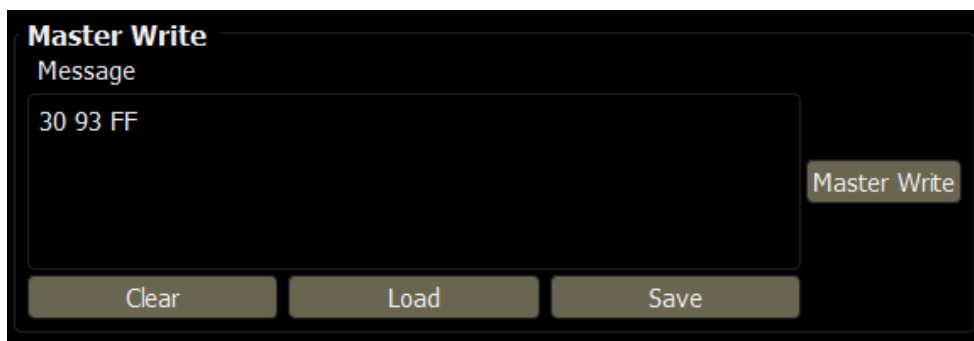


Figure 6.15. I²C Master Write Procedure

6.2.3.5. Master Read

The Master read option is used to read the data from the I²C device. This section describes the Master read procedure with an example with IMX258-0AQH5 camera.

Master Read from a Held Address

The Master Read option is used to read data from the held address of the I²C device. When a master transmits the slave address but does not designate an index or address, the previously read from address value is used or held. A single byte or sequential data can be read from the held address.

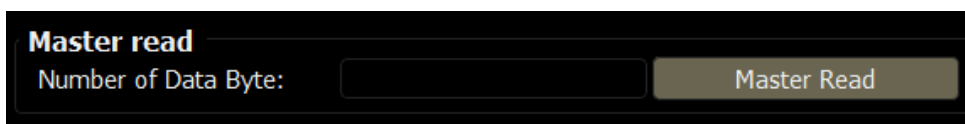


Figure 6.16. I²C Master Read

IMX258-0AQH5 Camera Read Protocol from a Held Address

[Figure 6.17](#) shows the communication protocol format for the IMX258 camera for a single read from the held address.

[Figure 6.18](#) shows the format for a sequential read starting from the held address.

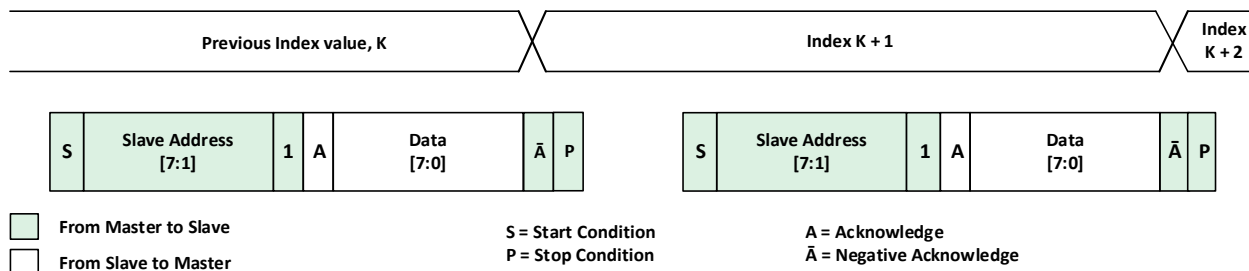


Figure 6.17. Single Read from the Held Address of IMX258-0AQH5 Camera

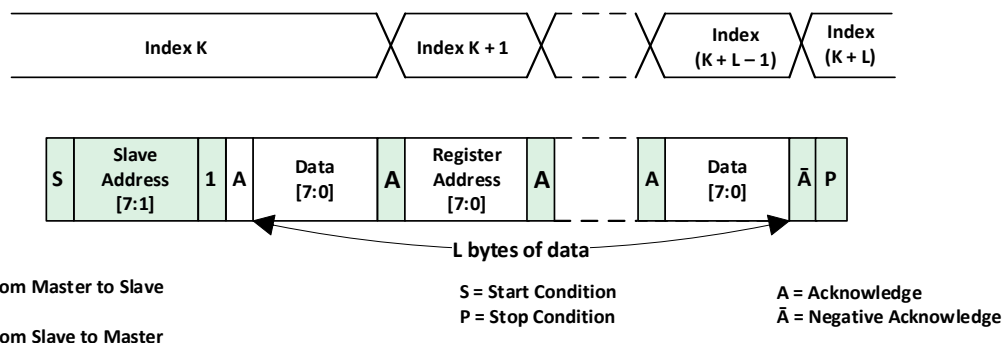


Figure 6.18. Sequential Read Starting from the Held Address of IMX258-0AQH5 Camera

An example of performing a single read from the held address (in this example, 0x3093) is shown below.

