



Automate Stack 2.0 Demo

User Guide

FPGA-UG-02164-1.0

June 2022

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

1. Introduction	9
2. Hardware Requirements	10
3. Software Requirements	11
4. Test Setup	12
4.1. System Block Diagram	12
4.2. Hardware Setup - Windows Machine	13
4.2.1. Hardware Connection	13
4.3. Hardware Setup - Linux Machine	14
4.3.1. Hardware Connection	14
4.4. Automate Stack Demonstration	15
4.4.1. Downloadable Demo Files	15
5. Hardware System Readiness	16
6. Steps to run the server (Server End)	17
7. Running the Motor through Test Application Software (Client Side)	18
7.1. Start the Application in Windows PC	18
7.2. Connect to the server	19
7.3. Select the Chain	20
7.4. Motor Configurations	21
7.5. Set the Target RPM from the Dashboard page.	23
7.6. Motor Status	24
7.7. Forward/Reverse rotation	25
7.8. Capture PDM data	27
7.8.1. Collect PDM Data	27
7.8.2. Batch Mode	29
7.9. Node Peripherals	31
7.9.1. I2C	31
7.9.2. SPI	33
7.9.3. Modbus	35
8. Running the Motor through the Script (PCIe connection)	36
8.1. The following steps of Readme.txt:	36
8.2. The following steps to run the motor through the commands.	36
Appendix A. Raspberry pi Setup	44
A.1 SD card Format	44
A.2 Raspberry Pi Imager Setup	45
A.3 Start up the Raspberry Pi (Server End)	47
Appendix B. Compile the complete code	52
B.1 Make file (OPCUA)	52
B.2 To run the server (OPCUA)	52
B.3 Make file (Mqtt)	52
B.4 To run the server (Mqtt)	53
Appendix C. Application Installation (Client End)	54
Appendix D. Automate Stack 2.0 Bitfile and Binary Generation	57
D.1 Steps for bit file generation	57
MAIN SYSTEM	57
NODE SYSTEM	64
D.2 Steps for binary generation	70
Appendix E. Programming the Automate Stack on Respective FLASH	74
E.1 Main System	74
E.2 Node	79
Appendix F. Troubleshooting (Main System Board)	85

Appendix G. Debugging Using Dock light.....	90
G.1 UART Command's Description	92
G.2 Commands.....	93
1. Main system commands - to enable different mode	94
2. MOTOR Config Command	94
Motor Status Command	95
PDM Data Fetched Command	95
Appendix H. OpcUA Modeler (Server End)	98
H.1 OpcUA Modeler Installation	98
H.2 OPCUA_Modeler : For creating a new Xml (Information model)	98
Appendix I. CSV (Comma separated value) File	104
Appendix J. Setting up the Auto-Bootable MQTT-Based Client	105
Technical Support Assistance	108
Revision History	109

Figures

Figure 4.1: Block diagram	12
Figure 4.2: System setup.....	13
Figure 4.3: PCIe system setup	14
Figure 6.1: server	17
Figure 6.2: running server.....	17
Figure 7.1: Login Page	18
Figure 7.2: Dashboard Page	19
Figure 7.3: System Configuration page	19
Figure 7.4: IP Address bar	20
Figure 7.5: select Chain.....	20
Figure 7.6: Both Chain.....	20
Figure 7.7: Select Chain.....	21
Figure 7.8: Single Chain.....	21
Figure 7.9: Motor Configuration page	22
Figure 7.10: Confirmation Pop up.....	22
Figure 7.11: Authentication page	22
Figure 7.12: Update Configuration	23
Figure 7.13: Set Target RPM	23
Figure 7.14: RPM Lock Achieved Status.....	24
Figure 7.15: Motor status page.....	24
Figure 7.16: Target RPM	25
Figure 7.17: Reverse Rotation Selection	26
Figure 7.18: Start Update.....	26
Figure 7.19: Motor Power OFF	27
Figure 7.20: Analyzing PDM Data.....	28
Figure 7.21: Collecting PDM Data	28
Figure 7.22: Collect PDM data	29
Figure 7.23: Zoom In and Zoom Out	29
Figure 7.24: Number of PDM file Name.....	30
Figure 7.25: Start Batch Mode	30
Figure 7.26: Collecting Multiple images.....	31
Figure 7.27: 12C	31
Figure 7.28: Connect option	32
Figure 7.29: Transaction log.....	32
Figure 7.30: I2C control.....	33
Figure 7.31: SPI	33
Figure 7.32: Connect option	34
Figure 7.33: Transaction log.....	34
Figure 7.34: SPI Control	34
Figure 7.35: Modbus.....	35
Figure 8.1: Check Device.....	36
Figure 8.2: run the script.....	37
Figure 8.3: initializing the object.....	38
Figure 8.4: Motor Configuration	39
Figure 8.5: Start motor Node id and target RPM	40
Figure 8.6: Stop motor node id	40
Figure 8.7: Power off node id	41
Figure 8.8: Power off node id prints	42
Figure 8.9: read a register.....	42
Figure 8.10: Disconnect and exit.....	43

Figure 8.11: SD card installation	44
Figure 8.12: SD card Formatter	45
Figure 8.13: Imager Installation	45
Figure 8.14: OS selection	46
Figure 8.15: SD card selection	46
Figure 8.16: Write the SD card	47
Figure 8.17: SD Card write successful	47
Figure 8.18: Raspberry pi setup image	48
Figure 8.19: Raspberry pi OS desktop	48
Figure 8.20: Initial step	49
Figure 8.21: Country, Language and Timezone selection	49
Figure 8.22: change password	50
Figure 8.23: update software	50
Figure 8.24: setup complete	51
Figure 8.25: Compilation	52
Figure 8.26: Make clean	52
Figure 8.27: run server	52
Figure 8.28: Compilation	53
Figure 8.29: Make clean	53
Figure 8.30: run server	53
Figure 8.31: Installer directory	54
Figure 8.32: Initial Install step	54
Figure 8.33: Next step	55
Figure 8.34: Click on Install	55
Figure 8.35: wait for the next	56
Figure 8.36: installation finish	56
Figure 8.37: Lattice Radiant Device Selector for Main System	57
Figure 8.38: Strategy for Build Generation for Main System	58
Figure 8.39: MAP analysis setting for Main system bitfile generation	59
Figure 8.40: PAR setting for Main system bitfile generation	60
Figure 8.41: PAR Timing analysis setting for Main system bitfile generation	61
Figure 8.42: Device Constraint Selection for Main System	61
Figure 8.43: Global Constraints for Main System	62
Figure 8.44. Run All button	62
Figure 8.45. System Initialization File	63
Figure 8.46. ISR RAM Initialization File	63
Figure 8.47. Validate Button	64
Figure 8.48. Generate SGE Button	64
Figure 8.49. Radiant Tool Button	64
Figure 8.50. Run All Button	64
Figure 8.51: Lattice Radiant Device Selector for Node System	65
Figure 8.52: Strategy for Build Generation for Node System	66
Figure 8.53: MAP analysis setting for Node system bitfile generation	66
Figure 8.54: PAR setting for Node system bitfile generation	67
Figure 8.55: PAR Timing analysis setting for Node system bitfile generation	67
Figure 8.56: Device Constraint Selection for Node System	68
Figure 8.57: Global Constraints for Node System	68
Figure 8.58. Run All	69
Figure 8.59. System0 Initialization	69
Figure 8.60. Validate Button	69
Figure 8.61. Generate SGE Button	70
Figure 8.62. Radiant Tool Button	70

Figure 8.63: Run All Button	70
Figure 8.64: Select Directory.....	70
Figure 8.65: Import Project.....	71
Figure 8.66: Existing Project into Workspace	71
Figure 8.67: Select project	72
Figure 8.68: Build Configurations.....	72
Figure 8.69: Console	72
Figure 8.70: Build All	73
Figure 8.71: Completing Process	73
Figure 8.72: Radiant Programmer Default Screen	74
Figure 8.73: Radiant Programmer- Initial Project Window.....	75
Figure 8.74: Radiant Programmer- Device Selection	75
Figure 8.75: Radiant Programmer- Device Operation	76
Figure 8.76: Erase all (Radiant Programmer).....	77
Figure 8.77: Radiant Programmer- Bitstream Flashing Settings	78
Figure 8.78: Radiant Programmer- Binary Flashing Settings	79
Figure 8.79: Radiant Programmer Default Screen	80
Figure 8.80: Radiant Programmer- Initial Project Window.....	80
Figure 8.81: Right-click and select Device Properties.	81
Figure 8.82: Radiant Programmer- Device Operation	81
Figure 8.83: Erase all (Radiant Programmer).....	82
Figure 8.84: Radiant Programmer- Bit stream Flashing Settings	83
Figure 8.85: Radiant Programmer- Binary Flashing Settings	84
Figure 8.86: Reset Button SW3 of CertusPro NX Versa Board.....	84
Figure 8.87: Reset Button SW3 of Certus NX Versa Board	84
Figure 8.88: Radiant Programmer Default Screen	85
Figure 8.89: Radiant Programmer- Initial Project Window.....	85
Figure 8.90: Radiant Programmer- Device Selection	86
Figure 8.91: Radiant Programmer- Device Operation	86
Figure 8.92: Quad Mode Programming (Radiant Programmer)	87
Figure 8.93: Bitfile Programming	88
Figure 8.94: Erase all (Radiant Programmer).....	89
Figure 8.95: Setup for docklight.....	90
Figure 8.96: Docklight Default Screen.....	91
Figure 8.97: Select Com port and Baud rate	91
Figure 8.98: Run button	92
Figure 8.99: Reset the main board(CertusPro-NX Versa)	92
Figure 8.100: System Initialization	92
Figure 8.101: OPCUA Modeler	98
Figure 8.102: Create OpcUA Modeler xml file	99
Figure 8.103: Create Main system	99
Figure 8.104: Add namespacearray in Browse	100
Figure 8.105: Object name.....	100
Figure 8.106: manual selection of node id	101
Figure 8.107: Add Variable.....	101
Figure 8.108: Define Variable	102
Figure 8.109: Add Method	102
Figure 8.110: opcua modeler.....	103
Figure 8.111: xml script.....	103
Figure 8.112: CSV file	104
Figure 8.113: IPV4 Address Setting.....	106
Figure 8.114: Network Preferences Settings	106

Figure 8.115. IP Configuration107

1. Introduction

This document describes the process for running the basic Lattice Automate Stack 2.0 Demo. It supports MQTT broker/client based host application, Python Interface as host control and It will also support PCIe interface as host for high speed applications. It support two chains of nodes which can be connected to one main system board and nodes will be synchronized if number of nodes are same in the both chains. and main board connected to the Raspberry pi.

This basic demo will include:

1. Raspberry pi: Communication between Server and Client end.
2. Two systems: Main System and Node System
3. Main System: for Host Communication, Host to Node communication and PDM.
4. Node System: for motor controlling and PDM data Collection.

2. Hardware Requirements

This demonstration requires the following hardware components:

- PC running Windows 10 Operating System of 1920X1080 resolution, 100% dpi
- Raspberry Pi 4 Model B with microSD card.
- USB Type-C cable with 5V adapter for Power connection in Raspberry Pi.
- Micro USB to HDMI cable for Raspberry pi to Monitor Connection.
- Lattice Certus-NX Versa Evaluation Boards with 12 V power adapters(4 Nodes or more)
- Lattice CertusPro-NX Versa Evaluation Boards with 12 V power adapters(1 Main)
- PCIe Cable for Host PC to CertusPro NX connection
- USB Type-A (UART) cable for programming the bit stream and binary files.
- Ethernet Cables for connecting Main system to Node systems
- Two electrical 1G SFPs (Methode, Finisar etc) for Main System Board (Insert at J15 and J16 ports of CertusPro-NX Versa Board)
- GB-42 BLS 24V 5000 RPM 4 or more motors with Trenz TEP0002 4 or more motor control boards
- 24V-0.5Amp DC Power Supplies for motors
- Monitor with HDMI support.
- Keyboard and Mouse, if required.

3. Software Requirements

The following software programs are available at www.latticesemi.com/en/Products/DesignSoftwareAndIP

The software programs are available for download only if you log in at www.latticesemi.com

- Lattice Radiant 3.0
- Lattice Propel SDK 2.0 or later
- Lattice Propel Builder 2.0 or later
- Lattice Radiant Programmer 3.1 or later
- Total Phase Control center 4.1
- Lattice Automate 2.0 Test Application Software

4. Test Setup

4.1. System Block Diagram

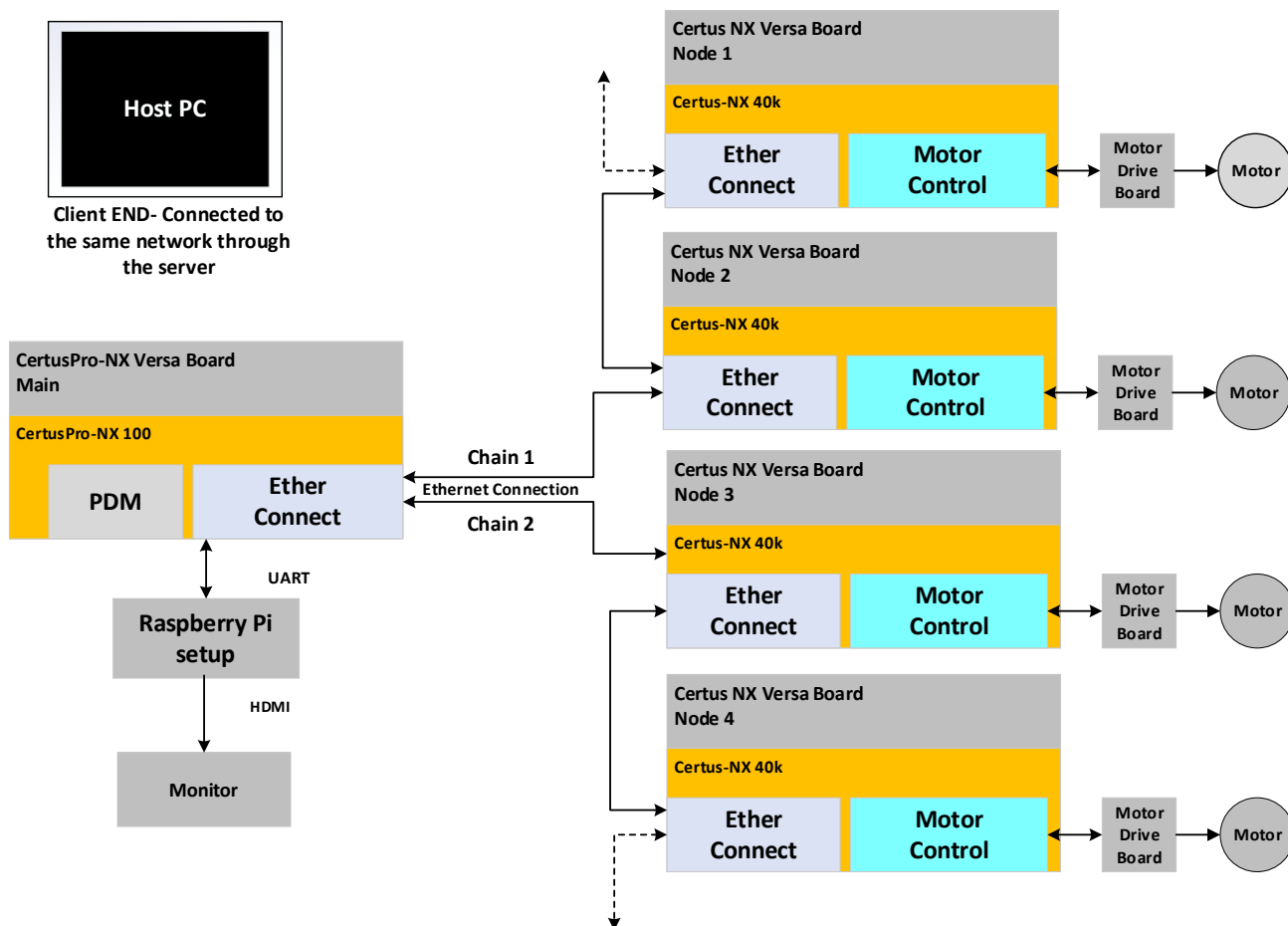


Figure 4.1: Block diagram

4.2. Hardware Setup - Windows Machine

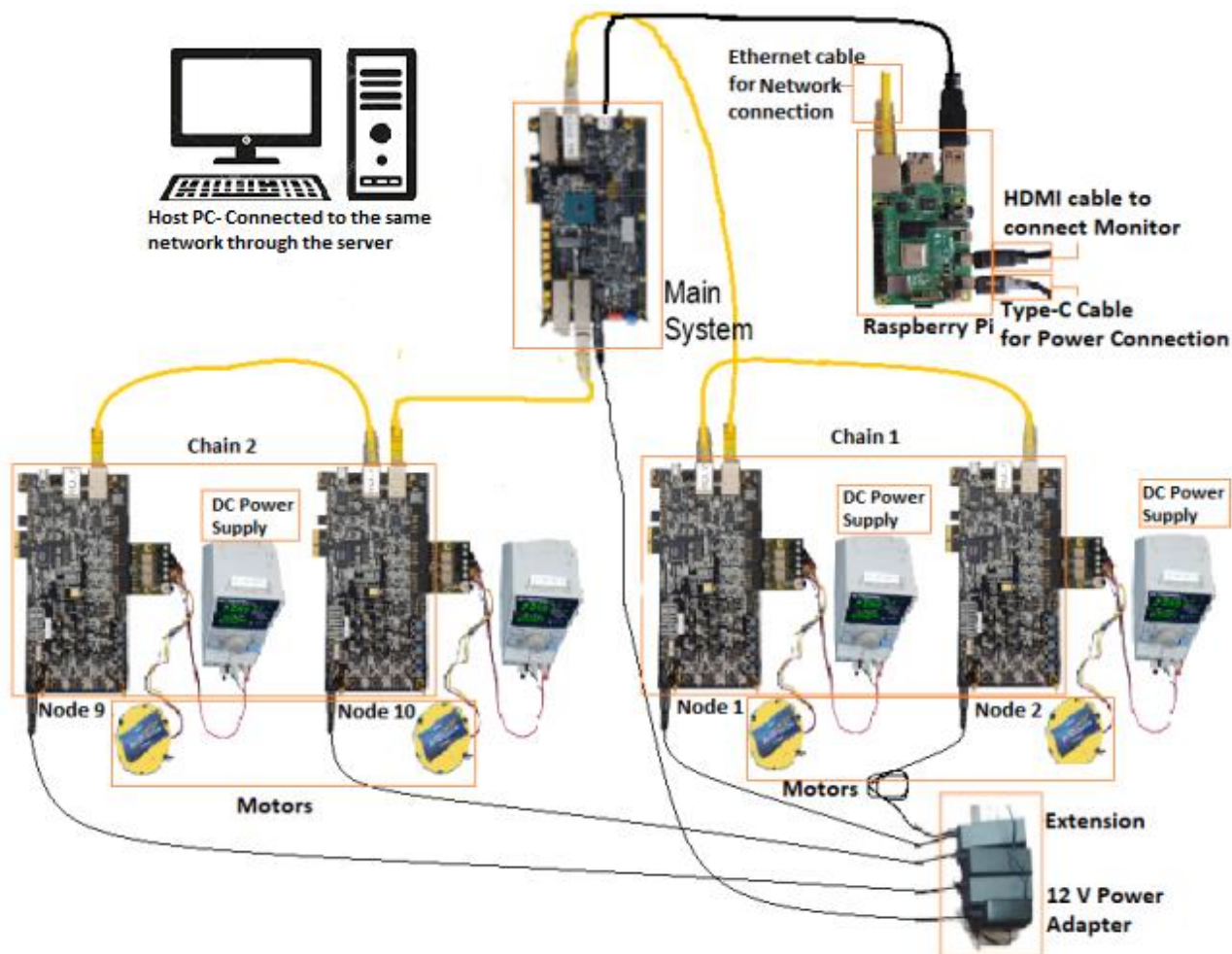


Figure 4.2: System setup

4.2.1. Hardware Connection

1. Host PC (Client End) to run the Lattice Automate 2.0 Application
2. Raspberry pi 4 Model B with micro-SD card.
3. USB Type-C cable with 5V adapter for Power connection in Raspberry Pi.
4. Lattice Certus-NX Versa Evaluation Boards with 12 V power adapters.
5. A Lattice CertusPro-NX Versa Evaluation Board with 12 V power adapter.
6. USB Type-A (UART) cable for Connect the Raspberry pi to Lattice CertusPro NX and CertusNX board.
7. Ethernet cables and two electrical 1G SFPs (Methode, Finisar etc) for main to Node Connection
8. Ethernet or Wi-Fi for Network connection at the client and server end.
9. DC power Supply with 24 V and 0.5 A current of minimum requirement.
10. GB-42 BLS 24V 5000 RPM 4 or more motors with Trenz TEP00-3 4 or more motor control boards.
11. Micro USB to HDMI cable for Raspberry pi to Monitor Connection.

4.3. Hardware Setup - Linux Machine

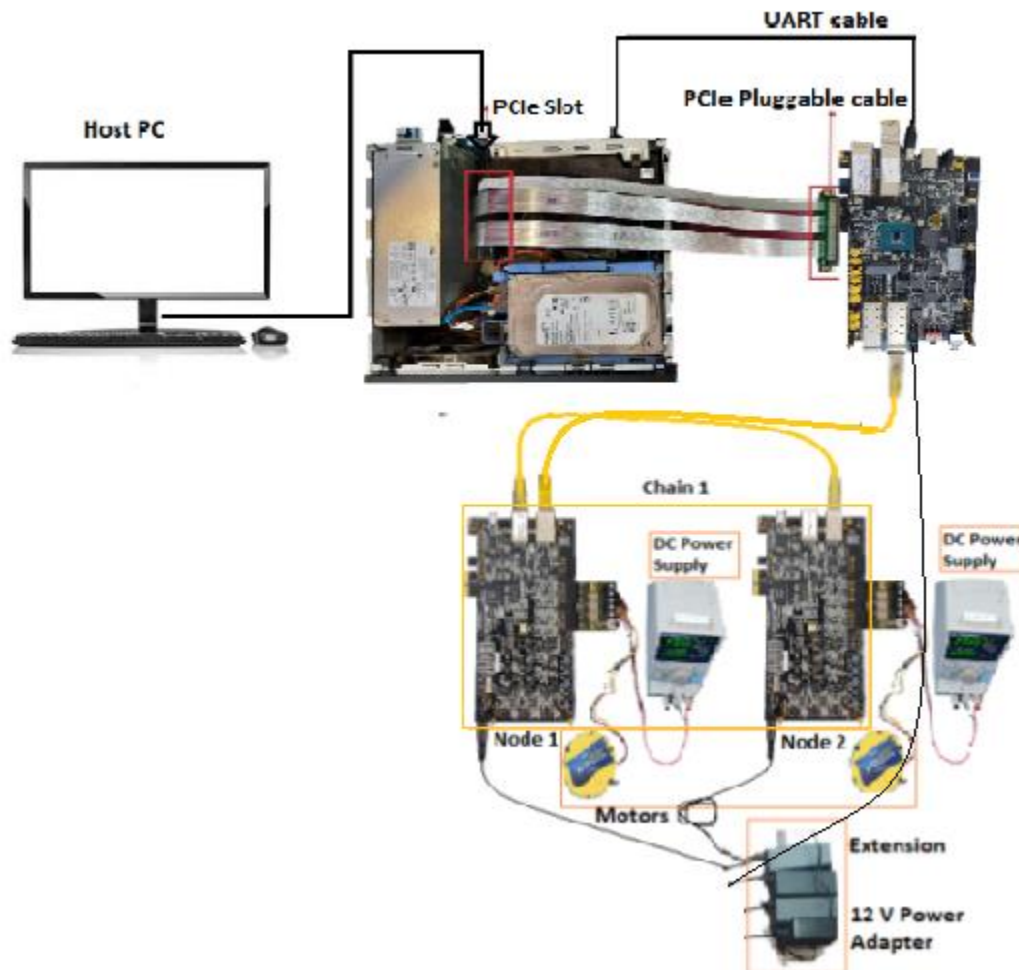


Figure 4.3: PCIe system setup

4.3.1. Hardware Connection

1. Host PC to run the Lattice Automate script.
2. PCIe Cable for Host PC to CertusPro NX connection
3. Lattice Certus-NX Versa Evaluation Boards with 12 V power adapters.
4. Lattice CertusPro-NX Versa Evaluation Board with 12 V power adapter.
5. USB Type-A (UART) cable for Connect the Raspberry pi to Lattice CertusPro NX and CertusNX board.
6. Ethernet cables and two electrical 1G SFPs (Methode, Finisar etc) for main to Node Connection
7. DC power Supply with 24 V and 0.5 A current of minimum requirement.
8. GB-42 BLS 24V 5000 RPM 4 or more motors with Trenz TEP00-3 4 or more motor control boards

Automate Stack Demo Package Directory Structure

The directory structure of the Automate Stack Demo Package is listed below.