To perform a Master Read from a held address:

1. Select **Normal Mode** from the **Test Mode** drop down box.
2. Select **100 kHz** bit-rate from the Bitrate drop down box and click the **Set** button.
3. Enter **0x1A** in the **Slave addr** box.
4. Select the **7-bit mode** radio button.
5. Enter the register address in the Master Write message box as shown in Figure 6.19 to set the current register address. For an example, to read from 0x3093 register address, enter **30 93** in the message box and click the **Master Write** button.



Figure 6.19. I²C Master Write to Write the Register Address

- Enter the number of bytes to read in the **Number of Data Byte** box as shown in [Figure 6.20](#) and click **Master Read**. The results of the read transaction are shown in the transaction log table, and the data value read out can be found in the data column of the transaction log table.

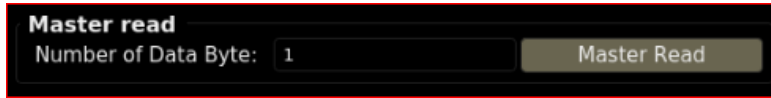


Figure 6.20. I²C Master Read

Master Register Read from an Arbitrary Address

The Master Register Read option is used to read data from an arbitrary address of the I²C device. A single byte or sequential data can be read from an arbitrary address.

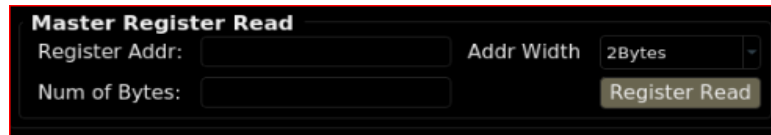


Figure 6.21. I²C Master Register Read

Table 6.3. Master Register Read Control Description

Controls	Description
Register Addr	Specifies the register address to be read.
Addr Width	Specifies the address width. Valid values are 1 Byte and 2 Bytes.
Num of Bytes	Specifies the number of bytes to be read from the I ² C device.
Register Read	When this button is pressed, the read operation is performed.

IMX258-0AQH5 Camera Read Protocol from an Arbitrary Address

[Figure 6.22](#) shows the communication protocol format single read from an arbitrary address and [Figure 6.23](#) shows the sequential read starting from an arbitrary address of MX258-0AQH5 Camera.

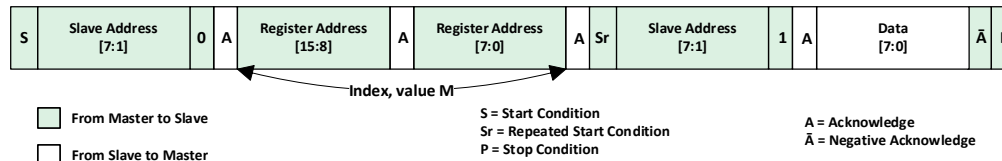


Figure 6.22. Single Read from an Arbitrary Address of MX258-0AQH5 Camera

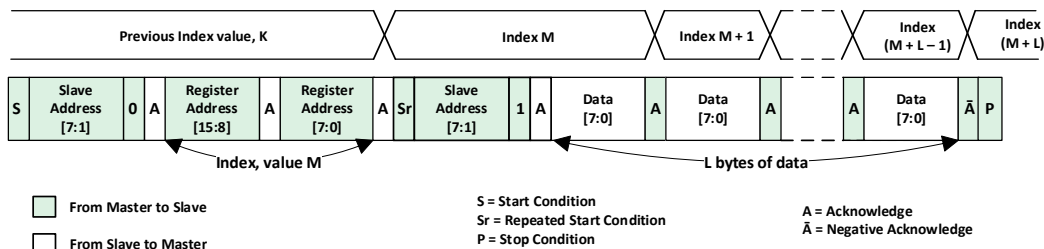


Figure 6.23. Sequential Read Starting from an Arbitrary Address of MX258-0AQH5 Camera

To perform a Master Read operation:

- Select **Normal Mode** from the **Test Mode** drop down box.
- Select either **100 kHz** or **400 kHz** from the Bitrate drop down box and click the **Set** button.

3. Enter **0x1A** in **Slave addr**.
4. Select the **7-bit mode** radio button.
5. Enter **3093** in the **Register Addr** field.
6. Select **2Bytes** in **Addr Width**.
7. Enter **1** in **Num of Bytes**.
8. Click the **Register Read** button and check the transaction log for the readback value.

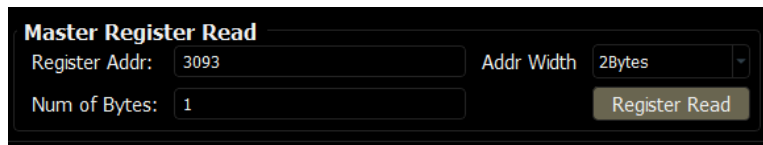


Figure 6.24. I²C Master Register Read

Note: if any step goes wrong, then I²C does not work without a power cycle.

6.2.3.6. Transaction Log

The Transaction Log shows the details of all transactions on the I²C device as shown in Figure 6.25. The user can save and clear the log. To save the log into file click **SaveToFile**, select the desired location and enter the log file name. The log is saved in text format. To clear the log from the table, click the **ClearLog** button.



Figure 6.25. I²C Master Register Read

The log table is updated only when the transaction is successfully completed. Failed transactions are not logged in the table. The description of each column of the transaction log table is described in Table 6.4.

Table 6.4. Transaction Log Control Description

Column	Description
Time	The start time of the transaction.
R/W	Specifies whether the transaction was a read or write.
Bit Rate	Specifies what bit rate was chosen for the transaction.
Slave Addr	Specifies the I ² C slave address of the transaction.
Length	Specifies the number of bytes transferred during the transaction.
Data	Specifies the transferred data during the transaction.

6.2.3.7. Batch Mode

Batch mode is used to read and write the I²C device using an XML file. The user can load an XML file by clicking the **Load** button. Click the **Execute** button to run the commands in the XML file. To abort the operation, press the **Stop** button. After completion, an **Updated successfully** message is displayed. An example of Batch mode XML is shown in Figure 6.26.

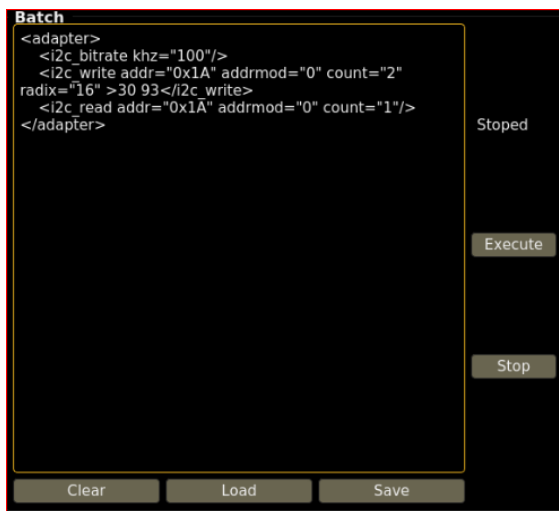


Figure 6.26. I²C Batch Mode Read/Write

XML node description:

- i2c_bitrate – elements used to configure the I²C device bit-rate.
- khz – attributes hold the I²C bit rate value.
- i2c_write – elements used to write address and data or address-only.
- addr – attribute that holds the I²C slave device address.
- addrmod – attribute that specifies the addressing mode, with a value of 0 for 7bit and 1 for 10bit.
- count – attribute that specifies the number of bytes to be written or read.
- radix – attribute that specifies the radix of the device address and data format.
- i2c_read – elements used to read data from the device.

Table 6.5. Batch Mode Control Description

Control	Description
Clear	Clears the input box.
Load	Loads the xml configuration file in the input box.
Save	Saves the input box configuration to a file.
Stopped	Indicates the Master device status when any operation is run.
Execute	Used to start batch file execution.
Stop	Used to stop the execution of the batch file.

7. Troubleshooting

7.1. SPI Flash Update

If you are getting a verification error while dumping the .bit file, try changing the TCK frequency to a value greater than 4. The TCK Divider Setting option is in the Cable Setup dialog box of the Lattice Radiant Programmer Window, as shown in [Figure 7.1](#). Restart programming by clicking the **Program** button.