4.4. Automate Stack Demonstration

4.4.1. Downloadable Demo Files

4.4.1.1. Documentation

- A. Executables
 - a. Main System
 - i. PDM_DataSection.mem
 - ii. PDM_ISRCodeSection.mem
 - iii. riscv-pdm.bin
 - iv. soc_main_system_impl_1.bit
 - b. Node System
 - i. NodeSystem_AS2_001.bin
 - ii. NodeSystem_AS2_001.bit
- B. Executables
 - a. Main System

4.4.1.2. Host PC

- A. GUI
- B. PciScript

4.4.1.3. Raspberrypi

- A. Mqtt_Lattice_Automate_2.0.zip
- B. Readme.txt

4.4.1.4. Script

- A. Lattice_Automate_Stack_2_0_Docklight.ptp

Below is the brief description of the main directories.

- The Automate Stack Demonstration folder will be the parent folder for all the files. It will have 3 sub folders, namely Images, Project and Documentation.
- The Images sub-folder will have the FPGA Images (bit files) and Binary Images(Firmware) for both main system and node.
- The Project sub-folder will contain the whole project package and files for both main system and node. The FPGA project can be accessed in the soc_main_system/soc_node section and firmware project can be accessed in the c_main_system/c_node section.
- The documentation sub-folder will contain the user guide for the project.

5. Hardware System Readiness

1. Hardware should be Connected properly as mentioned in [Section 4.2](#).
2. All Boards Should be programmed
3. Power cycle the each boards After programming all the boards.
4. After Power cycle done, Reset the Main Board.
5. For Raspberry Pi Connections and installation process refer to [Appendix A](#).
6. Application Installation, refer to [Appendix C](#).
7. To start the server, refer the [Section 7](#).
8. To launch the GUI/Application, refer the [Section 8](#).

6. Steps to run the server (Server End)

Note: For Raspberry pi Board bring up, refer the [Appendix A](#).

1. Open the terminal.
2. Go to the Directory
3. Lattice_Automate_Stack_1_1/OPCUA_Server/OPCUA_serverApp/bin
4. Write the command “./server” to start the server

```
pi@raspberrypi:~ $ cd Rel18May/Mqtt_Lattice_AutomateStack_2_0/
pi@raspberrypi:~/Rel18May/Mqtt_Lattice_AutomateStack_2_0 $ cd bin/
pi@raspberrypi:~/Rel18May/Mqtt_Lattice_AutomateStack_2_0/bin $ ./server -v5
```

Figure 6.1: server

5. Server will start.

```
pi@raspberrypi:~ $ cd Rel18May/Mqtt_Lattice_AutomateStack_2_0/
pi@raspberrypi:~/Rel18May/Mqtt_Lattice_AutomateStack_2_0 $ cd bin/
pi@raspberrypi:~/Rel18May/Mqtt_Lattice_AutomateStack_2_0/bin $ ./server -v5
Host : eth0
IP : 10.0.1.244

-----Lattice Automate stack 2.0 version : 0.0.1 ---May 12, 2022 -----
MainSystemVariablesRead Begin
MainSystemVariablesRead end

----- State handler -----
CC 77 0C 00 00 02 00 00 00 01 10 00
TX packet Ret: Waiting for Data. Total timeout duration = 18000 seconds
Data Captured
AA 88 10 00 00 02 00 00 00 00 01 00 00 00
RX packet Ret: 0
Response func: 2, error: 0
RX response_data: 1
setNodeValue, UpdateVariable value : 1
MainSystem_List_arr[0][ms.FrameType-64]: ac
CC 77 0C 00 00 02 00 00 00 01 10 00
TX packet Ret: Waiting for Data. Total timeout duration = 18000 seconds
Data Captured
AA 88 10 00 00 02 00 00 00 00 00 40 00 00 00
RX packet Ret: 0
Response func: 2, error: 0
RX response_data: 40
setNodeValue, UpdateVariable value : 64
MainSystem_List_arr[0][ms.NodeDataLength-64]: ad
setNodeValue, UpdateVariable value : 0
CC 77 10 00 00 01 00 00 19 70 19 00 01 00 00 00
TX packet Ret: Waiting for Data. Total timeout duration = 18000 seconds
Data Captured
AA 88 0C 00 00 01 00 00 00 00 00 00 00
RX packet Ret: 0
Response func: 1, error: 0
CC 77 0C 00 00 02 00 00 04 80 10 00
TX packet Ret: Waiting for Data. Total timeout duration = 18000 seconds
Data Captured
AA 88 10 00 00 02 00 00 00 00 00 00 00 02 00
RX packet Ret: 0
Response func: 2, error: 0
setNodeValue, UpdateVariable value : 0
----- callback - -----ms_chain1LinkupStatus: 0
setNodeValue, UpdateVariable value : 1
----- callback - -----ms_chain2LinkupStatus: 1
CC 77 0C 00 00 02 00 00 06 80 10 00
TX packet Ret: Waiting for Data. Total timeout duration = 18000 seconds
Data Captured
AA 88 10 00 00 02 00 00 00 00 00 00 01 01 00
RX packet Ret: 0
Response func: 2, error: 0
setNodeValue, UpdateVariable value : 0
----- callback - -----ms_chainActiveNode: 0
setNodeValue, UpdateVariable value : 1
```

Figure 6.2: running server

7. Running the Motor through Test Application Software (Client Side)

This section provides the procedure for running the motor through Graphical User Interface/Test Application.

Note: For Lattice Automate Stack 2.0 Application Installation, refer to [Appendix C](#).

Note: For generation the Bit and Binary file, refer to [Appendix D](#).

Note : Make sure that Main system and each nodes programmed before running the Lattice Automate Stack 2.0 Complete setup, refer to [Appendix E](#).

7.1. Start the Application in Windows PC

1. Open the Lattice Automate Stack 2.0 Application.
2. GUI will be launch.

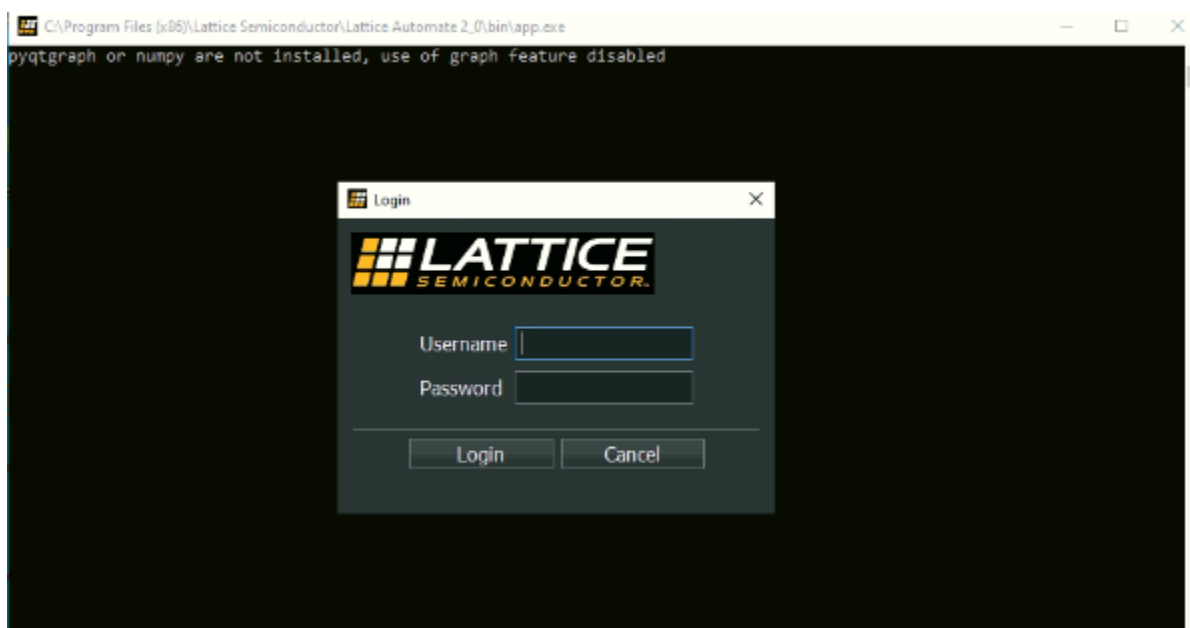


Figure 7.1: Login Page

3. Enter the Username: lattice, Password: lattice
4. Dashboard Page will open.

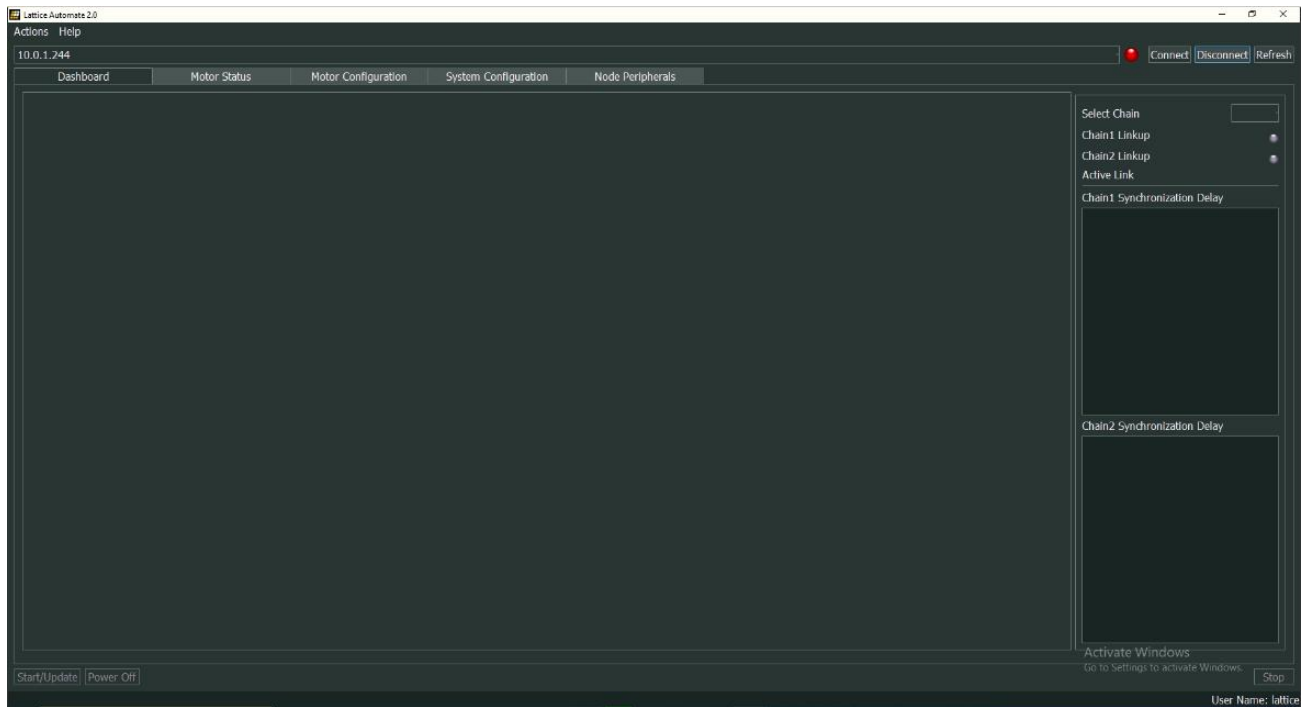


Figure 7.2: Dashboard Page

7.2. Connect to the server

1. Go on the System Configuration page.

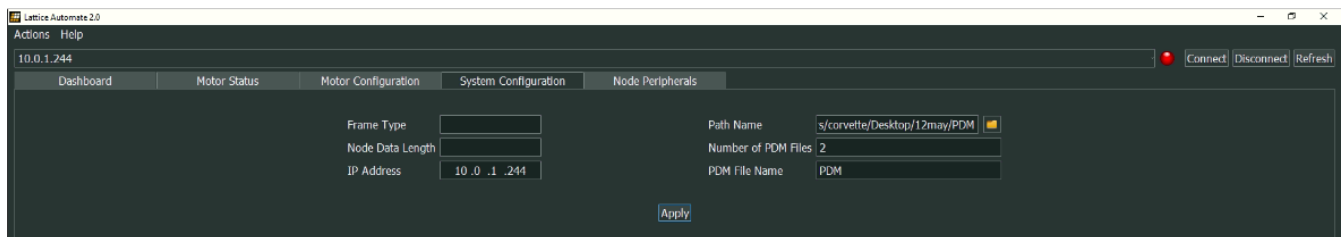


Figure 7.3: System Configuration page

2. Type the correct IP Address on the IP Address bar.
Note: IP Address should be from the raspberry pi setup.
3. Now press the **Apply** button than **Update Successfully** pop up will come.
4. Now click on the **Connect** button.
5. Once IP Address configured, it will update on the top of the Address bar.

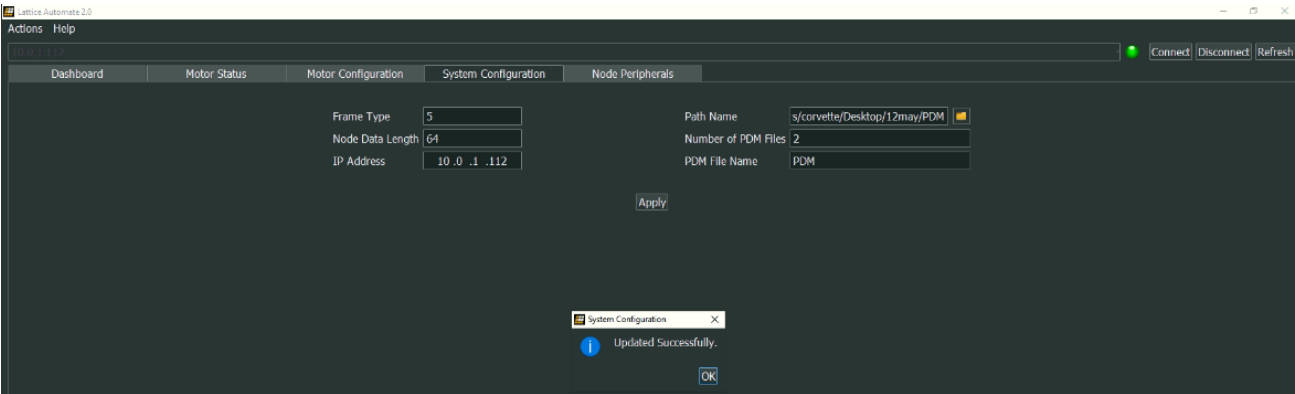


Figure 7.4: IP Address bar

7.3. Select the Chain

1. Select the **Select Chain** option as **Both**.

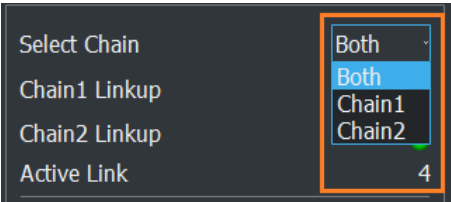


Figure 7.5: select Chain

2. It will display the both chains.

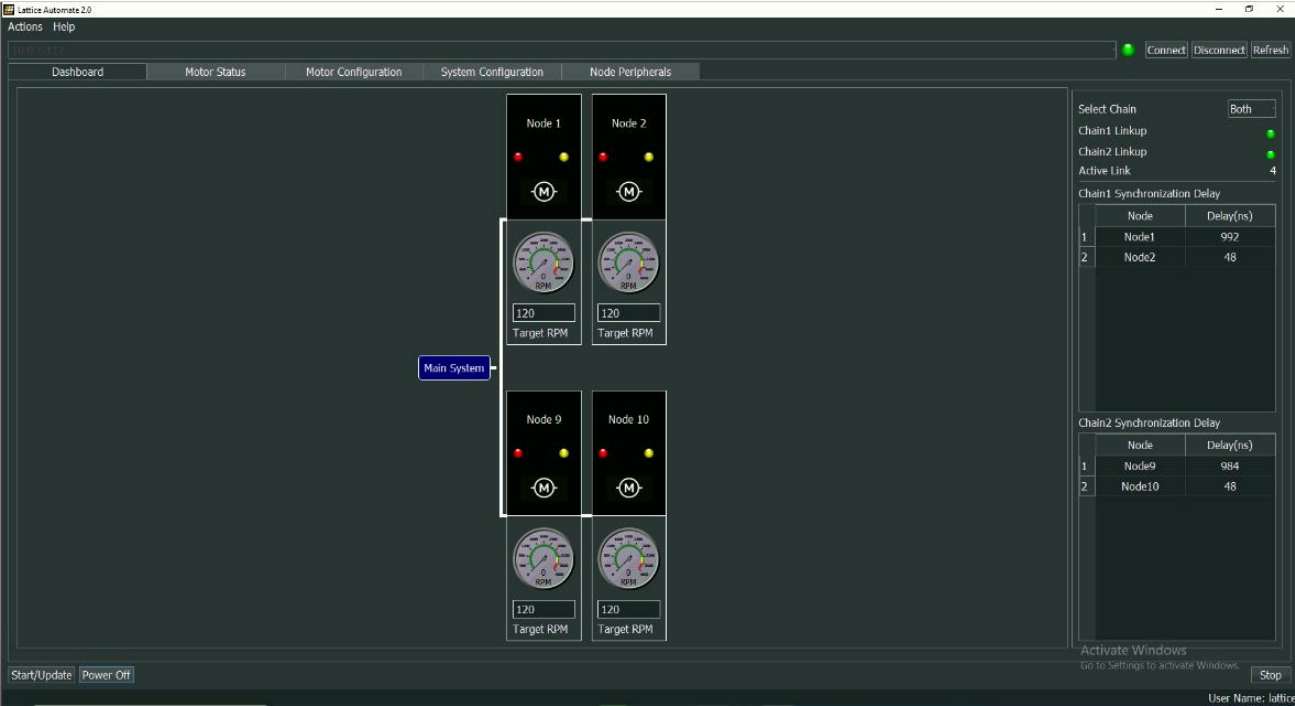


Figure 7.6: Both Chain

3. If Selected the **Select Chain** option as **Chain1** or **Chain 2**

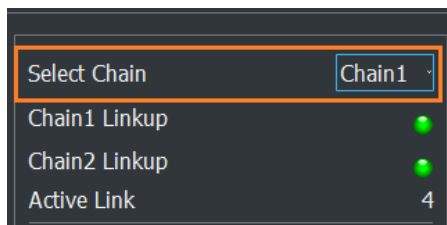


Figure 7.7: Select Chain

4. It will display the only Single Chain.

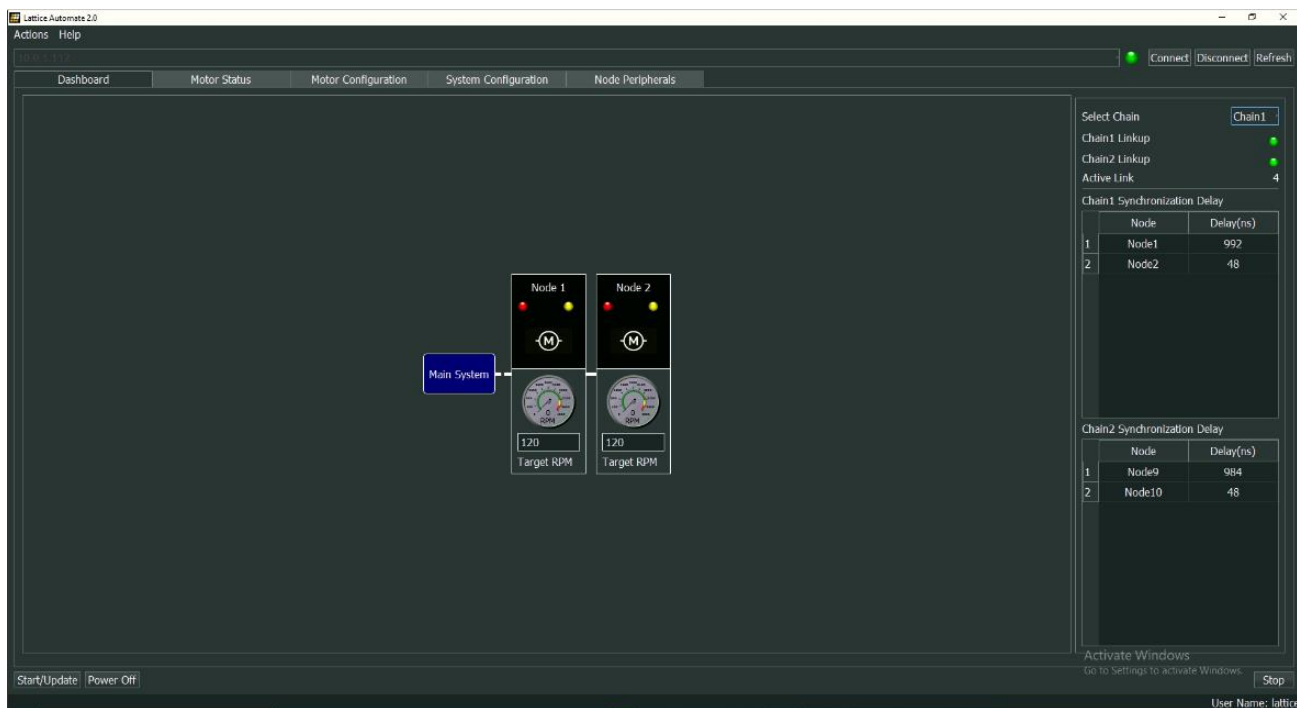


Figure 7.8: Single Chain

7.4. Motor Configurations

1. Now go on the Motor Configuration page
2. Select the number of nodes.

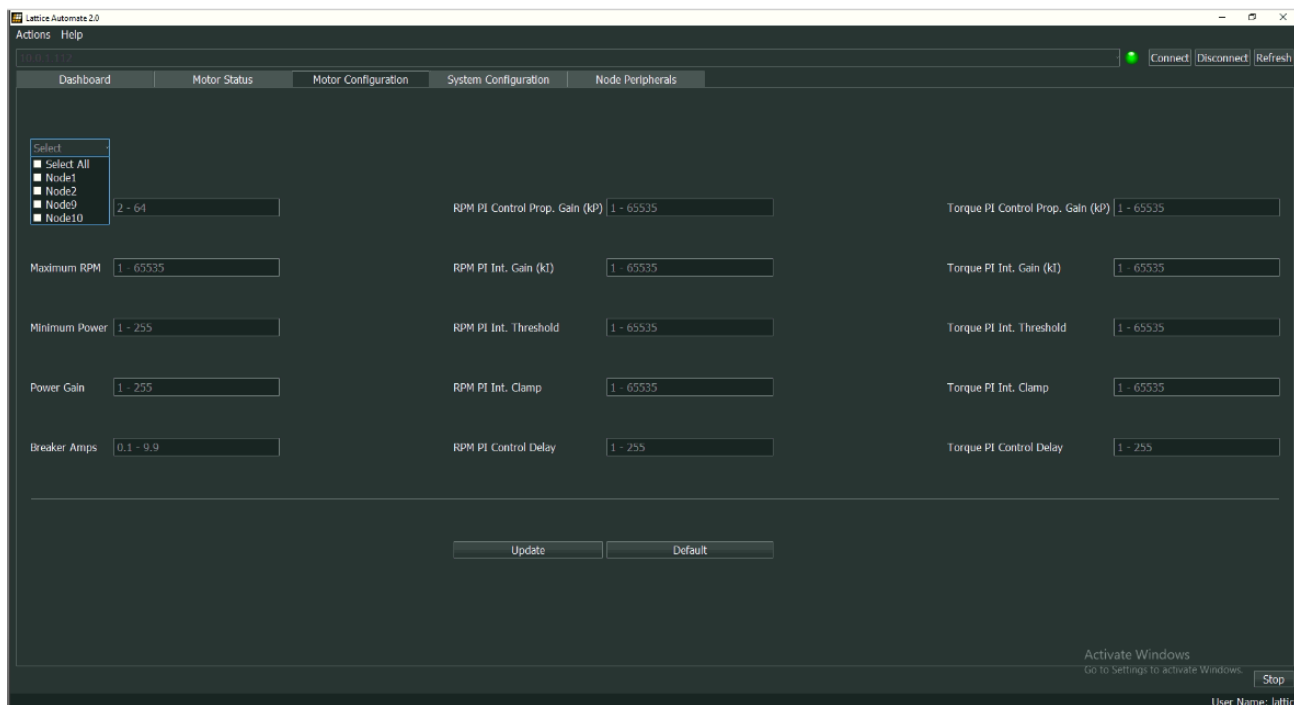


Figure 7.9: Motor Configuration page

3. Click on the default option.
4. Click on Update option.
5. Pop message will come and Confirm the update action by clicking on Yes button.