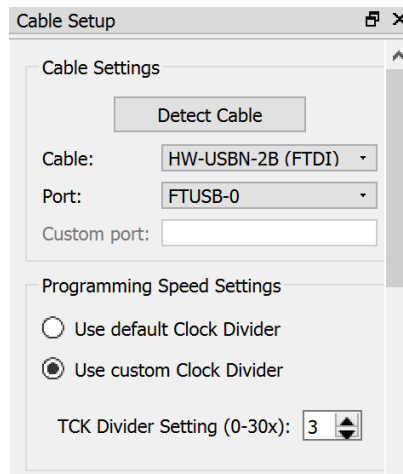


Figure 7.1. TCK Frequency Setting

If the verification error problem still occurs, press and hold the **PROGRAMN** push button on the board before clicking the **Program** button.

If the device is not detected on port FTUSB-0, then click **Detect cable** and select the port **FTUSB-1**.

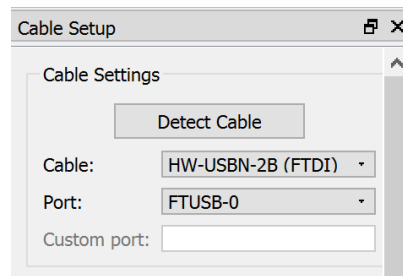


Figure 7.2. Port Selection

7.2. Driver Installation and User Interface Launch for Windows

7.2.1. Problem in Driver Installation

- If the hardware IDs are not found in Device Manager, ensure that the board is properly installed in a PCIe slot.
- Ensure that the board has a valid PCIe bitstream file loaded in the SPI Flash. Shut down the system and try another slot. If the board is present, check the Properties and the Resource tab to verify if memory is assigned to it.
- Also, verify if the Vendor ID and Device ID are valid, as seen by Windows Plug-n-Play. If the values are invalid, perhaps the bitstream file is corrupt and needs to be reloaded into SPI flash. If the PCIe board is shown in the list of Device Manager, install the driver manually.

- If the driver is not being installed even though the device is present in the Device Manager, make sure that the driver signature enforcement is disabled permanently. If not, follow the steps described in the Disabling Driver Signature Enforcement Permanently section.

7.2.2. Problem with Launching User Interface

Ensure that the driver is installed properly. Otherwise, the Device Info section of the user interface displays the message shown in Figure 7.3.

The *Driver Open [FAILED]* message is displayed if the driver is not installed properly. The Device information is present if the drivers are installed properly.

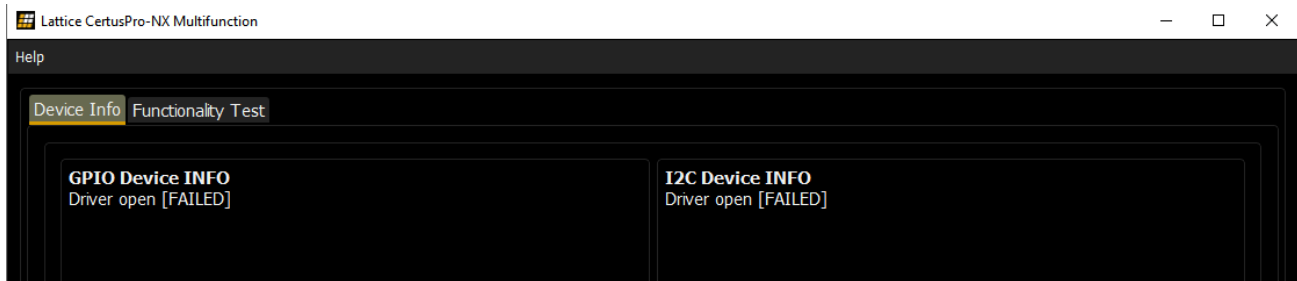


Figure 7.3. User Interface with No Device Driver

If any such error message occurs, make sure that the device is properly inserted in the system. After that, verify that drivers are installed. When all the issues are resolved, restart the user application.

When using file saving operations in the Memory portion of the demo, make sure that the user have write permissions for the location where the file is being saved. Similarly, for file reading operations, the directory from where the file is being loaded must have Read Permission.

If the Multifunction demo short cut icon is accidentally deleted, then the application can be launched by double clicking on *lattice_bd.exe* file present in *[INSTALLATION FOLDER]/Lattice Semiconductor/Multifunction Demo/bin* folder.

7.3. Driver Installation User Interface Launch for Linux

This section describes troubleshooting steps for Linux.

7.3.1. Problem in Driver Installation

The following error may occur during software driver installation.

Issue: insmod: ERROR: could not insert module *lattice_main.ko*: Operation not permitted.

Resolution: Ensure that the Secure Boot option is disabled in the BIOS of user's system. Refer to user's BIOS user manual to learn about the steps for turning off the Secure Boot option.

Issue: insmod: ERROR: could not insert module *drv_src/drvr/lattice_main.ko*: Invalid module format.

Resolution: Run the following commands sequentially in the terminal.

```
sudo apt update && sudo apt upgrade
sudo apt remove --purge linux-headers-*
sudo apt autoremove && sudo apt autoclean
sudo apt install linux-headers-generic
sudo apt-get install build-essential linux-headers-`uname -r`
```

7.3.2. Problem in Driver Loading

After launching the user interface, if the driver is not loaded properly, a Driver open[Failed] message is displayed in the Info section.

- Make sure that the board is properly connected to the PC.
- Run the command below in a terminal window to show the list of PCIe devices connected to PC.

```
sudo lspci -vnm
```

Ensure that the Lattice device is present by checking the device and vendor IDs, as shown in [Figure 7.4](#).

```
Device: 09:00.0
Class: 0702
Vendor: 1204
Device: 9c40
SVendor:      19aa
SDevice:      e004
Rev:         10

Device: 09:00.1
Class: 0702
Vendor: 1204
Device: 9c42
SVendor:      19aa
SDevice:      e004
Rev:         10
```

Figure 7.4. lspci -vnm for CertusPro-NX Output Image

- If the device is present, perform the manual setup and installation steps one by one.
- If the driver does not build properly, check for any software or kernel dependencies.

7.3.3. Problem with User Interface Launching

Go to the *Demonstration/Linux* directory. Check the content of this directory. The default content is shown in [Figure 7.5](#). Check the permissions of these files.

```
lattice@lattice:~/mul6/CL-NX_BridgeBoard_PCIe_Multifunction/Demonstration/Linux$ ls -l
total 16
-rw-r--r-- 1 root root 175 Feb 16 11:53 install.sh
-rw-r--r-- 1 root root 237 Feb 16 11:53 Readme.txt
drwxr-xr-x 6 root root 4096 Feb 16 11:53 Source_Code
-rw-r--r-- 1 root root 47 Feb 16 11:53 uninstall.sh
```

Figure 7.5. Content List of Demonstration/Linux Directory

- The script files should have *execute* permission – to set permissions on the file, run the command below in the terminal and try to run the application again.

```
sudo chmod 777 filename
```

- If the user interface still does not launch, change directory into the Source_Code directory and verify that all files are present in this directory, according to [Figure 7.6](#).

```
lattice@lattice:~/mul6/CL-NX_BridgeBoard_PCIe_Multifunction/Demonstration/Linux/Source_Code$ ls -l
total 36
drwxr-xr-x 5 root root 4096 Feb 16 11:53 app_src
-rw-r--r-- 1 root root 27 Feb 16 11:53 compile.sh
drwxr-xr-x 5 root root 4096 Feb 16 11:53 drv_src
drwxr-xr-x 2 root root 4096 Feb 16 11:53 include
-rw-r--r-- 1 root root 137 Feb 16 11:53 install_drv.sh
-rw-r--r-- 1 root root 52 Feb 16 11:53 launch_gui.sh
drwxr-xr-x 2 root root 4096 Feb 16 11:53 lib
-rw-r--r-- 1 root root 566 Feb 16 11:53 Makefile
-rw-r--r-- 1 root root 156 Feb 16 11:53 Readme.txt
```

Figure 7.6. Content List of Software/Linux Directory

- If all files are present in this directory, change directory to `app_src/gui/deploy` and try to run the command below directly.

```
sudo ./MFDemo.sh
```

References

For more information, refer to:

- [CertusPro-NX Versa Board User Guide \(FPGA-EB-02053\)](#)
- [CertusPro-NX FPGA web page](#)
- [FPGA Design Software web page](#)
- [Lattice Radiant Software FPGA web page](#)
- [Lattice Insights](#) for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.0, August 2023

Section	Change Summary
All	Initial release.



www.latticesemi.com