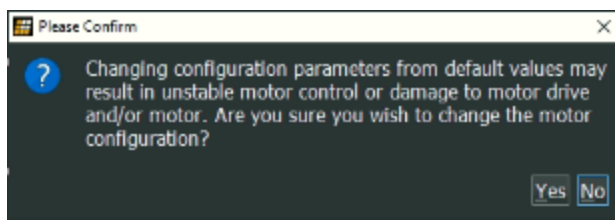


Figure 7.10: Confirmation Pop up

6. Authentication Page will come.
7. Enter the **Username: lattice** and **Password: lattice**
8. Click on the **Login** button.

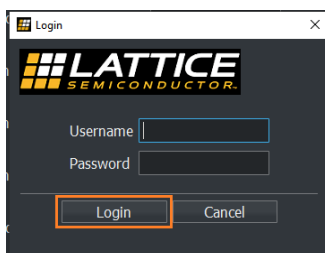


Figure 7.11: Authentication page

- Configuration **Successfully updated the configuration of the selected node** pop will come.

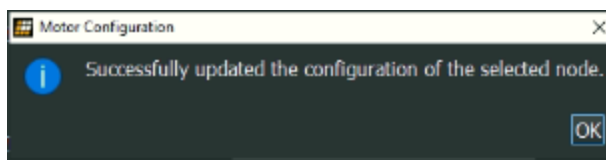


Figure 7.12: Update Configuration

7.5. Set the Target RPM from the Dashboard page.

- Go on the Dashboard Page.
- First Enter the Target RPM Value 120.

Note: Targeted RPM value should not be more than the Maximum RPM Value of Motor Configuration Page.

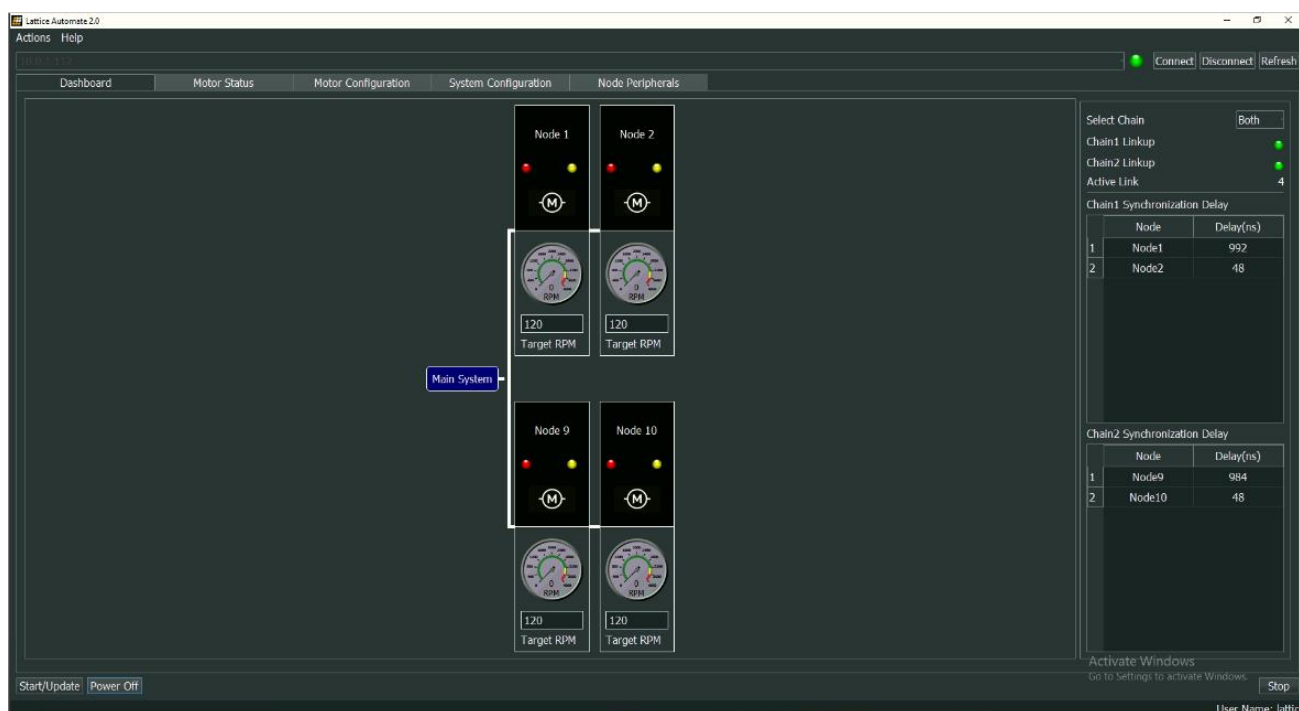


Figure 7.13: Set Target RPM

- Click on the Start Update Option.
- After the RPM Lock Achieved the Node LED glow green.

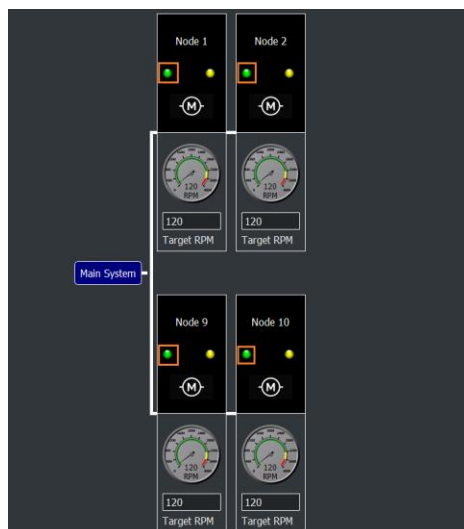


Figure 7.14: RPM Lock Achieved Status

5. Now go on the Motor status page to Check the RPM, Voltage and Current value.
6. Now Select the any Nodes as **Node 1/ Node 2/ Node 3/ Node4** or **Select all**.

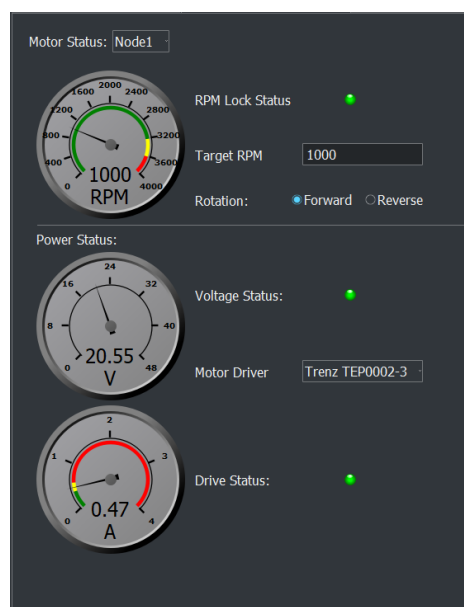


Figure 7.15: Motor status page

7. For Stopping the Motor, Click on the **Stop** from the Motor status page or dashboard.
8. In case of any Emergency or to Withdraw the current in the Motor, Click on the **Power Off** button.

7.6. Motor Status

1. Go on the Motor Status page.
2. Set the Target RPM 120
3. Now click on the **Start/Update** button.

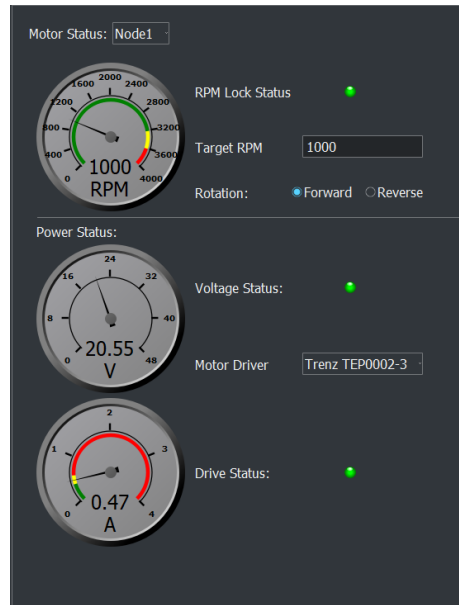


Figure 7.16: Target RPM

4. Meter gauge update to 120RPM and it will update to 120 RPM on the Target RPM bar too.
5. Set the Target RPM 1000 RPM to update the RPM Speed.
6. Meter gauge update to 1000 RPM and it will update to 1000 RPM on the Target RPM bar too.
7. For stopping the Motor, click on the **Stop** button.
8. Click on the **Power Off** button to stop the voltage and current in motor.

7.7. Forward/Reverse rotation

1. Go on the Motor Status page.
2. Select the Forward Rotation and Enter the target RPM 1000.
3. Click on **Start/Update** button.

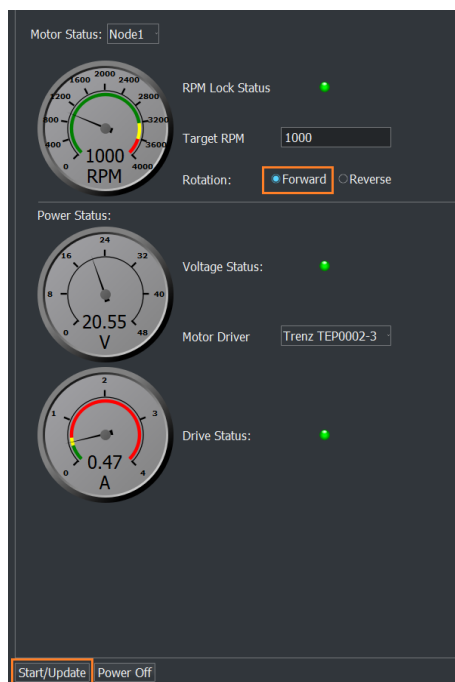


Figure 7.17: Reverse Rotation Selection

4. Motor will Start rotating in an Clockwise direction.
5. Now select the Reverse rotation.

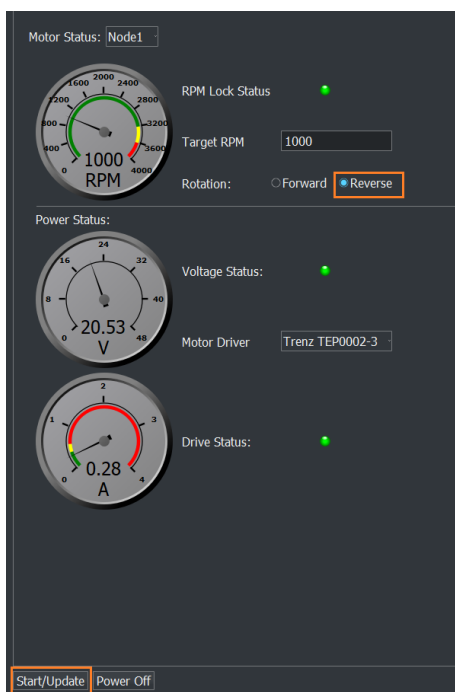


Figure 7.18: Start Update

6. Click on **Start/Update** button.
7. Motor will Start rotating in an Anti-Clockwise direction.

8. In case of Emergency, power off the motor by clicking on **Power off** button.

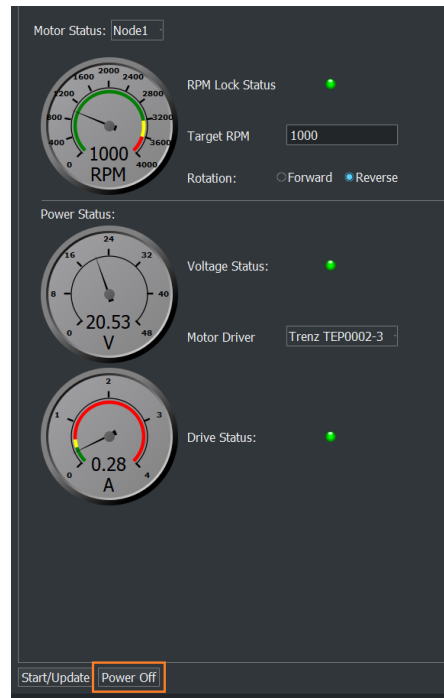


Figure 7.19: Motor Power OFF

7.8. Capture PDM data

7.8.1. Collect PDM Data

1. Go on the Motor status page.
2. Set the target RPM between 1400 and 1800.
3. Now click on the **Start/Update** button, wait until the RPM lock.
4. Now click on the **Collect PDM data**, wait for the PDM data process will complete.
5. It will show the **Analyzing PDM Data from Node** and **Collecting PDM Data from Node** while capturing the image.

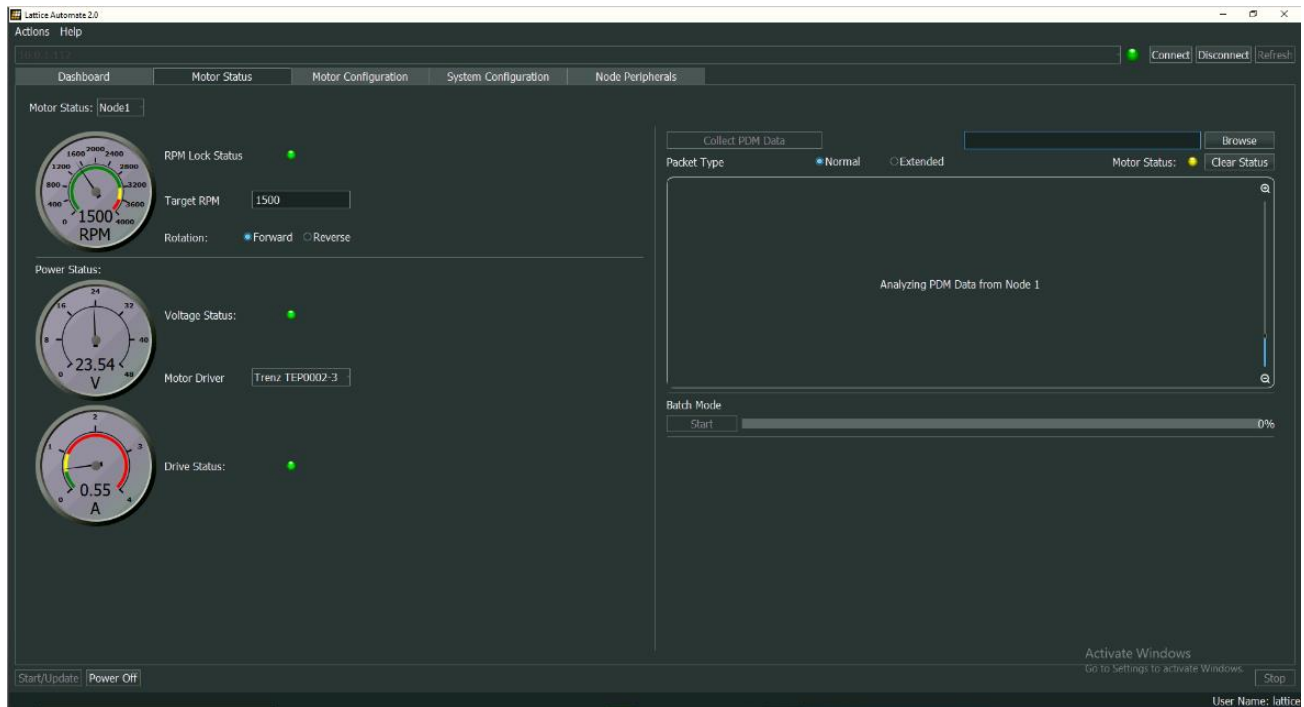


Figure 7.20: Analyzing PDM Data

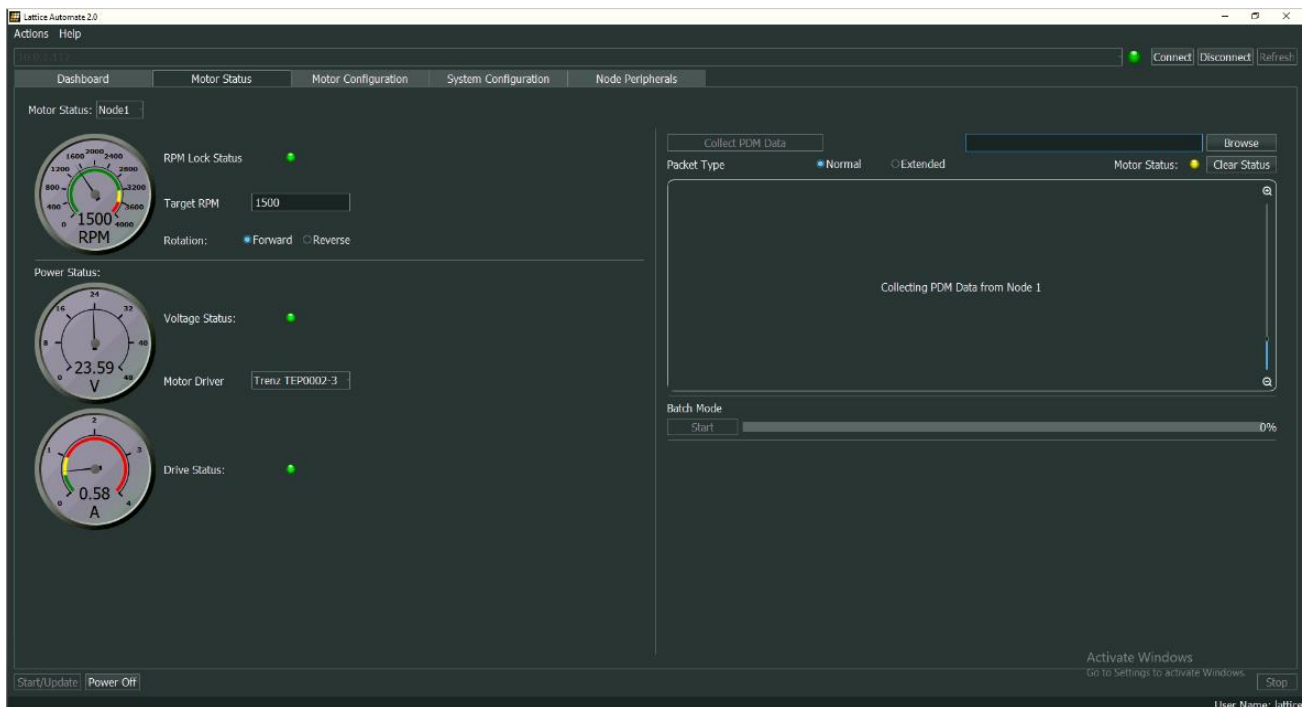


Figure 7.21: Collecting PDM Data

6. Check the PDM image, it will display on the Screen.
7. If collected data is red then click on the **Clear status** button to clear the status.
8. To fetch the previous images, click on the **Browse** button.

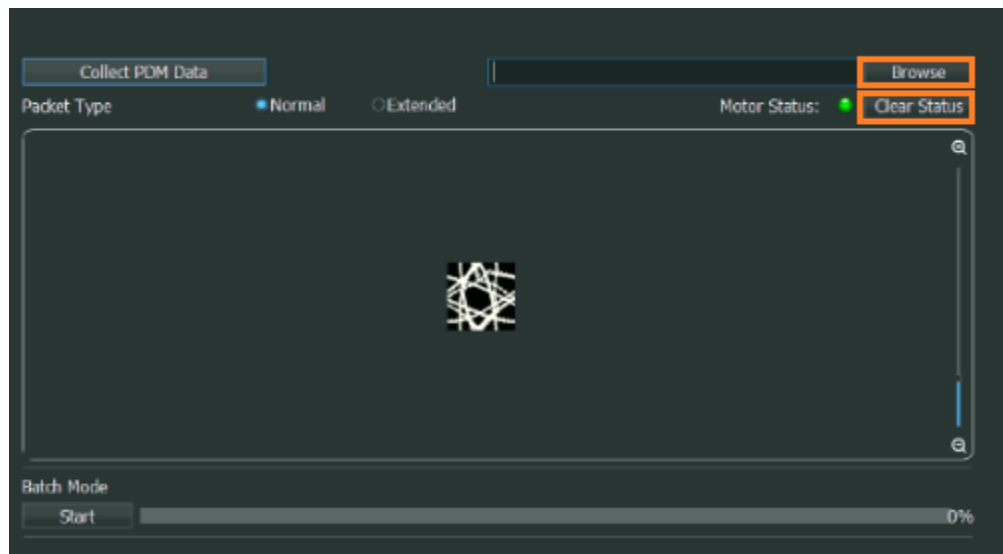


Figure 7.22: Collect PDM data

9. To Zoom-in or Zoom-out the size of PDM image, move the cursor on the blue line from the black line.

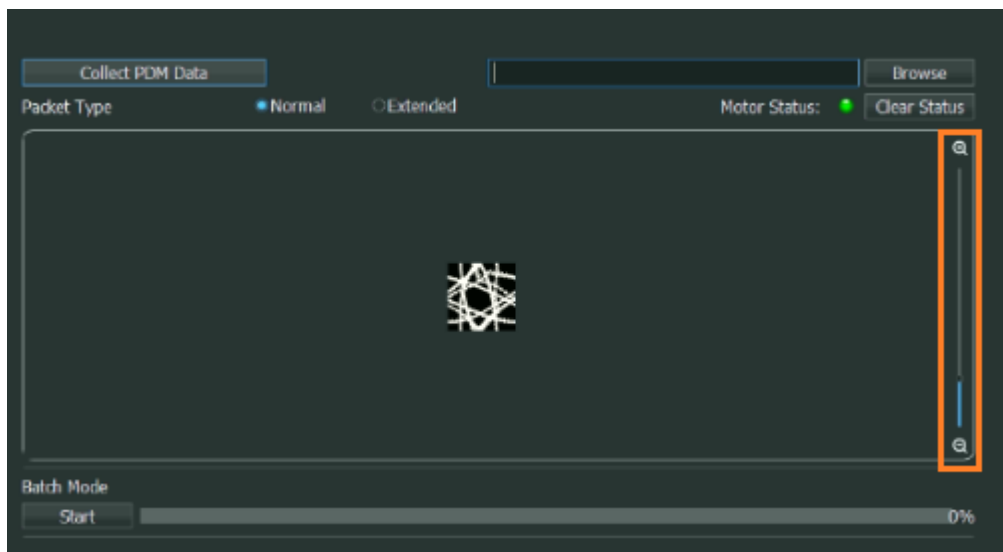


Figure 7.23: Zoom In and Zoom Out

7.8.2. Batch Mode

1. Follow the same process of Collect PDM Data process after that Batch Mode.
2. Go on the system configuration and set the Number of PDM files 5 .

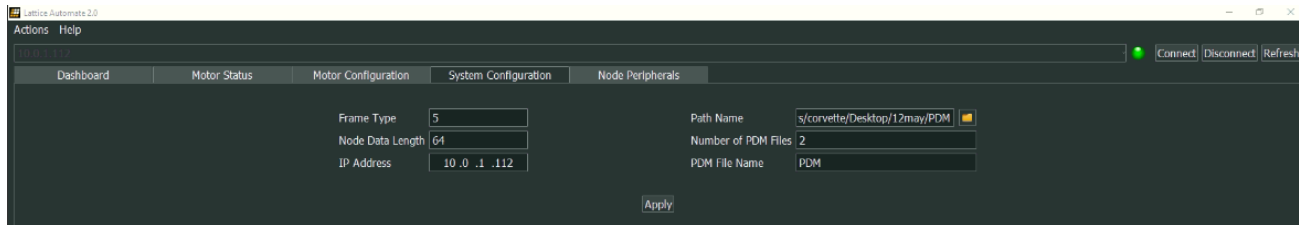


Figure 7.24: Number of PDM file Name

3. Now go back to the Motor status page.
4. For collecting the Multiple images, Click on the Batch Mode.

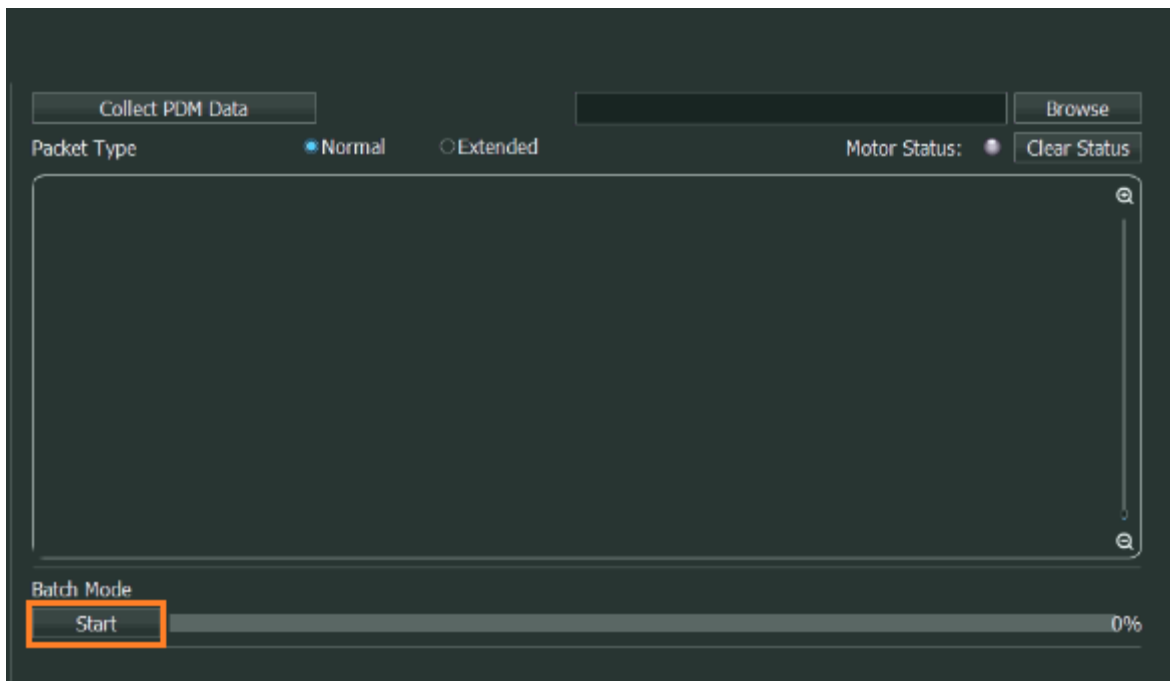


Figure 7.25: Start Batch Mode

5. Wait for the some time to collect the multiple images till 100% .

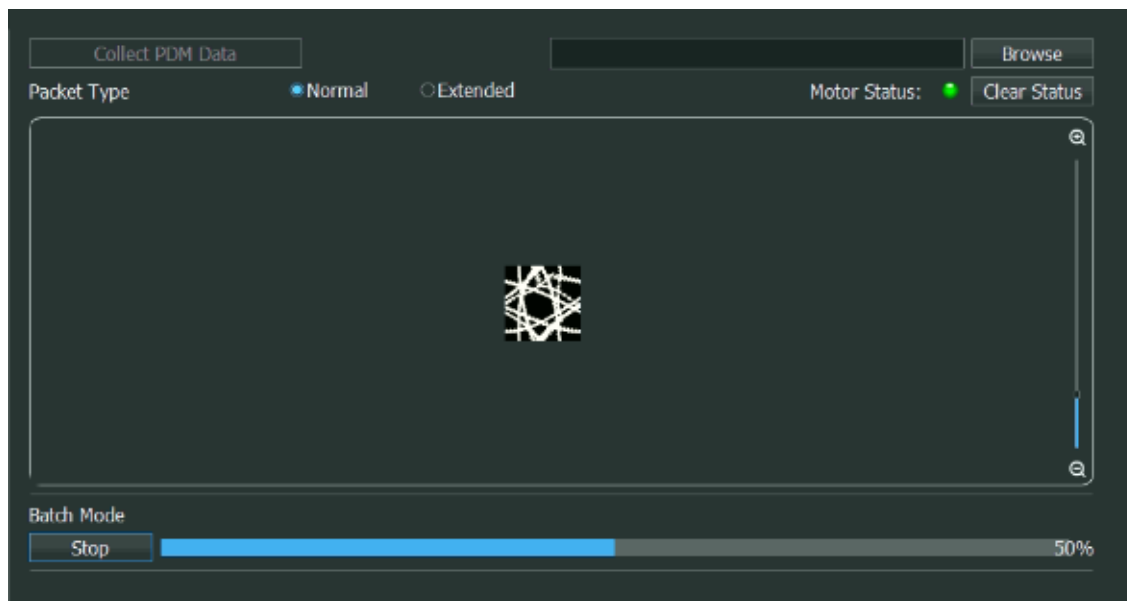


Figure 7.26: Collecting Multiple images

- After Collecting the Multiple images, Click the **Stop** button.

7.9. Node Peripherals

7.9.1. I2C

- Select any of the node.
- Select the protocol: I2C
- Select the option: Write
- Enter the slave Address one byte: for eg 4A
- Enter the Register Address one byte: for eg. 01
- Enter the Data 4 bytes: for eg. 14527
- Now Click on the write button.

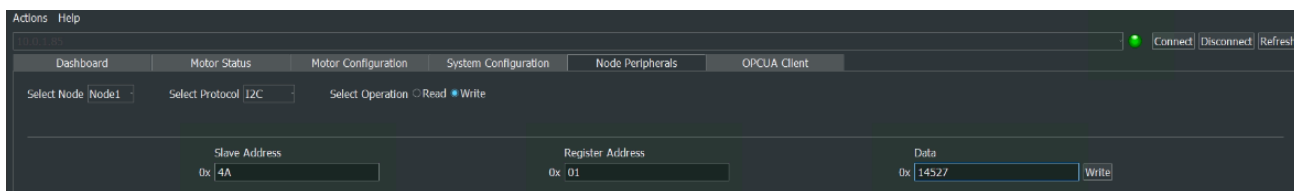


Figure 7.27: I2C

- Now open the Total Phase Control Center tool
- Go to the Adapter option
- Now go to the connect option as shown in below fig.

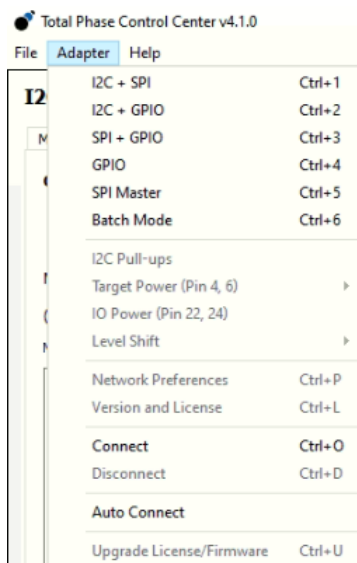


Figure 7.28: Connect option

11. Select the configuration as I2C/SPI
12. Click on the OK button
13. Click on the enable button
14. Check the transaction log

Note: Make sure that UART cable should be connected properly.

Transaction Log								
Time	Mod.	R/W	M/S	Feat.	B.R.	Addr.	Length	Data
2022-06-16 11:12:04.201	I2C	R	S	---		0X5A	2	12 54
2022-06-16 11:13:31.628	SPI	R	S	RSML			4	21 56 94 78
2022-06-16 11:13:31.628	SPI	W	S	RSML			4	00 00 00 00

Figure 7.29. Transaction log

Figure 7.30: I2C control

7.9.2. SPI

1. Select any of the node.
2. Select the protocol: SPI
3. Select the option: Write
4. Enter the slave Address one byte: for eg 9F
5. Enter the Register Address one byte: for eg. 05
6. Enter the Data 4 bytes: for eg. 25784
7. Now Click on the write button.

Figure 7.31: SPI

8. Now open the Total Phase Control Center tool
9. Go to the Adapter option
10. Now go to the connect option as shown in below fig.

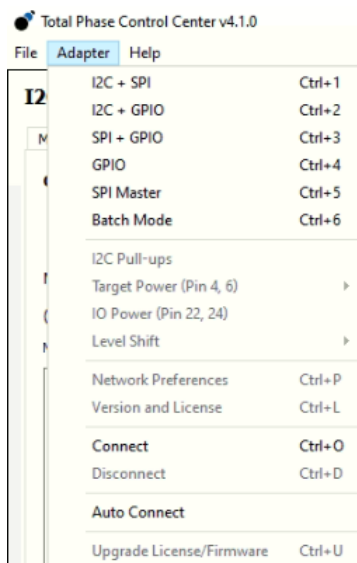


Figure 7.32: Connect option

11. Select the configuration as I2C/SPI
12. Click on the OK button
13. Click on the enable button.
14. Check the transaction log

Note: Make sure that UART cable should be connected properly

Transaction Log									
Time	Mod.	R/W	M/S	Feat.	B.R.	Addr.	Length	Data	
2022-06-16 11:12:04.201	I2C	R	S	---		0X5A	2	12 54	
2022-06-16 11:13:31.628	SPI	R	S	RSML			4	21 56 94 78	
2022-06-16 11:13:31.628	SPI	W	S	RSML			4	00 00 00 00	

Figure 7.33. Transaction log

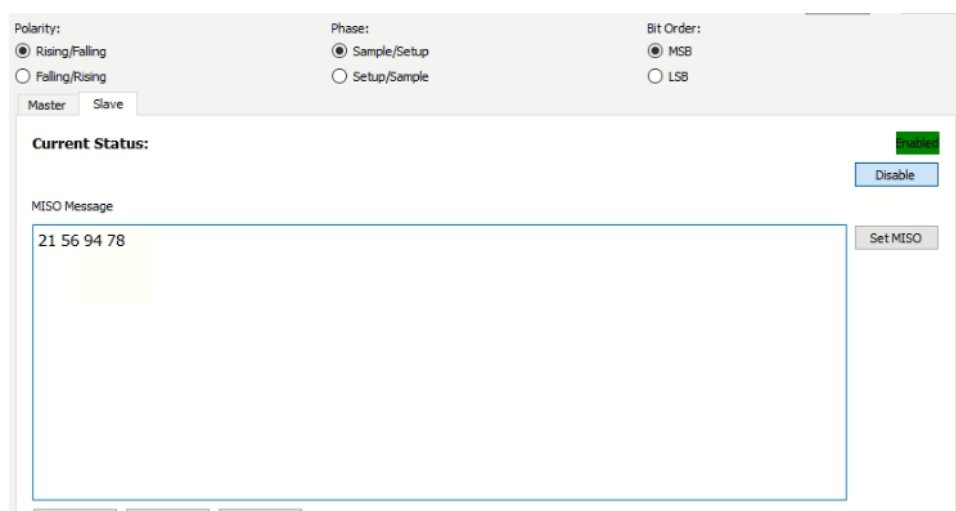


Figure 7.34: SPI Control

7.9.3. Modbus

1. For holding register, Enter the register no. : 0x6
2. Enter the Data 2 bytes: 12547

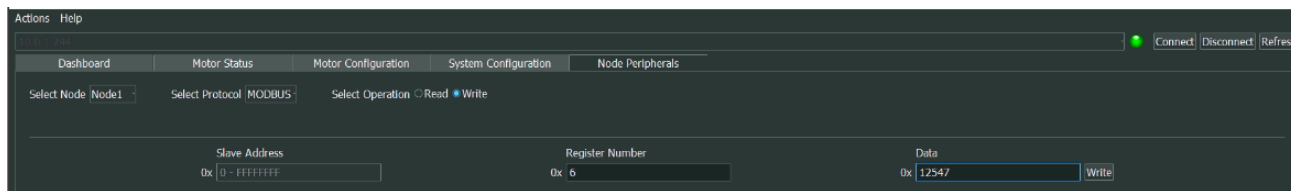


Figure 7.35: Modbus

8. Running the Motor through the Script (PCIe connection)

8.1. The following steps of Readme.txt:

1. Python packages required: pybind11, invoke
2. Python package installation commands:

```
pip3 install pybind11
```

```
pip3 install invoke
```
3. In Source_Code/wrapper/build_wrapper.py file on line 7 replace the python version if you have a different python version than 3.7
 - a. For instance if the installed python version is 3.6 then update the version from 3.7 to 3.6, as shown below
 - a. `"-o {1}`python3.6-config --extension-suffix`"`
 - b. Python version can be found by running this command:
 - a. `python3 -V`

4. Before running the demo run the following command:

```
python3 Source_Code/wrapper/build_wrapper.py
```

This command will create the python binding over c shared library "libmem_rw.so". This is a one time step

Kindly note that this step is not required when the python script is run using script.sh. The shell script is written to handle this step.

5. Execute the following commands (One time step)

```
chmod 777 script.sh console_app.sh
```

6. To run the python script run this command. This script builds the driver, library, inserts the driver, builds the python wrapper (if required) and then runs the python script.

```
sudo script.sh
```

7. To run the C++ console application run this command. This script builds the driver, library, inserts the driver and then runs the console application.

```
sudo console_app.sh
```

Check whether the board has linkup with PC , run the command: `lspci - xxx`

```
01:00.0 ATA controller: Lattice Semiconductor Corporation Device 9c1e (rev 10)
00: 04 12 1e 9c 07 00 10 00 10 20 05 01 10 00 00 00
10: 00 00 d0 f7 00 00 00 00 00 00 00 00 00 00 00 00
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 aa 19 04 e0
30: 00 00 00 00 40 00 00 00 00 00 00 00 00 0b 01 00 00
```

Figure 8.1: Check Device

8.2. The following steps to run the motor through the commands.

1. Go to the trunk directory.
2. Now run the following command: `sudo ./script.sh`

```
File Edit View Search Terminal Help
user@user-OptiPlex-3040:~$ cd Desktop/
user@user-OptiPlex-3040:Desktop$ ls
Flash-center-linux-1686-v1.45  linux-5.4          menrw  ref.c
l2c_MultiFunc                 linux_5.4.0.orig.tar.gz  PCIe  Yastlr
user@user-OptiPlex-3040:Desktop$ cd PCIe/
user@user-OptiPlex-3040:PCIe$ ls
code
user@user-OptiPlex-3040:PCIe$ cd code/
user@user-OptiPlex-3040:code$ ls
trunk
user@user-OptiPlex-3040:code$ cd trunk/
user@user-OptiPlex-3040:trunk$ ls
console_app.sh  Readme.txt  script.sh  Source_Code  uninstall.sh
user@user-OptiPlex-3040:trunk$ sudo ./script.sh
[sudo] password for user:
make clean -C lib/
make[1]: Entering directory '/home/user/Desktop/PCIe/code/trunk/Source_Code/lib'
rm -f libmem_rw.so *.o
make[1]: Leaving directory '/home/user/Desktop/PCIe/code/trunk/Source_Code/lib'
make clean -C app_src/console_app/
make[1]: Entering directory '/home/user/Desktop/PCIe/code/trunk/Source_Code/app_src/console_app'
rm -f ThroughputDemo *.o
make[1]: Leaving directory '/home/user/Desktop/PCIe/code/trunk/Source_Code/app_src/console_app'
make clean -C drv_src/lthruput_drv/
make[1]: Entering directory '/home/user/Desktop/PCIe/code/trunk/Source_Code/drv_src/lthruput_drv'
rm *.o *.ko *.mod.c
rm: cannot remove '*.o': No such file or directory
rm: cannot remove '*.ko': No such file or directory
rm: cannot remove '*.mod.c': No such file or directory
Makefile:32: recipe for target 'clean' failed
make[1]: *** [clean] Error 1
make[1]: Leaving directory '/home/user/Desktop/PCIe/code/trunk/Source_Code/drv_src/lthruput_drv'
Makefile:16: recipe for target 'clean' failed
make: *** [clean] Error 2
make -C lib/
make[1]: Entering directory '/home/user/Desktop/PCIe/code/trunk/Source_Code/lib'
g++ -I../include -std=c++11 -fPIC -Wall -Wextra -O2 -g -Wno-write-strings -DBUILD_LIB -DLOG_IN_FILE -DDMA_TEST -shared -o libmem_rw.so mem_rw.cpp
mem_rw.cpp: In member function 'uint8_t MemRW::ReadFPGAREg_64(uint32_t, uint64_t)':
mem_rw.cpp:1320:67: warning: format '%lx' expects argument of type 'long unsigned int', but argument 3 has type 'unsigned int' [-Wformat=]
    printf("%s: reg:0x%lx val:0x%lx\n", __FUNCTION__, rw.reg, rw.value);
                                                                    ^
mem_rw.cpp: In member function 'uint8_t MemRW::WriteFPGAREg_64(uint32_t, uint64_t)':
mem_rw.cpp:1397:66: warning: format '%lx' expects argument of type 'long unsigned int', but argument 3 has type 'unsigned int' [-Wformat=]
    printf("%s: reg:0x%lx val:0x%lx\n", __FUNCTION__, rw.reg, rw.value);
                                                                    ^
make[1]: Leaving directory '/home/user/Desktop/PCIe/code/trunk/Source_Code/lib'
make -C app_src/console_app/
make[1]: Entering directory '/home/user/Desktop/PCIe/code/trunk/Source_Code/app_src/console_app'
g++ -I../include -L../lib/main.cpp -std=c++11 -Wall -Wextra -g -DBUILD_LIB -DDMA_TEST -lmem_rw -lc -o ThroughputDemo
main.cpp: In function 'void displayPciResources()':
main.cpp:1014:16: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
    for (i = 0; i < PciInfo->numBARs; i++) {
                    ^
main.cpp:1016:57: warning: format '%x' expects argument of type 'unsigned int', but argument 2 has type 'ULONG [aka long unsigned int]' [-Wformat=]
    printf(" Addr: 0x%08x", PciInfo->BAR[i].physStartAddr);
```

Figure 8.2: run the script

- For run the motor initialization object will appear as mentioned in below figure.

```
File Edit View Search Terminal Help
main.cpp:1649:54: warning: format '%lx' expects argument of type 'long unsigned int', but argument 2 has type 'u64 (aka long long unsigned int)' [-Wformat=]
printf("data_64_CMD_ADDR: 0x%lx\n", data_64_CMD_ADDR);
^
main.cpp:1650:58: warning: format '%lx' expects argument of type 'long unsigned int', but argument 2 has type 'u64 (aka long long unsigned int)' [-Wformat=]
printf("data_64_DATA_DUMMY: 0x%lx\n", data_64_DATA_DUMMY);
^
main.cpp: In function 'void getMotorStatus(u32)':
main.cpp:1821:54: warning: format '%lx' expects argument of type 'long unsigned int', but argument 2 has type 'u64 (aka long long unsigned int)' [-Wformat=]
printf("data_64_CMD_ADDR: 0x%lx\n", data_64_CMD_ADDR);
^
main.cpp:1822:58: warning: format '%lx' expects argument of type 'long unsigned int', but argument 2 has type 'u64 (aka long long unsigned int)' [-Wformat=]
printf("data_64_DATA_DUMMY: 0x%lx\n", data_64_DATA_DUMMY);
^
main.cpp:1963:14: warning: unused variable 'data' [-Wunused-variable]
uint64_t data;
^
make[1]: Leaving directory '/home/user/Desktop/PCIE/code/trunk/Source_Code/app_src/console_app'
make -C drv_src/lthruput_drv/
make[1]: Entering directory '/home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv'
make -C /lib/modules/5.4.0-110-generic/build SUBDIRS=/home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv modules
make -C /lib/modules/5.4.0-110-generic/build ns/home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv modules
make[2]: Entering directory '/usr/src/linux-headers-5.4.0-110-generic'
CC [M] /home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv/lthruput_main.o
/home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv/lthruput_main.c: In function 'PCIEIOCTL':
/home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv/lthruput_main.c:393:17: warning: unused variable 'hw_buf' [-Wunused-variable]
unsigned char *hw_buf;
^
/home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv/lthruput_main.c: In function 'ParsePCIElinkCap':
/home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv/lthruput_main.c:993:5: warning: this statement may fall through [-Wimplicit-fallthrough=]
printf("lthru_demo: MSI Capability Structure @ %x\n", index);
^
/home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv/lthruput_main.c:995:4: note: here
case 6: // CompactPCI Hot Swap
^
Building modules, stage 2.
MODPOST 1 modules
CC [M] /home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv/lthruput_main.mod.o
LD [M] /home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv/lthruput_main.ko
make[2]: Leaving directory '/usr/src/linux-headers-5.4.0-110-generic'
make[1]: Leaving directory '/home/user/Desktop/PCIE/code/trunk/Source_Code/drv_src/lthruput_drv'
NOK
After the invoke
Initializing the object

Kindly enter one of the options.
To get the number of active nodes enter 1
To do default motor configuration enter 2
To Start a motor enter 3
To Stop a motor enter 4
To Power Off a motor enter 5
To read a register enter 6
To disconnect and exit enter 100
```

Figure 8.3: initializing the object

4. To get the number of active nodes: Enter 1
5. To do default motor configuration: Enter 2
6. After that enter the node between 1 to 16
7. Then Motor configuration will appear as mentioned in below figure.

Figure 8.4: Motor Configuration

- © 2022 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.


```

        Kindly enter one of the options.
        To get the number of active nodes enter 1
        To do default motor configuration enter 2
        To Start a motor enter 3
        To Stop a motor enter 4
        To Power Off a motor enter 5
        To read a register enter 6
        To disconnect and exit enter 100

3
Enter the node id and the target RPM (separated by space).
1 120
Going to run at 120 RPM.
=====DoMotorConfiguration=====
WriteFPGAREg_64: reg:0x3028 val:0xFFFFFFFF
ReadFPGAREg_64: reg:0x3020 val:0x0
ReadFPGAREg_64: reg:0x3020 val:0x0
WriteFPGAREg_64: reg:0x3020 val:0x2
ReadFPGAREg_64: reg:0x3008 val:0x40000000
ReadFPGAREg_64: reg:0x3008 val:0x40000000
Chain : 1 FINAL ACTIVE NODE 0

WriteFPGAREg_64: reg:0x3108 val:0X10000000
WriteFPGAREg_64: reg:0x3110 val:0X40
WriteFPGAREg_64: reg:0x3100 val:0X10000000
WriteFPGAREg_64: reg:0x3100 val:0X0
WriteFPGAREg_64: reg:0x3118 val:0X1
ReadFPGAREg_64: reg:0x3110 val:0x400000040
ReadFPGAREg_64: reg:0x3110 val:0x400000040
Frame_length : 8, node_data_length : 64,node_enable : 1, chain_select_var : 1, add: 0x3110,

Node Num 0

Node Enable 1

WriteFPGAREg_64: reg:0x3100 val:0X1
WriteFPGAREg_64: reg:0x3100 val:0X0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0

```

Figure 8.5: Start motor Node id and target RPM

10. To Stop a Motor: Enter 4

11. Then enter the node id same as selected before.

```

        Kindly enter one of the options.
        To get the number of active nodes enter 1
        To do default motor configuration enter 2
        To Start a motor enter 3
        To Stop a motor enter 4
        To Power Off a motor enter 5
        To read a register enter 6
        To disconnect and exit enter 100

4

        Enter the node id.

1

=====DoMotorConfiguration=====
WriteFPGAREg_64: reg:0x3028 val:0xFFFFFFFF
ReadFPGAREg_64: reg:0x3020 val:0x1
ReadFPGAREg_64: reg:0x3020 val:0x1
=====DoMotorConfiguration=====
WriteFPGAREg_64: reg:0x3028 val:0xFFFFFFFF
ReadFPGAREg_64: reg:0x3020 val:0x1
ReadFPGAREg_64: reg:0x3020 val:0x1

```

Figure 8.6: Stop motor node id

12. To Power Off a Motor: Enter 5
13. Enter the Node id same as entered before

```

        Kindly enter one of the options.
        To get the number of active nodes enter 1
        To do default motor configuration enter 2
        To Start a motor enter 3
        To Stop a motor enter 4
        To Power Off a motor enter 5
        To read a register enter 6
        To disconnect and exit enter 100

5

        Enter the node id.

1
=====DoMotorConfiguration=====
WriteFPGAREg_64: reg:0x3028 val:0xFFFFFFFF
ReadFPGAREg_64: reg:0x3020 val:0x0
ReadFPGAREg_64: reg:0x3020 val:0x0
WriteFPGAREg_64: reg:0x3020 val:0x2
ReadFPGAREg_64: reg:0x3008 val:0x40000000
ReadFPGAREg_64: reg:0x3008 val:0x40000000
Chain : 1 FINAL ACTIVE NODE 0

WriteFPGAREg_64: reg:0x3108 val:0X10000000
WriteFPGAREg_64: reg:0x3110 val:0X40
WriteFPGAREg_64: reg:0x3100 val:0X10000000
WriteFPGAREg_64: reg:0x3100 val:0X0
WriteFPGAREg_64: reg:0x3118 val:0X1
ReadFPGAREg_64: reg:0x3110 val:0x400000040
ReadFPGAREg_64: reg:0x3110 val:0x400000040
frame_length : 8, node_data_length : 64,node_enable : 1, chain_select_var : 1, add: 0x3110,

Node Num 0

Node Enable 1

WriteFPGAREg_64: reg:0x3100 val:0X1
WriteFPGAREg_64: reg:0x3100 val:0X0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0
ReadFPGAREg_64: reg:0x3010 val:0x0

```

Figure 8.7: Power off node id

```

ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
Config: no int err

=====DoMotorConfiguration=====
WriteFPGAReg_64: reg:0x3028 val:0xFFFFFFFF
ReadFPGAReg_64: reg:0x3020 val:0x2
ReadFPGAReg_64: reg:0x3020 val:0x2
WriteFPGAReg_64: reg:0x3020 val:0x2
ReadFPGAReg_64: reg:0x3008 val:0x40000000
ReadFPGAReg_64: reg:0x3008 val:0x40000000
Chain : 1 FINAL ACTIVE NODE 0

WriteFPGAReg_64: reg:0x3108 val:0X10000000
WriteFPGAReg_64: reg:0x3110 val:0X40
WriteFPGAReg_64: reg:0x3100 val:0X10000000
WriteFPGAReg_64: reg:0x3100 val:0X0
WriteFPGAReg_64: reg:0x3118 val:0X1
ReadFPGAReg_64: reg:0x3110 val:0x400000040
ReadFPGAReg_64: reg:0x3110 val:0x400000040
frame_length : 8, node_data_length : 64, node_enable : 1, chain_select_var : 1, add: 0x3110,

Node Num 0

Node Enable 1

WriteFPGAReg_64: reg:0x3108 val:0X1
WriteFPGAReg_64: reg:0x3100 val:0X0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0x0
ReadFPGAReg_64: reg:0x3010 val:0xFFFFFFFFFFFFFFFF
ReadFPGAReg_64: reg:0x3120 val:0xFFFFFFFFFFFFFFFF
value: 0xffffffff
Config: int info err

```

Figure 8.8: Power off node id prints

14. To read a register: Enter 6
15. Enter Reg for read: 1

```

        Kindly enter one of the options.
        To get the number of active nodes enter 1
        To do default motor configuration enter 2
        To Start a motor enter 3
        To Stop a motor enter 4
        To Power Off a motor enter 5
        To read a register enter 6
        To disconnect and exit enter 100
0
Enter Reg for read:1
ReadFPGAReg_64: reg:0x0 val:0xFFFFFFFFFFFFFFFF
reg:1 val:18446744073709551615

```

Figure 8.9: read a register

16. To disconnect and exit: Enter 100

```
Kindly enter one of the options.  
To get the number of active nodes enter 1  
To do default motor configuration enter 2  
To Start a motor enter 3  
To Stop a motor enter 4  
To Power Off a motor enter 5  
To read a register enter 6  
To disconnect and exit enter 100  
100  
/hone/user/Desktop/PCie/code/trunk  
user@user-OptiPlex-3040:trunk$
```

Figure 8.10: Disconnect and exit

Appendix A. Raspberry pi Setup

A.1 SD card Format

1. Insert your **microSD card** into the USB card reader and plug into the PC or laptop.
2. Download the SD card formatter from the given link <https://www.sdcard.org/downloads/formatter/sd-memory-card-formatter-for-windows-download/>.

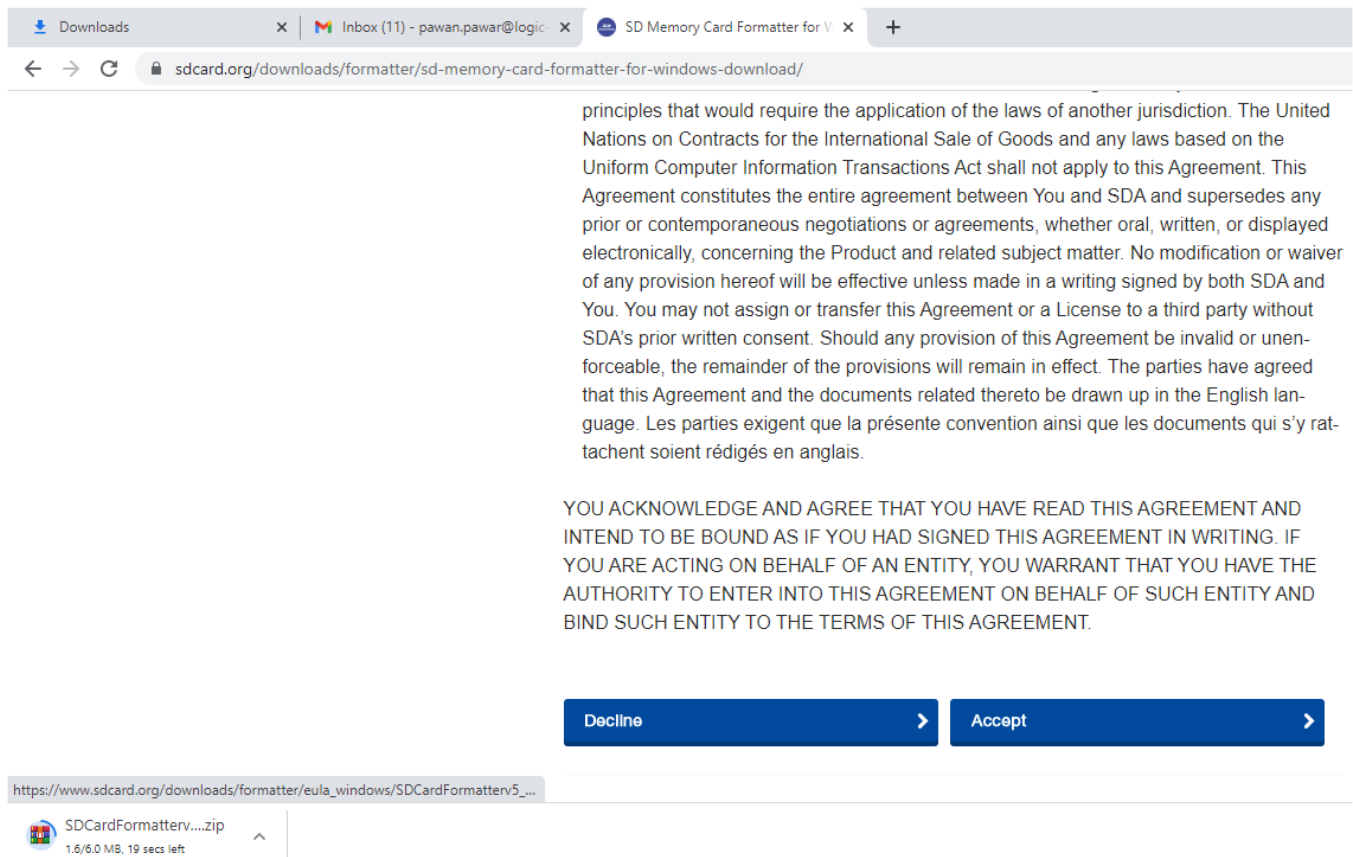


Figure 8.11: SD card installation

3. Click on the Accept and file will be download.
4. Unzip the Downloaded file and install the SD card format Application.
5. Now Open the SD card Formatter 5.0.1 setup.
6. Now choose the **Quick format** and click on **Format** option.

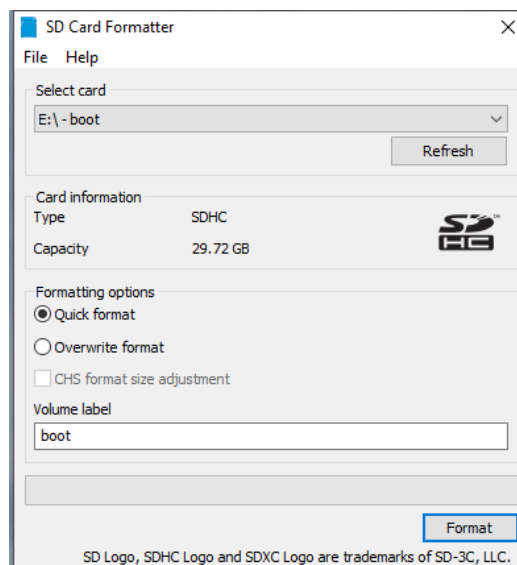


Figure 8.12: SD card Formatter

A.2 Raspberry Pi Imager Setup

1. Download the Imager 1.6.2 Application for raspberry pi from the given link <https://www.raspberrypi.com/software/>.
2. Now Open the downloaded [imager_1.6.2.exe file](#).

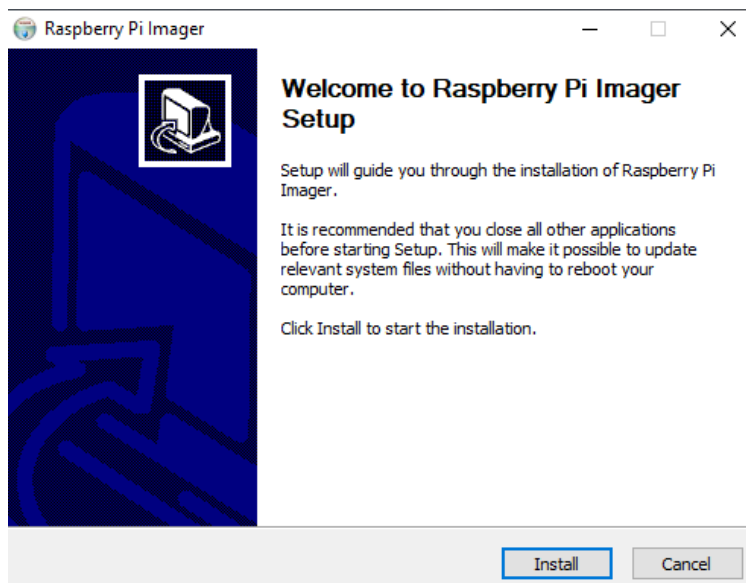


Figure 8.13: Imager Installation

3. Now click on the **Install** button.
4. Once it will completed then click on **Finish** button.
5. In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install it on.
6. Now select the **Raspberry PI OS (32 bit)** option.

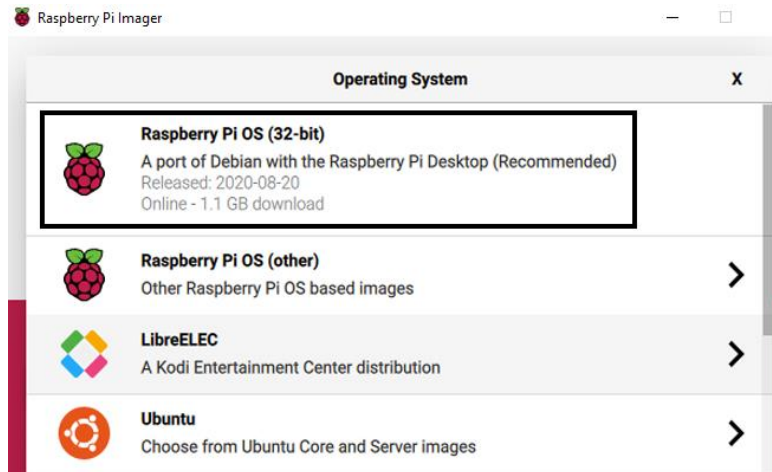


Figure 8.14: OS selection

7. Now select the **Mass Storage device Media** option.

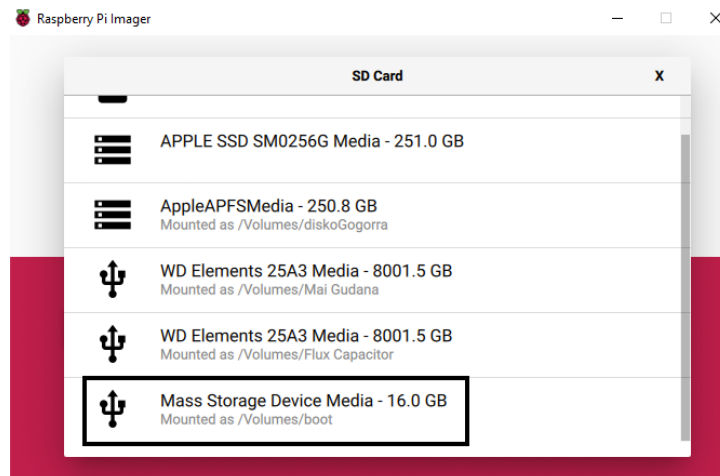


Figure 8.15: SD card selection

8. Now simply click on the **Write** Button.



Figure 8.16: Write the SD card

9. Wait for the Raspberry Pi Imager to finish writing
10. Once you get the following message, you can eject your SD card

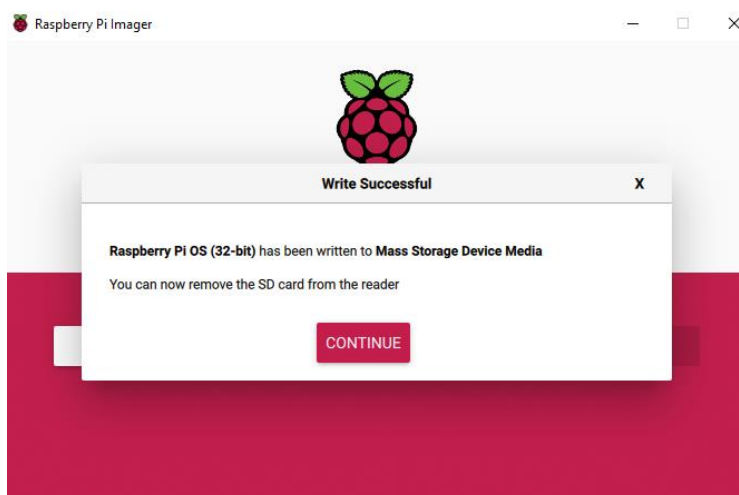


Figure 8.17: SD Card write successful

A.3 Start up the Raspberry Pi (Server End)

1. Insert the SD card into the Raspberry Pi slot.
2. Connect the Ethernet cable into the Ethernet port.
3. Connect the Micro USB cable into the Raspberry Pi and HDMI side cable into the Monitor.
4. Connect the Type-C Power cable into the Type-C port.
5. Connect the Mouse and Key-Board into the USB 3.0 port, if required.

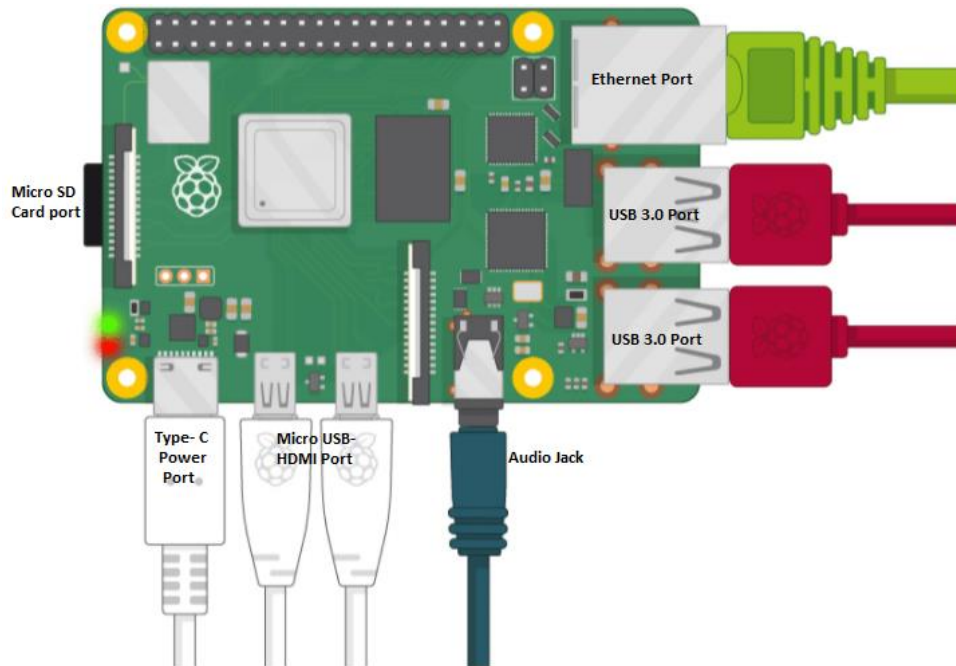


Figure 8.18: Raspberry pi setup image

6. Now turn ON the Power of Monitor and raspberry pi.
7. After a few seconds the Raspberry Pi OS desktop will appear.

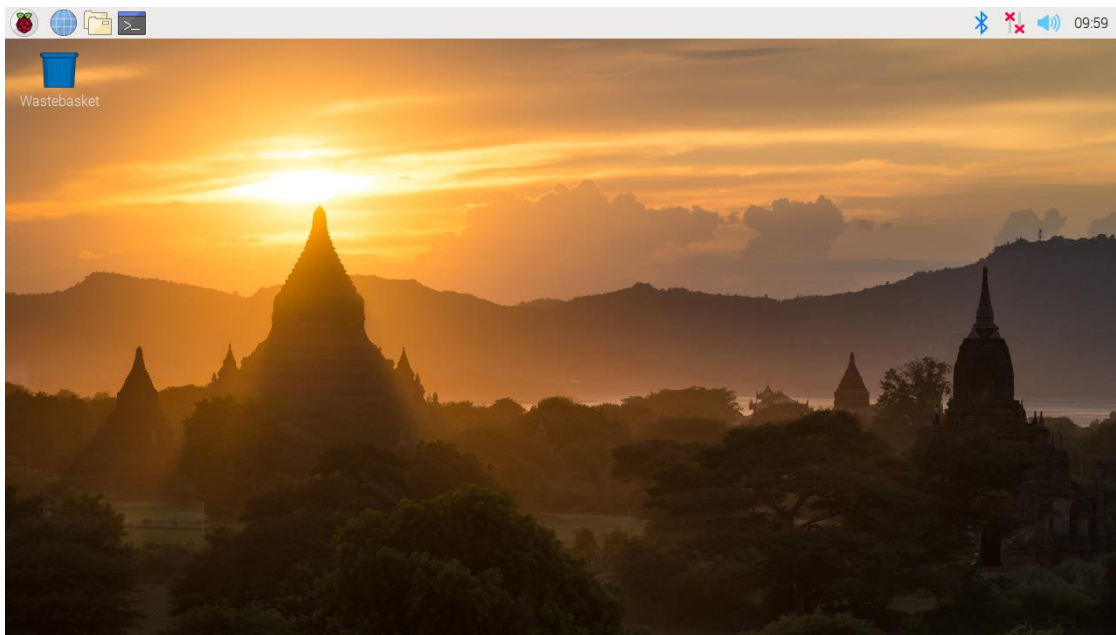


Figure 8.19: Raspberry pi OS desktop

8. When you start your Raspberry Pi for the first time, the **Welcome to Raspberry Pi** application will pop up and guide you through the initial setup.



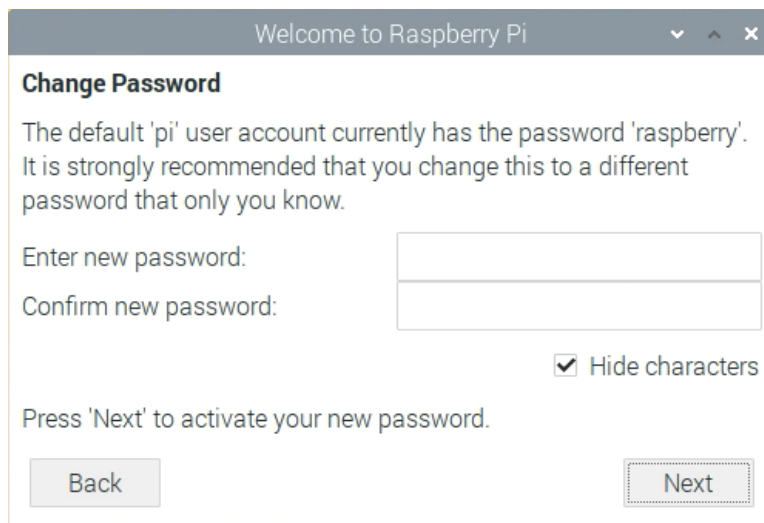
Figure 8.20: Initial step

9. Click on the **Next** Button.
10. Set your **Country**, **Language**, and **Timezone**, then click on **Next** again.



Figure 8.21: Country, Language and Timezone selection

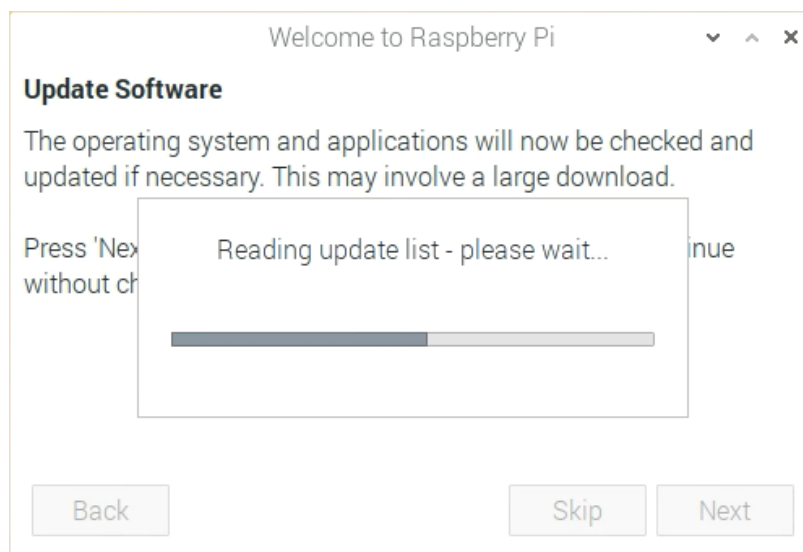
11. Enter a new password for your Raspberry Pi and click on **Next**.



The screenshot shows a window titled "Welcome to Raspberry Pi" with a close button (X) in the top right corner. The main heading is "Change Password". Below it, a message states: "The default 'pi' user account currently has the password 'raspberry'. It is strongly recommended that you change this to a different password that only you know." There are two input fields: "Enter new password:" and "Confirm new password:". To the right of these fields is a checkbox labeled "Hide characters" which is checked. Below the input fields, a message says "Press 'Next' to activate your new password." At the bottom, there are two buttons: "Back" on the left and "Next" on the right.

Figure 8.22: change password

12. Click on **Next**, and let the wizard check for updates to Raspberry Pi OS and install them.



The screenshot shows the same "Welcome to Raspberry Pi" window, but now the heading is "Update Software". The message reads: "The operating system and applications will now be checked and updated if necessary. This may involve a large download." Below this, there is a smaller window with the text "Reading update list - please wait..." and a progress bar that is approximately half-filled. To the left of this smaller window, the text "Press 'Next' without ch" is partially visible. At the bottom of the main window, there are three buttons: "Back" on the left, "Skip" in the middle, and "Next" on the right.

Figure 8.23: update software

13. Click on **Restart** to finish the setup.



Figure 8.24: setup complete

Appendix B. Compile the complete code

B.1 Make file (OPCUA)

1. Open the terminal.
2. Unzip the bundle
3. Write the command: **unzip OPCUA_ServerApp_v0_1_2.zip**
4. Write the command: **cd OPCUA_ServerApp_v0_1_2/**
5. Go to the Scripts directory
6. Write the command: **cd Scripts/**
7. Now Write the command for compilation: **make -j4**

```
pi@raspberrypi:~/OPCUA_Server $ ls
bin lib Objs Scripts src tools
pi@raspberrypi:~/OPCUA_Server $ cd Scripts/
pi@raspberrypi:~/OPCUA_Server/Scripts $ make -j4
gcc -std=c99 -D_GNU_SOURCE -DUA_ENABLE_AMALGAMATION=1 -o server MainSys_SerApp.o open62541.o xml_code.o ConfigMgr.o RFL.o CommMgr.o UartIface.o
-lpthread -lopen62541
pi@raspberrypi:~/OPCUA_Server/Scripts $
```

Figure 8.25: Compilation

8. For removing the all objects and binary files, if necessary
9. Write the command: **make clean**

```
pi@raspberrypi:~/OPCUA_Server/Scripts $ make clean
rm ../bin/server ../Objs/*.o
pi@raspberrypi:~/OPCUA_Server/Scripts $
```

Figure 8.26: Make clean

B.2 To run the server (OPCUA)

1. After make clean or make -j4, Go to the OPCUA_Server/Scripts directory.
2. Go to the bin directory.
3. Write the command: **cd ../bin/**
4. to start the server, write the command: **./server**
5. Server will start.

```
pi@raspberrypi:~/OPCUA_Server/Scripts $ cd ../bin/
pi@raspberrypi:~/OPCUA_Server/bin $ ls
server
pi@raspberrypi:~/OPCUA_Server/bin $ ./server
```

Figure 8.27: run server

B.3 Make file (Mqtt)

1. Open the terminal.

2. Unzip the bundle
3. Write the command: **unzip Mqtt_Lattice_AutomateStack_2_0.zip**
4. Write the command: **cd Mqtt_Lattice_AutomateStack_2_0/**
5. Go to the Scripts directory
6. Write the command: **cd Scripts/**
7. Now Write the command for compilation: **make**

```
pi@raspberrypi:~/Mqtt_Lattice_AutomateStack_2_0/Scripts $ make
gcc -L/home/pi/Mqtt_Lattice_AutomateStack_2_0/Scripts -c ../src/client.c ../src/config.h ../src/ConfigMgr.h ../src/ConnMgr.h ../src/RFL.h -lpthread
gcc -c ../src/ConfigMgr.c ../src/ConfigMgr.h ../src/RFL.h
gcc -c ../src/RFL.c ../src/RFL.h ../src/ConnMgr.h ../src/UartInterface.h ../src/ConfigMgr.h ../src/config.h
gcc -c ../src/ConnMgr.c ../src/ConnMgr.h ../src/UartInterface.h
gcc -c ../src/UartInterface.c ../src/UartInterface.h
gcc -o ../bin/server client.o ConfigMgr.o RFL.o ConnMgr.o UartInterface.o -lpaho-mqtt3c -lpthread -lm
pi@raspberrypi:~/Mqtt_Lattice_AutomateStack_2_0/Scripts $
```

Figure 8.28: Compilation

8. For removing the all objects and binary files, if necessary
9. Write the command: **make clean**

```
pi@raspberrypi:~/Mqtt_Lattice_AutomateStack_2_0/Scripts $ make clean
rm ../bin/server ../src/*.gch *.o
pi@raspberrypi:~/Mqtt_Lattice_AutomateStack_2_0/Scripts $
```

Figure 8.29: Make clean

B.4 To run the server (Mqtt)

1. After make clean or make -j4, Go to the OPCUA_Server/Scripts directory.
2. Go to the bin directory.
3. Write the command: **cd ../bin/**
4. to start the server, write the command: **./server**
5. Server will start.

```
pi@raspberrypi:~/Mqtt_Lattice_AutomateStack_2_0/Scripts $ cd ../bin/
pi@raspberrypi:~/Mqtt_Lattice_AutomateStack_2_0/bin $ ls
server
pi@raspberrypi:~/Mqtt_Lattice_AutomateStack_2_0/bin $ ./server -v5
Host : eth0
IP : 10.0.1.244
-----Lattice Automate stack 2.0 version : 0.0.1 ---May 12, 2022-----
```

Figure 8.30: run server

Appendix C. Application Installation (Client End)

Note: All this Installation process will be done on a separate Desktop or Laptop, not in Raspberry pi system.

1. Check the Installer folder, where file is located.
2. Now Click on the Installer to install the application.

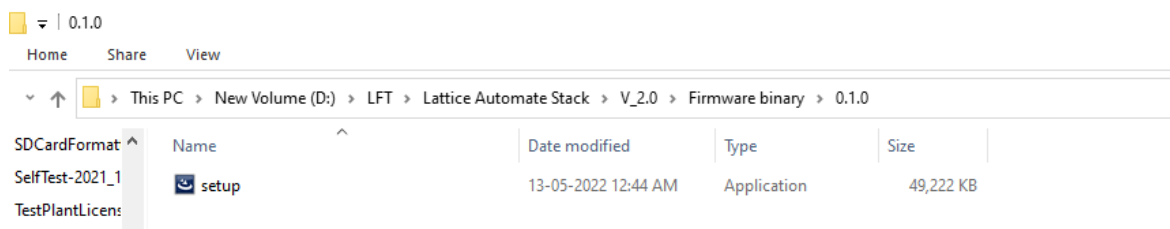


Figure 8.31: Installer directory

3. Wait for the Next.

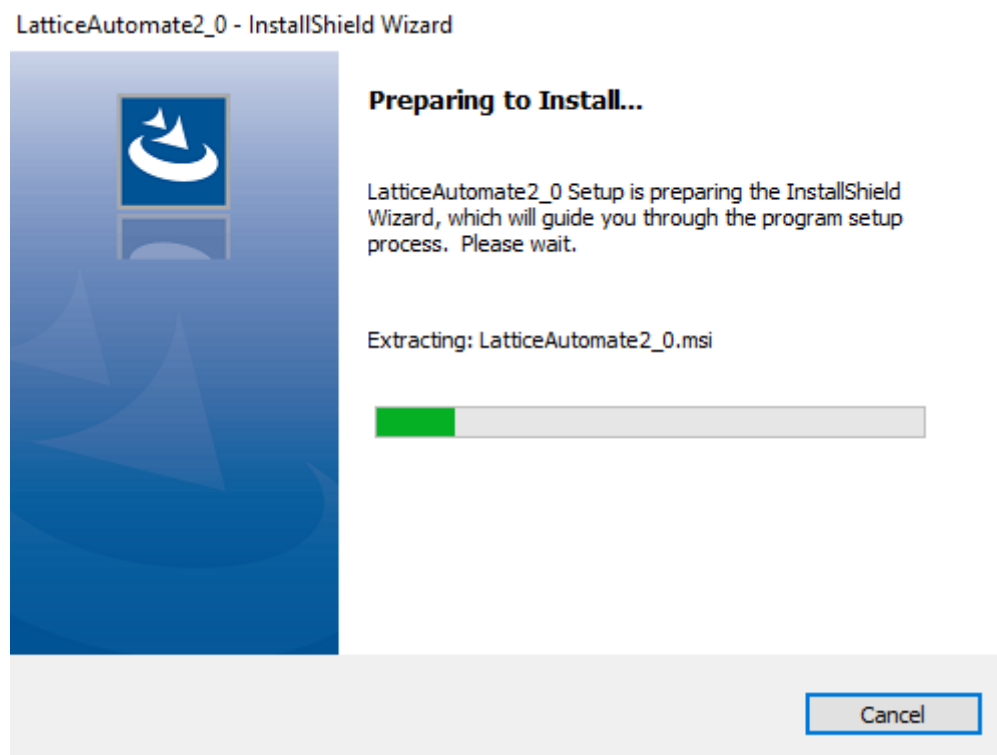


Figure 8.32: Initial Install step

4. Now again click on the **Next** Button.

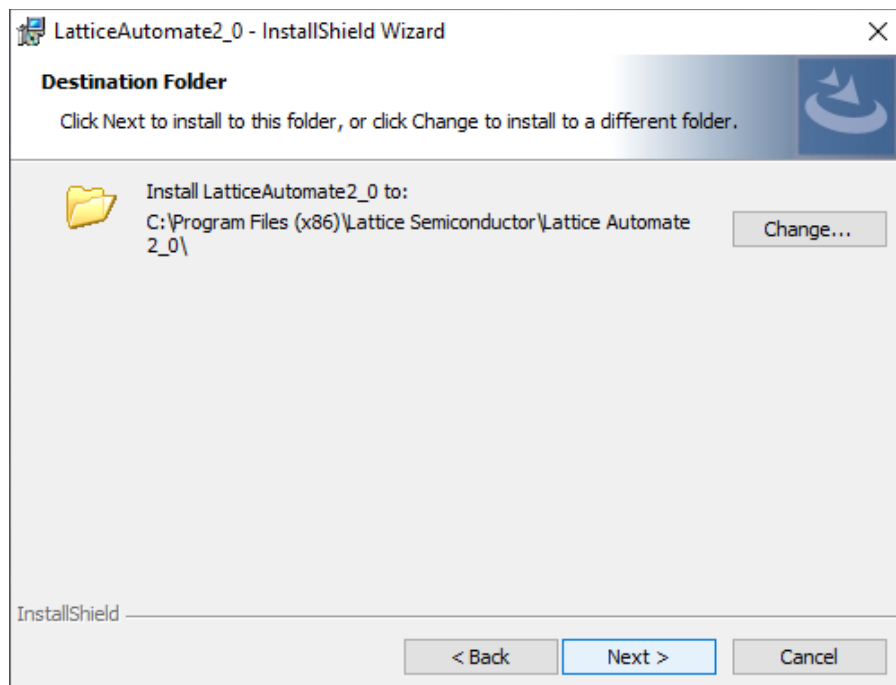


Figure 8.33: Next step

5. Now click on the **Install** button.

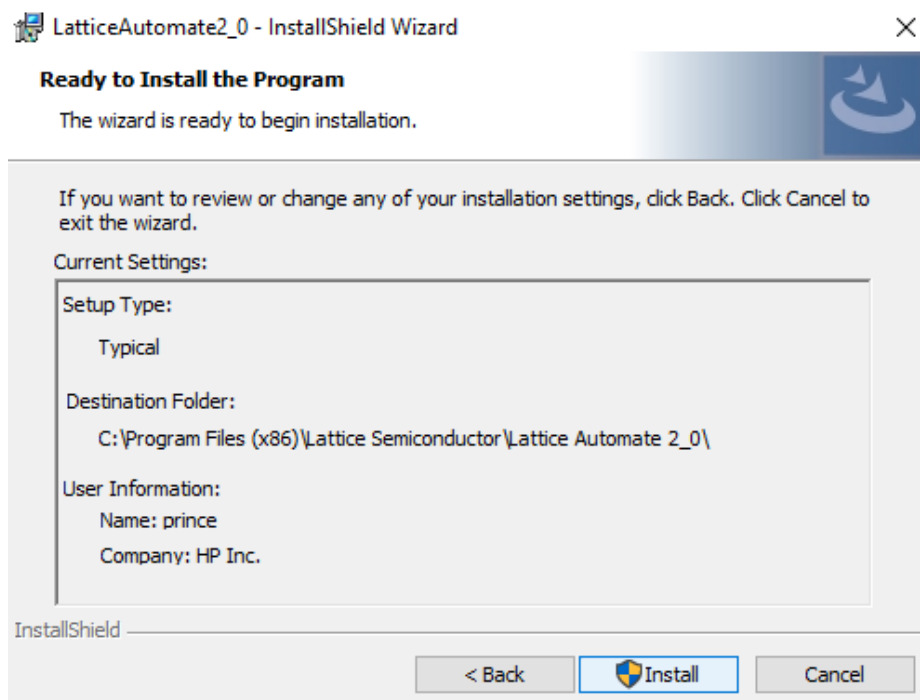


Figure 8.34: Click on Install

6. Wait for the **Next** button

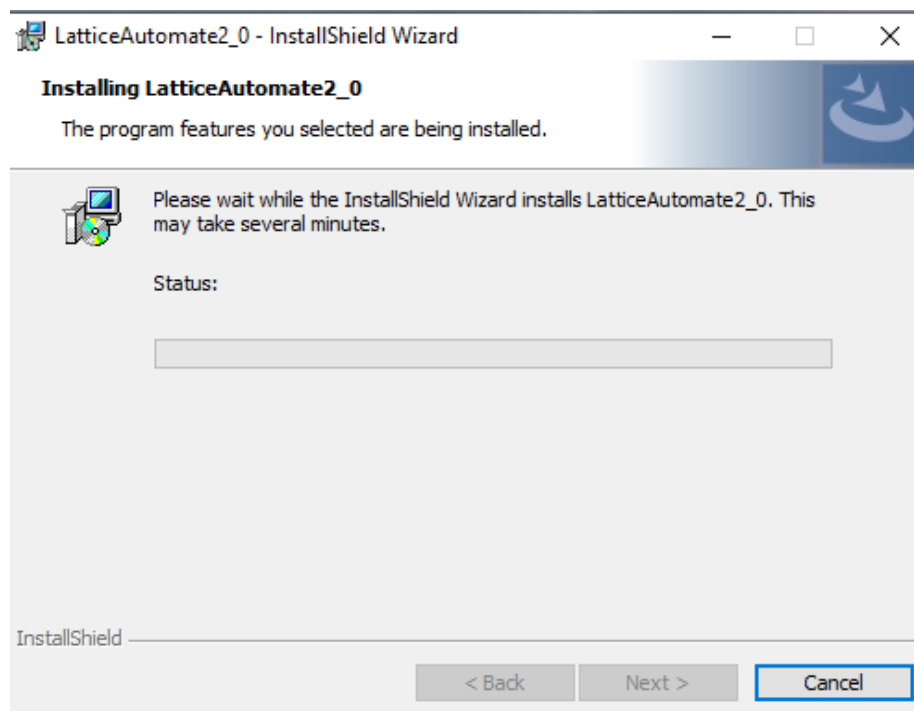


Figure 8.35: wait for the next

7. Wait for the Installation then click on the **Finish**.

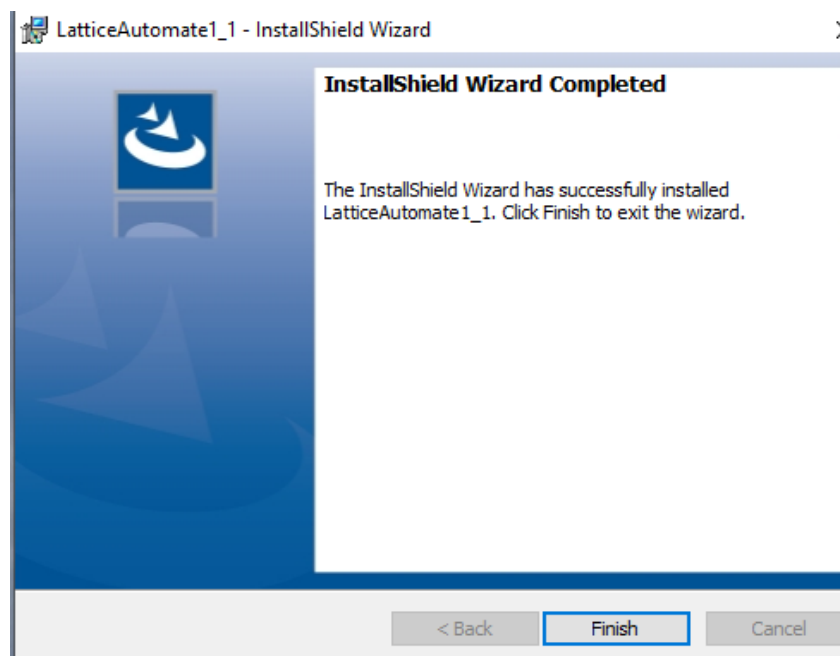


Figure 8.36: installation finish

Appendix D. Automate Stack 2.0 Bitfile and Binary Generation

D.1 Steps for bit file generation

MAIN SYSTEM

1. Open Generated Radiant Project in Radiant Tool.
2. Select Family: LFCPNX
3. Select Device: LFCPNX-100
4. Select Operating Condition: Industrial
5. Select Package: LFG672
6. Performance Grade: 9_High-Performance_1.0V
7. Part Number: LFCPNX-100-9LFG672I

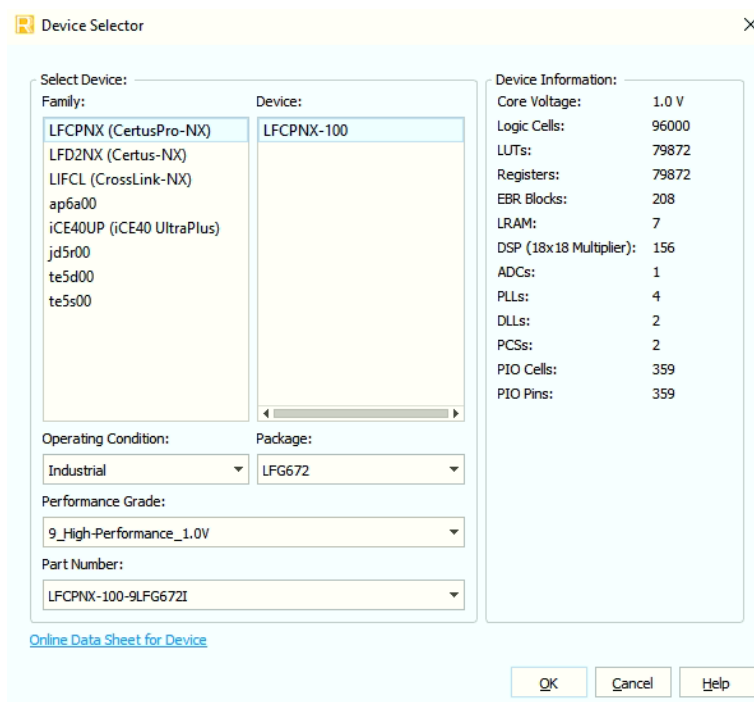


Figure 8.37: Lattice Radiant Device Selector for Main System

8. Change strategy as shown below: Frequency Parameter 250 MHz

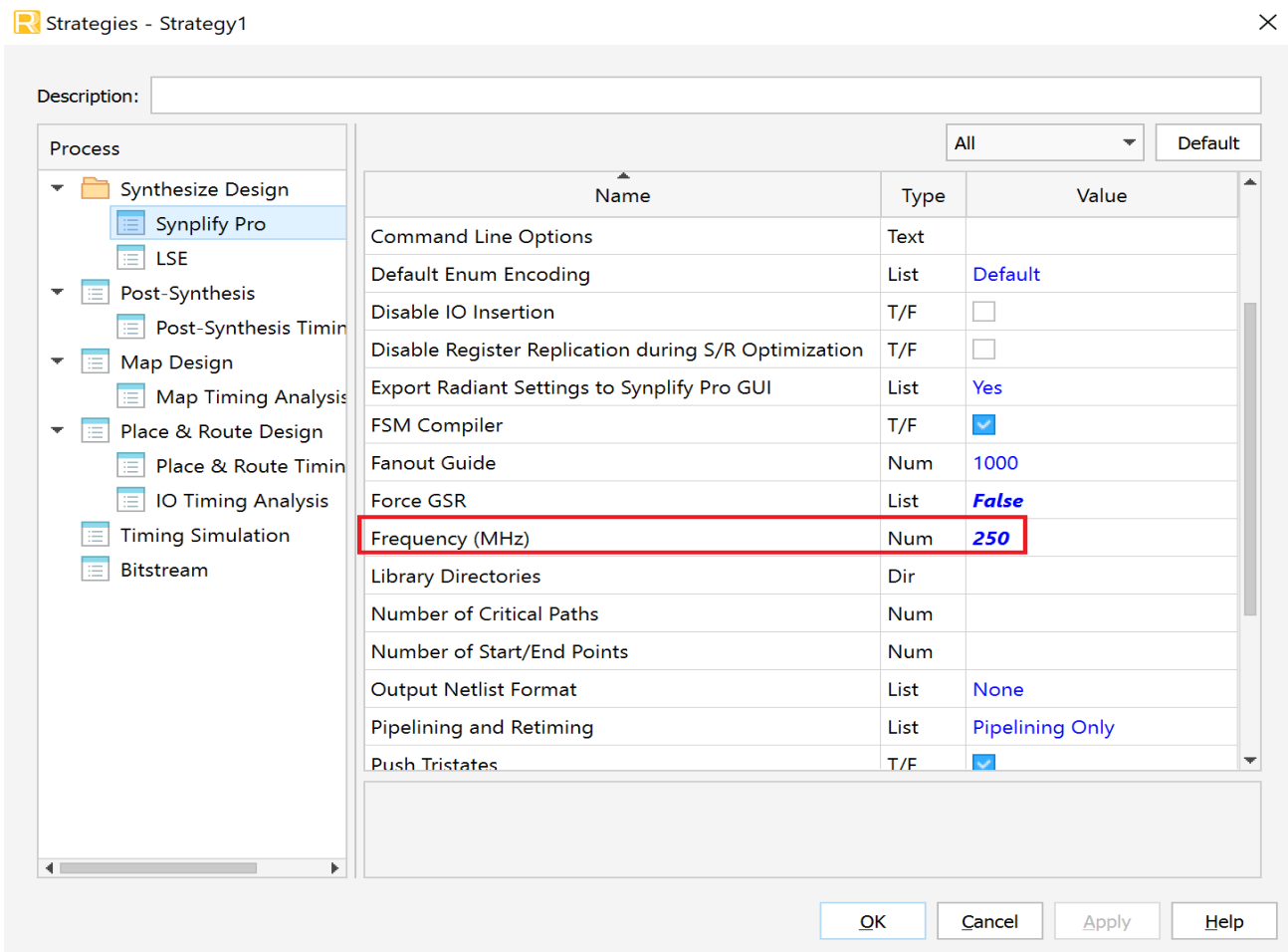


Figure 8.38: Strategy for Build Generation for Main System

9. Now go to the Strategy and select the Map Design
10. Now select the Map Timing Analysis.
11. Now select the highlighted part as mentioned below fig.

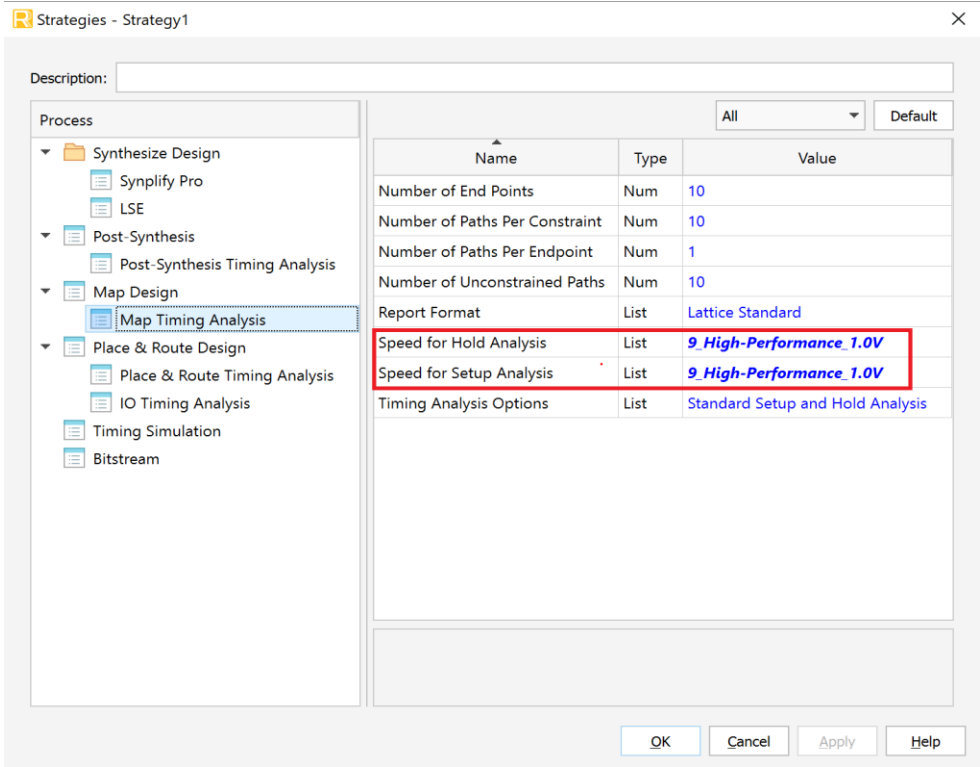


Figure 8.39: MAP analysis setting for Main system bitfile generation

12. Select the Place & Route Design
13. Select the only highlighted parts as mentioned below fig.

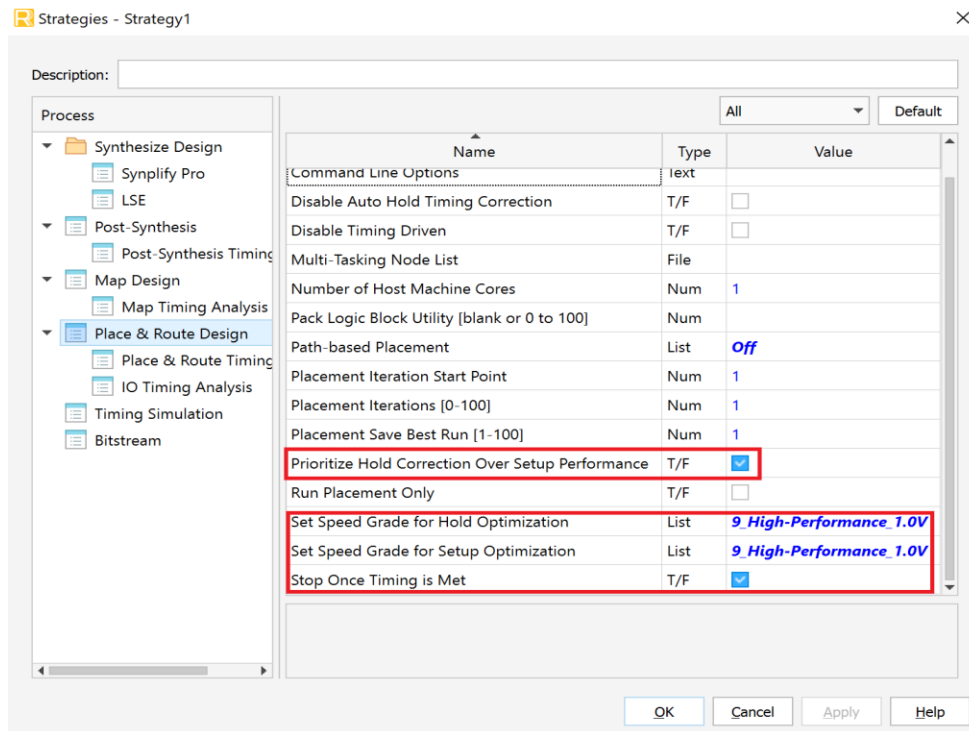


Figure 8.40: PAR setting for Main system bitfile generation

14. Select the Place and Route Timing Analysis.
15. Select the only highlighted parts as mentioned below fig.

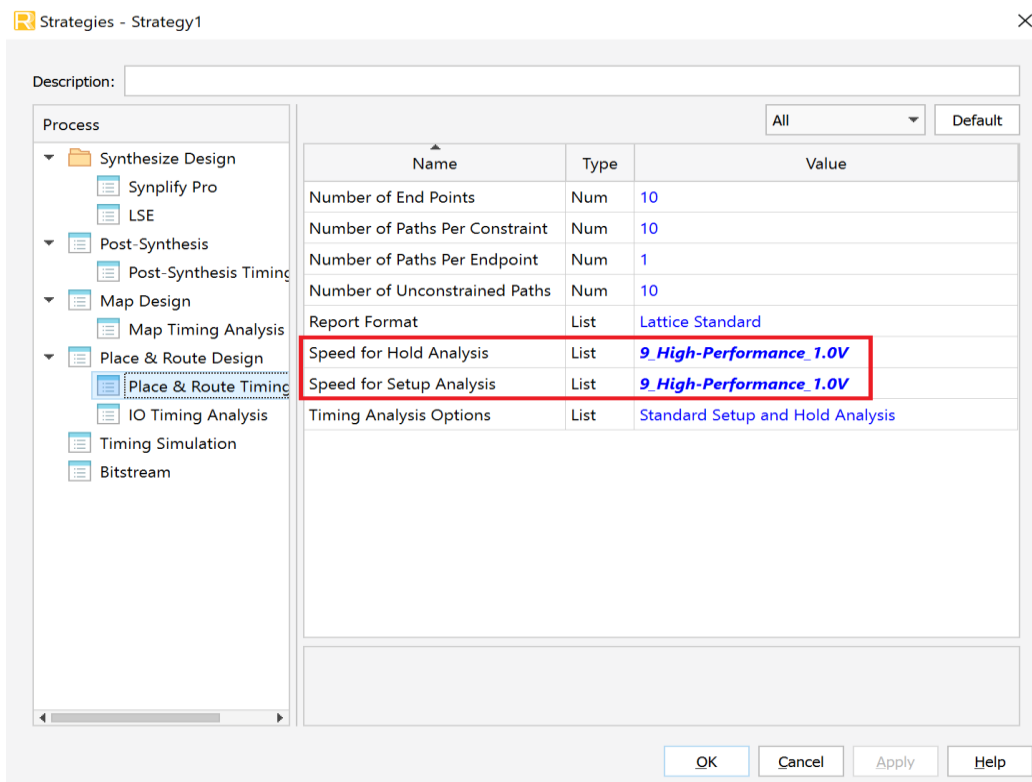


Figure 8.41: PAR Timing analysis setting for Main system bitfile generation

16. Use same PDC if FPGA is same otherwise update PDC file as per FPGA pins
17. Now click on the Device Constraint Editor as mentioned below.

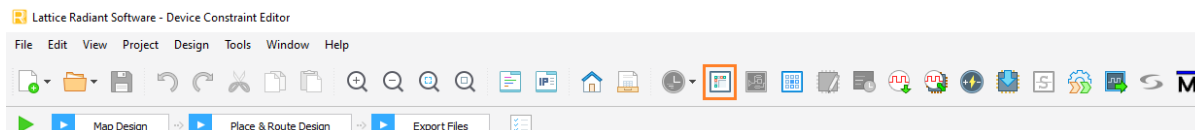


Figure 8.42: Device Constraint Selection for Main System

18. Now Click on the Global
19. Use below Global constraint: **Clock is 90 MHz.**

Name	Value
Junction Temperature(T)(C)	85
Voltage (V)	1
▼ SysConfig	
SLAVE_SPI_PORT	DISABLE
MASTER_SPI_PORT	DISABLE
SLAVE_I2C_PORT	DISABLE
SLAVE_I3C_PORT	DISABLE
JTAG_PORT	ENABLE
DONE_PORT	DISABLE
INITN_PORT	DISABLE
PROGRAMN_PORT	ENABLE
BACKGROUND_RECONFIG	OFF
DONE_EX	OFF
DONE_OD	ON
MCCLK_FREQ	90
TRANSFR	OFF
CONFIG_IOSLEW	FAST

Figure 8.43: Global Constraints for Main System

20. Click on the run all. It generates a bit file.

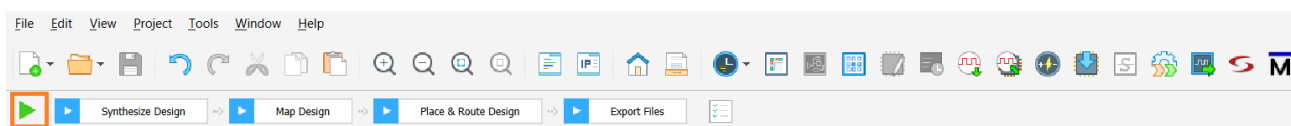


Figure 8.44. Run All button

21. Now open the propel builder 2.2 tool.
22. Double click on the system0_inst.
23. A pop-up will appear on the screen as mentioned below.

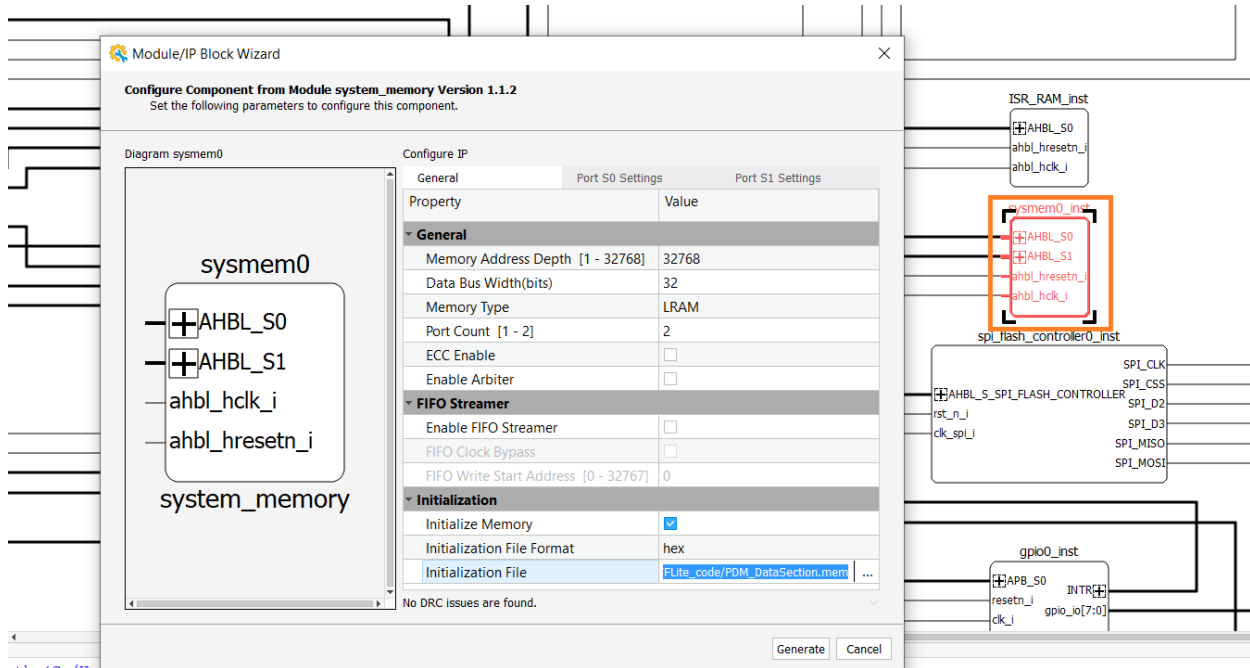


Figure 8.45. System Initialization File

24. Initialize Data memory with generated PDM_DataSection.mem file in TFLite_code folder of C project.
25. Click on the Generate button
26. Double click on the ISR_RAM_inst.
27. A pop-up will appear on the screen as mentioned below.

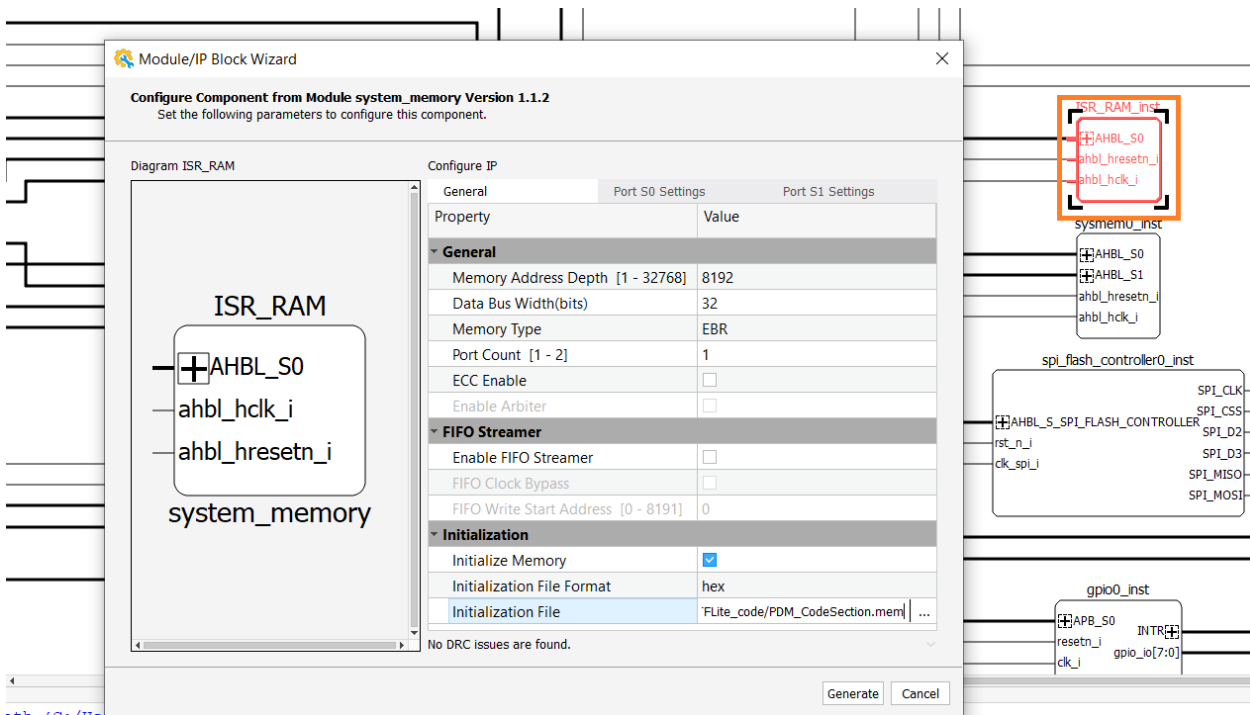


Figure 8.46. ISR RAM Initialization File

28. Initialize Data memory with generated PDM_codeSection.mem file in TFLite_code folder of C project.
29. Click on the Generate button
30. Click on the Validate button

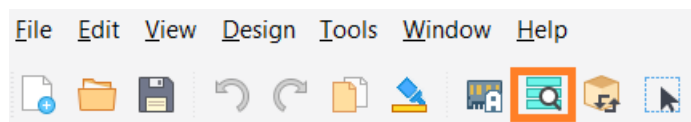


Figure 8.47. Validate Button

31. Now click on the Generate SGE button.

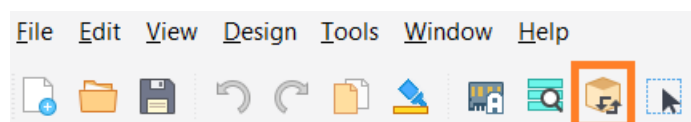


Figure 8.48. Generate SGE Button

32. Open the radiant tool from propel builder interface.

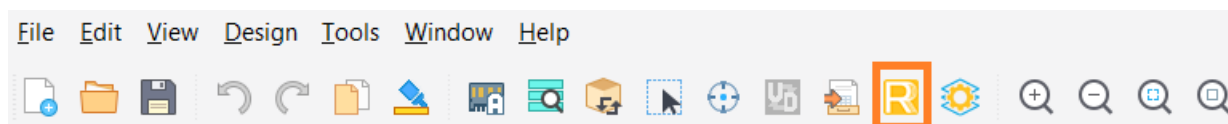


Figure 8.49. Radiant Tool Button

33. Select FPGA and update constraint (refer to steps 1 to 19).
34. Click on the run all. It generates a bit file.

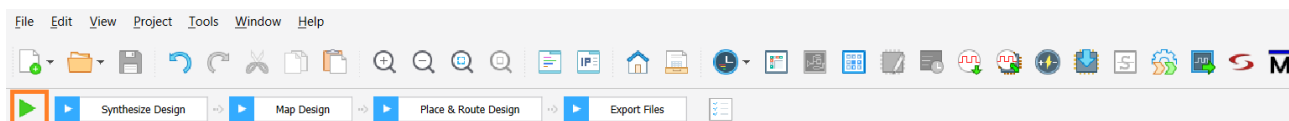


Figure 8.50. Run All Button

NODE SYSTEM

1. Open Generated Radiant Project in Radiant Tool.
2. Select Family: LFD2NX
3. Select Device: LFD2NX-40
4. Select Operating Condition: Commercial
5. Select Package: CABGA256
6. Performance Grade: 8_High-Performance_1.0V
7. Part Number: LFD2NX-40-8BG256C

Device Selector

×

Select Device:

Family:

LFCPNX (CertusPro-NX)
LFD2NX (Certus-NX)
LIFCL (CrossLink-NX)
ap6a00
iCE40UP (iCE40 UltraPlus)
jd5r00
te5d00
te5s00

Device:

LFD2NX-17
LFD2NX-40

Operating Condition:

Commercial

Package:

CABGA256

Performance Grade:

8_High-Performance_1.0V

Part Number:

LFD2NX-40-8BG256C

Device Information:

Core Voltage: 1.0 V
Logic Cells: 39000
LUTs: 32256
Registers: 32256
EBR Blocks: 84
LRAM: 2
DSP (18x18 Multiplier): 56
ADCs: 1
PLLs: 3
DLLs: 2
PCs: 1
PIO Cells: 197
PIO Pins: 197

[Online Data Sheet for Device](#)

OK

Cancel

Help

Figure 8.51: Lattice Radiant Device Selector for Node System

8. Change strategy as shown below: Frequency Parameter 150 MHz

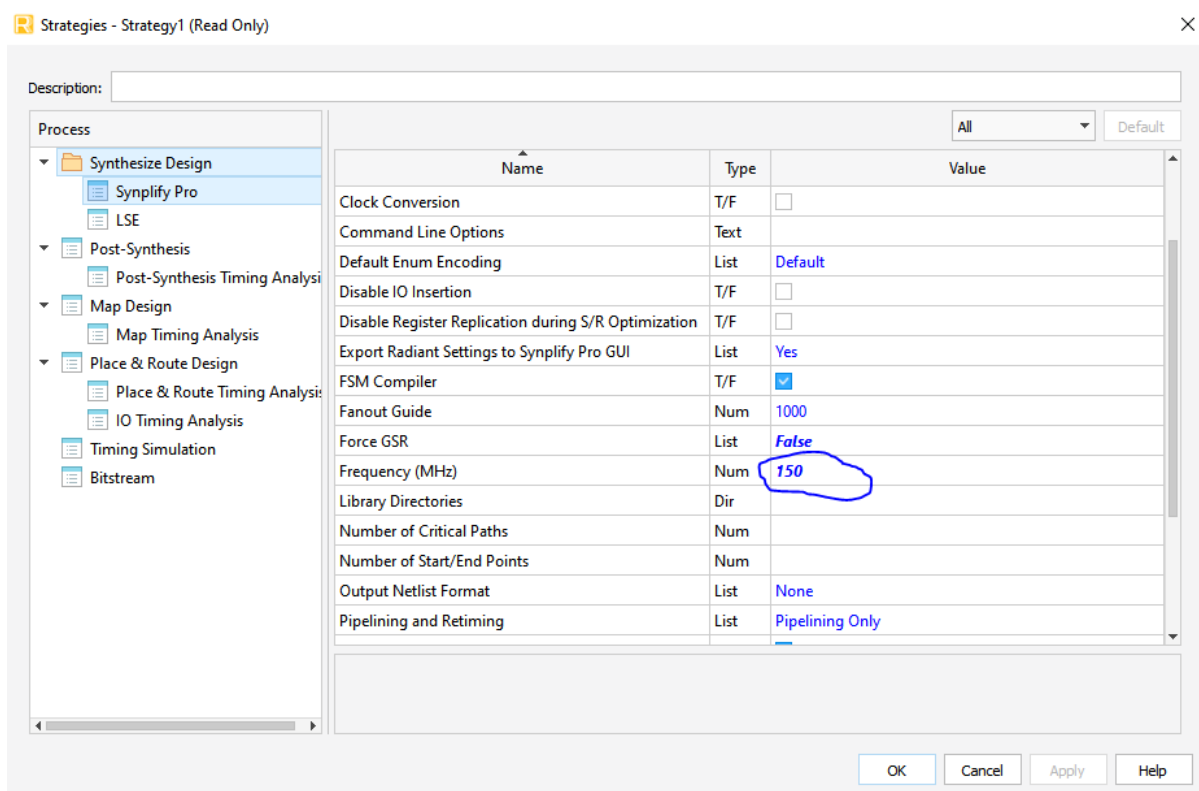


Figure 8.52: Strategy for Build Generation for Node System

9. Now go to the Strategy and select the Map Design
10. Now select the Map Timing Analysis.
11. Now select the highlighted part as mentioned below fig.

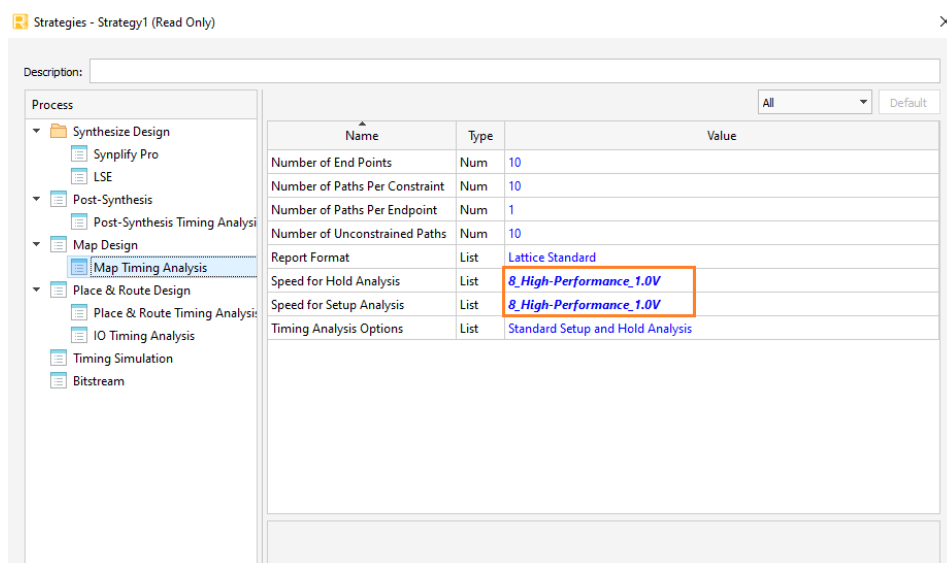


Figure 8.53: MAP analysis setting for Node system bitfile generation

12. Select the Place & Route Design

13. Select the only highlighted parts as mentioned below fig.

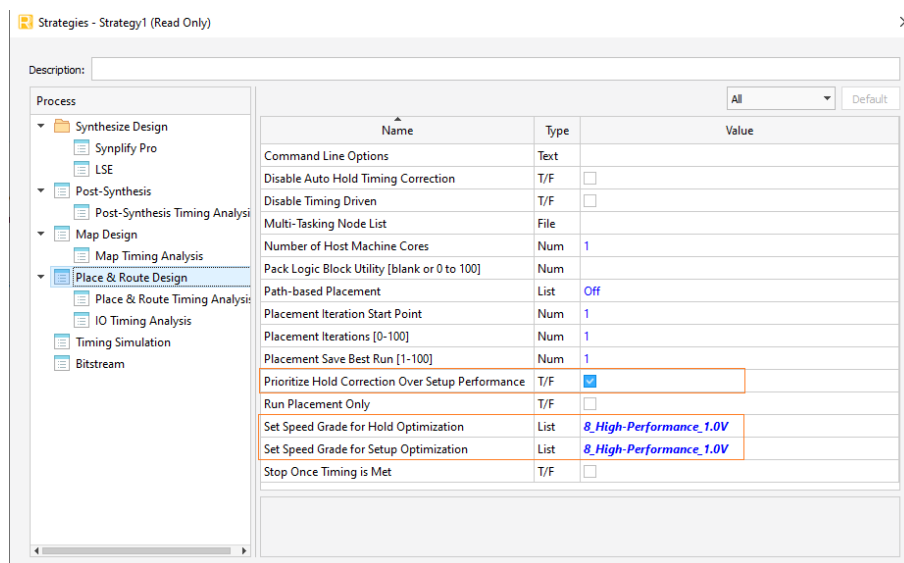


Figure 8.54: PAR setting for Node system bitfile generation

14. Select the Place and Route Timing Analysis.

15. Select the only highlighted parts as mentioned below fig.

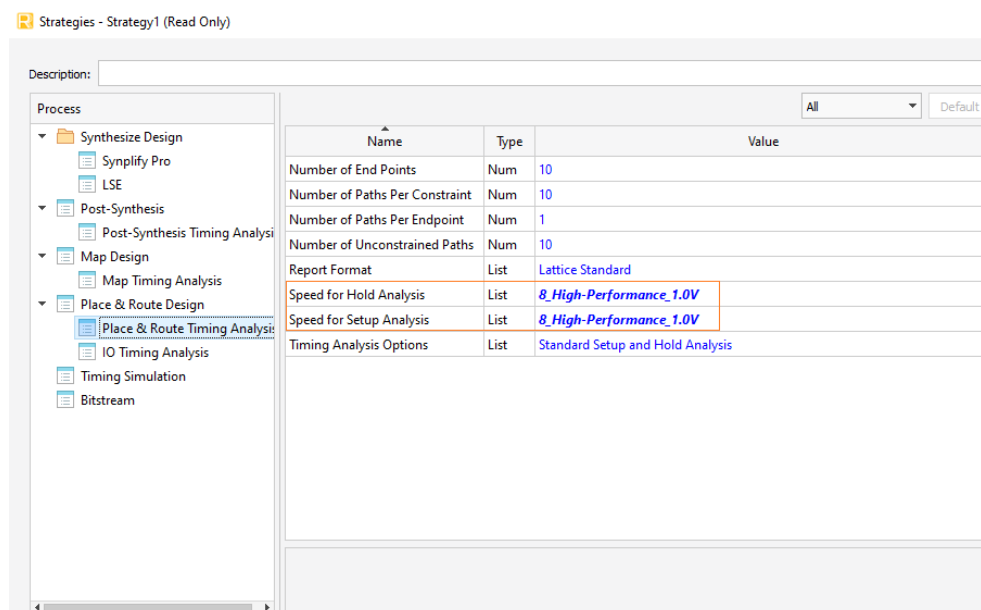


Figure 8.55: PAR Timing analysis setting for Node system bitfile generation

16. Use same PDC if FPGA is same otherwise update PDC file as per FPGA pins

17. Now click on the Device Constraint Editor as mentioned below.

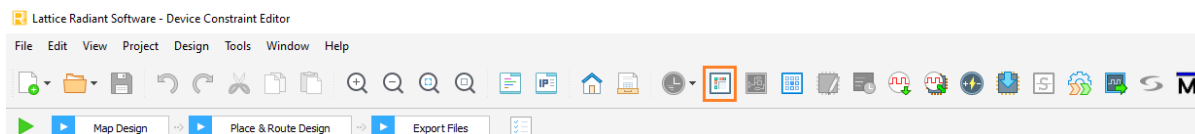


Figure 8.56: Device Constraint Selection for Node System

18. Now Click on the Global

19. Use below Global constraint: **Clock is 90 MHz.**

Name	Value
Junction Temperature(T)(C)	85
Voltage (V)	0.95
▼ SysConfig	
SLAVE_SPI_PORT	DISABLE
MASTER_SPI_PORT	DISABLE
SLAVE_I2C_PORT	DISABLE
SLAVE_I3C_PORT	DISABLE
JTAG_PORT	ENABLE
DONE_PORT	DISABLE
INITN_PORT	DISABLE
PROGRAMN_PORT	ENABLE
BACKGROUND_RECONFIG	OFF
DONE_EX	OFF
DONE_OD	ON
MCCLK_FREQ	90
TRANSFR	OFF
CONFIG_IOSLEW	FAST
CONFIG_SECURE	OFF
WAKE_UP	ENABLE_DONE_SYNC
COMPRESS_CONFIG	OFF
EARLY_IO_RELEASE	OFF
BOOTMODE	DUAL
CONFIGIO_VOLTAGE_BANK0	NOT_SPECIFIED
CONFIGIO_VOLTAGE_BANK1	NOT_SPECIFIED
▼ User Code	
UserCode Format	Binary
UserCode	00000000000000000000000000000000
Unique Id	0000

Figure 8.57: Global Constraints for Node System

20. Click on the run all. It generates a bit file.

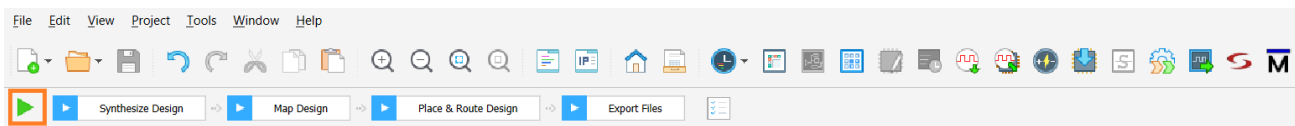


Figure 8.58. Run All

21. Now open the propel builder 2.2 tool.

22. Double click on the system0_inst.

23. A pop-up will appear on the screen as mentioned below.

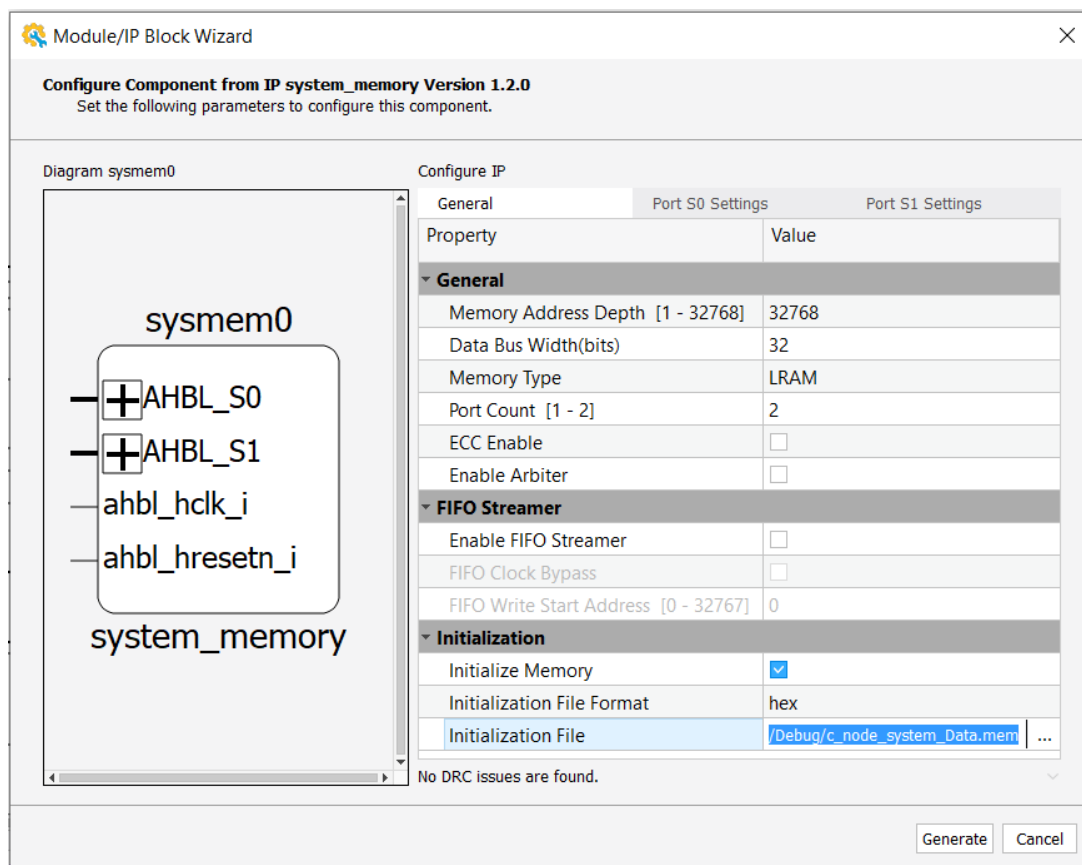


Figure 8.59. System0 Initialization

24. Initialize Data memory with generated c_node_system.mem file in debug folder of C project.

25. Click on the Validate button

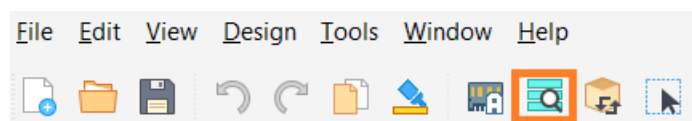


Figure 8.60. Validate Button

26. Now click on the Generate SGE button.

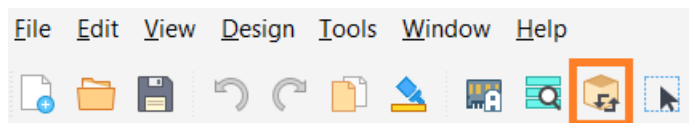


Figure 8.61. Generate SGE Button

27. Open the radiant tool from propel builder interface.

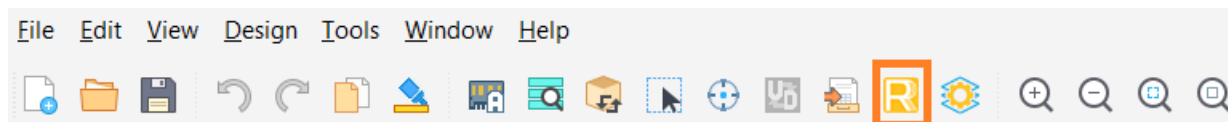


Figure 8.62. Radiant Tool Button

28. Select FPGA and update constraint (refer to steps 1 to 19).

29. Click on the run all. It generates a bit file.

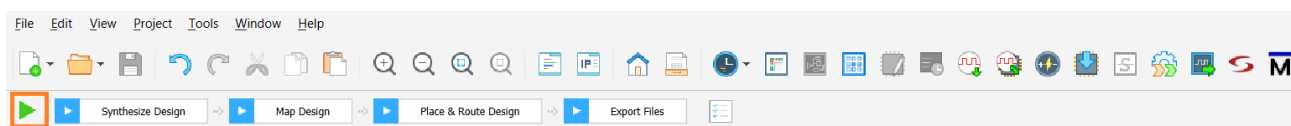


Figure 8.63. Run All Button

D.2 Steps for binary generation

1. Double click on the “Lattice Propel SDK 2.0” to open the dialogue box as shown in below fig.

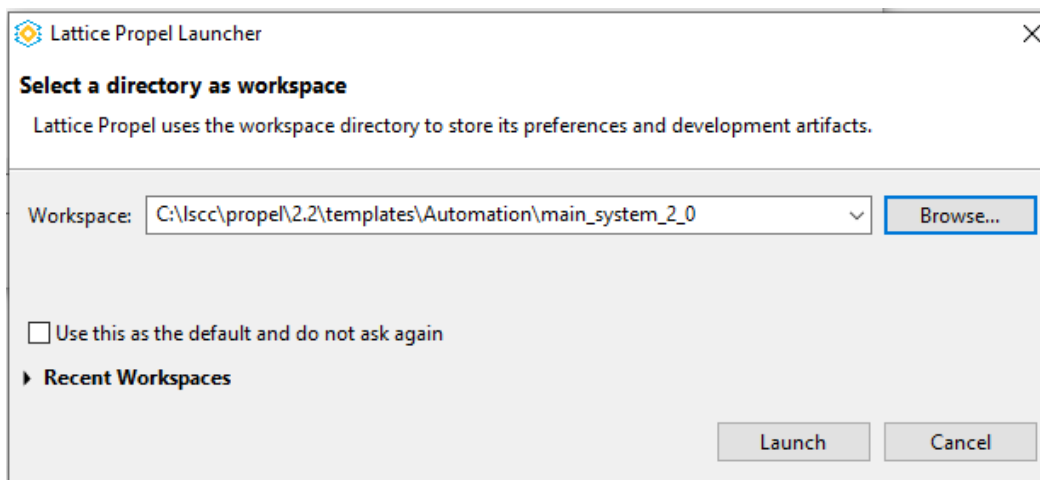


Figure 8.64: Select Directory

2. To select the workspace, browse to the template location
3. “C:\Iscc\propel\2.0\templates\Automate\main_system” by clicking on the “Browse” option as shown in below Fig. and then click on the “Launch” to launch the workspace.

There are no projects in your workspace.
To add a project:

-  [Create a new Lattice SoC Design Project](#)
-  [Create a new Lattice C/C++ Project](#)
-  [Create a project...](#)
-  [Import projects...](#)

Figure 8.65: Import Project

4. Click on the “import” to import firmware project template.
5. Select Existing Project in Workspace from General list and click on next as shown in below fig.

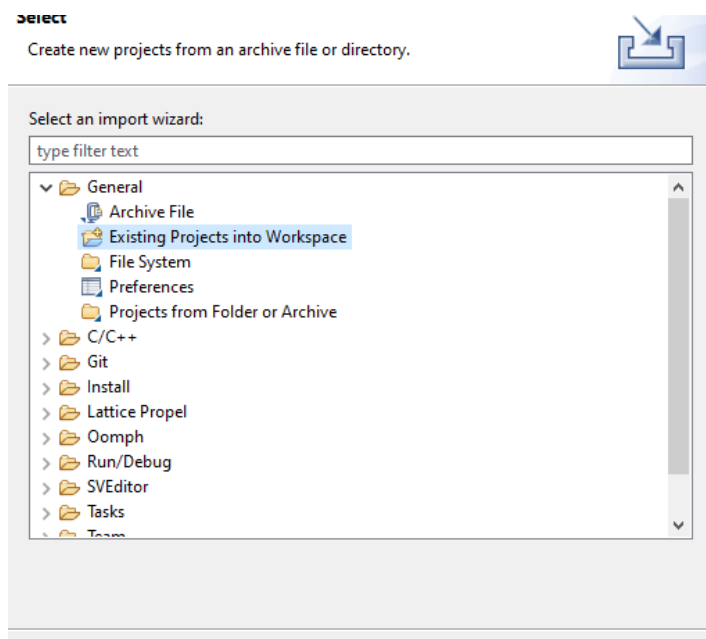


Figure 8.66: Existing Project into Workspace

6. Select the root directory and browse template location.
7. Select the project as shown in below fig. and click on finish.

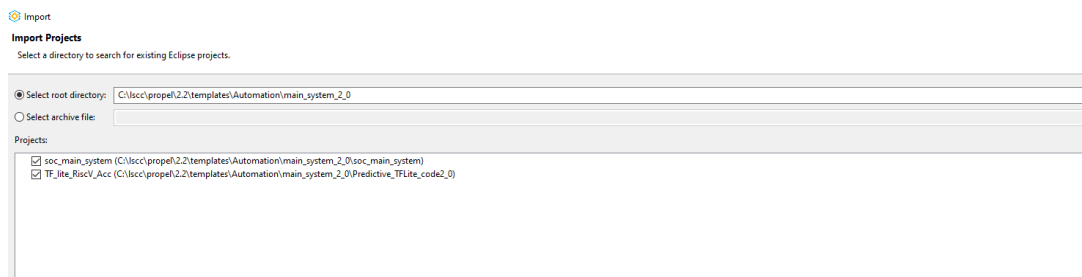


Figure 8.67: Select project

8. Right click on the firmware project folder “TF_lite_RiscV_Acc” and select the option as shown in below fig. to clean the project before building.

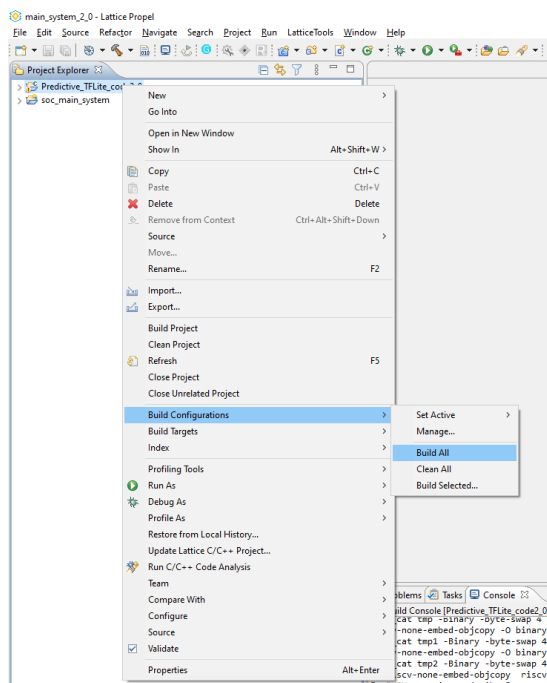


Figure 8.68: Build Configurations

9. After selecting the option as shown in above fig. observe the console and wait for the process to complete to 100%. After completion, the message shown in below fig. will appear on console.

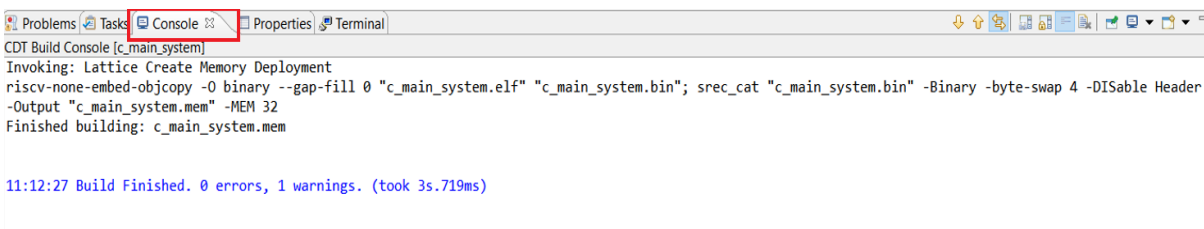


Figure 8.69: Console

10. After cleaning, right click on the “TF_lite_RiscV_Acc” and select the option as shown in below fig. to build the project.

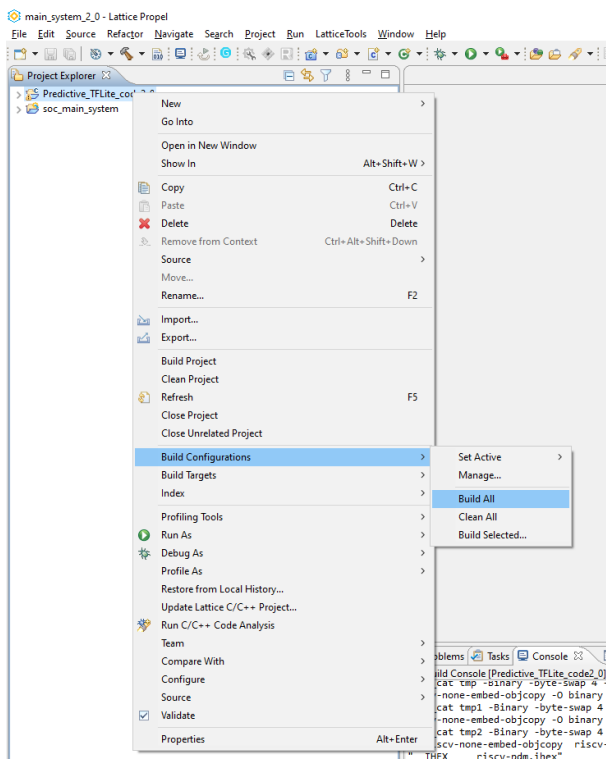


Figure 8.70: Build All

11. Wait for the process to complete to 100%. After completion, the message shown in below fig. will appear on console.



Figure 8.71: Completing Process

Appendix E. Programming the Automate Stack on Respective FLASH

E.1 Main System

This section provides the procedure for programming the SPI Flash on the CertusPro-NX Versa board for the main system. There are two different files that should be programmed into the SPI Flash. These files are programmed to the same SPI Flash, but at different addresses:

- Bitstream (FPGA SOC Design)
- Binary (RISC V Firmware)

Board Jumper Connections

Ensure that following jumpers are connected on board-

1. Pin 1 and 2 of J32 and J33 should be shorted to select UART.
2. Pin 1 and 2 of J58 should be shorted to select the 3.3 V as Flash IO.

If you are programming the Main System Board for the first time, please refer to the troubleshooting section and then come back to this section to follow further steps.

To program the SPI Flash in Radiant Programmer:

1. Connect the CertusPro-NX Versa board to the PC using a USB cable.
2. Start Radiant Programmer. In the **Getting Started** dialog box, select **Create a new blank project**.

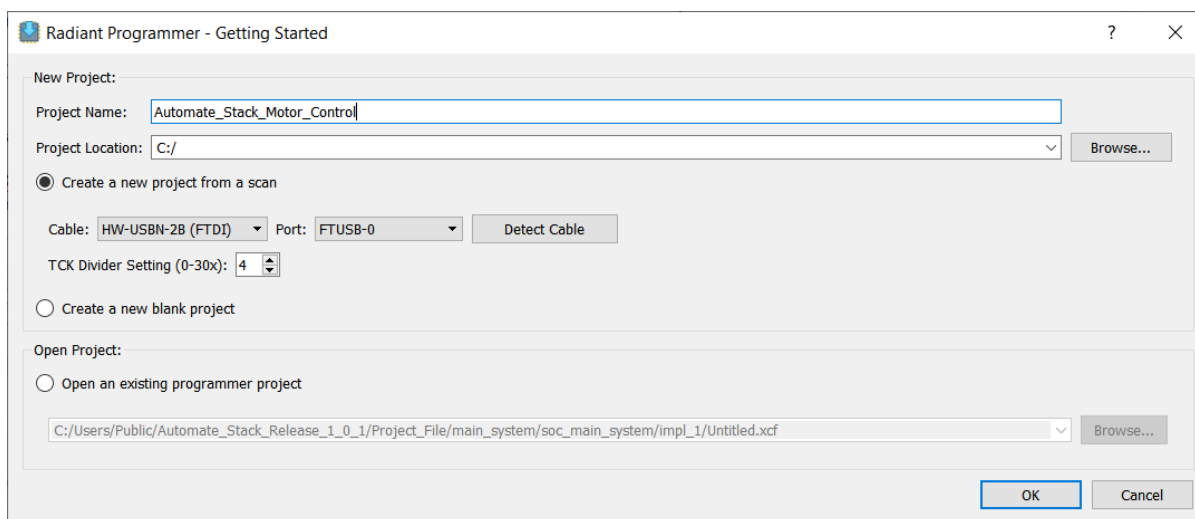


Figure 8.72: Radiant Programmer Default Screen

3. Click **OK**.

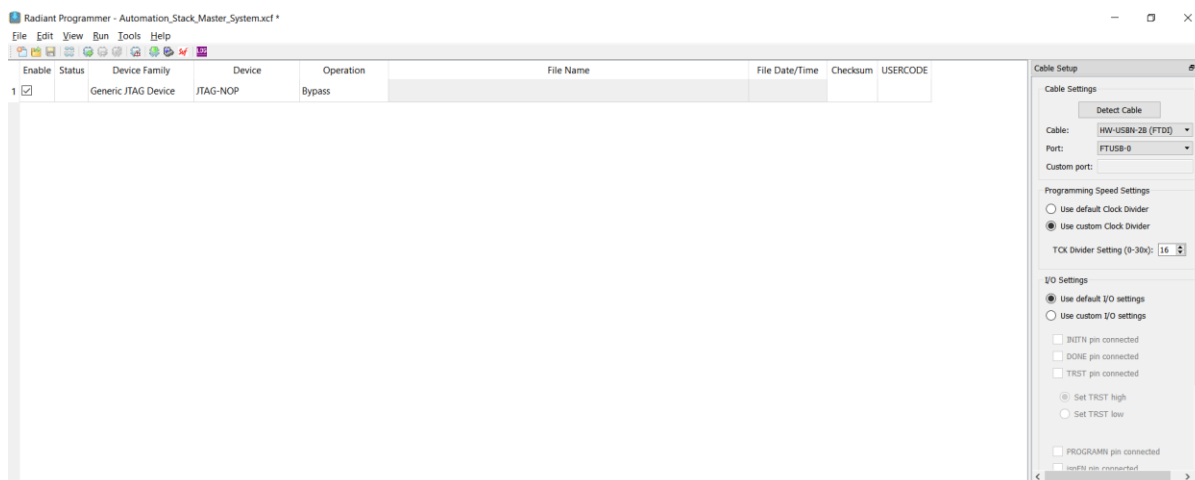


Figure 8.73: Radiant Programmer- Initial Project Window

4. In the Radiant Programmer main interface, select **LFCPNX** for **Device Family** and **LFCPNX-100** for **Device** or detect automatically as shown in below Figure

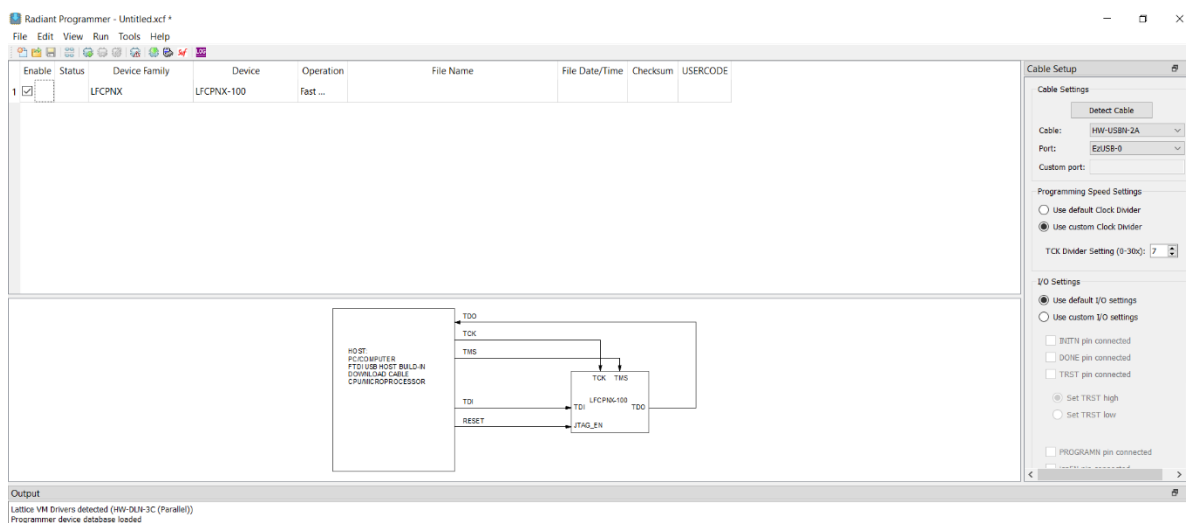


Figure 8.74: Radiant Programmer- Device Selection

5. Right-click and select **Device Properties**.

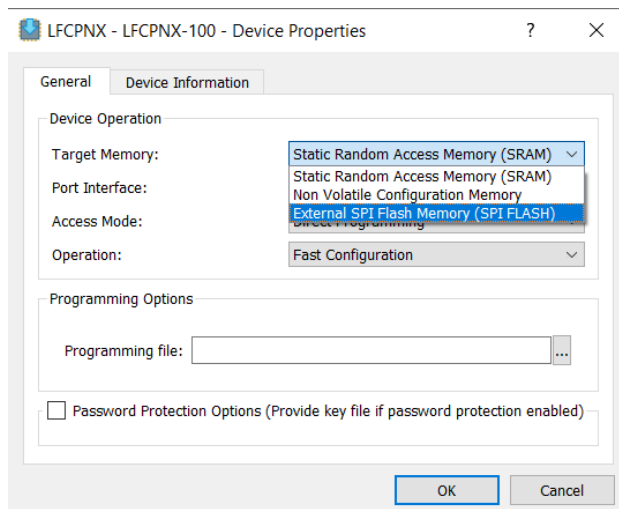


Figure 8.75: Radiant Programmer- Device Operation

6. Before programming, it is necessary to erase the flash memory. For this, Apply the settings below:

- a. Under Device Operation, select the options below:
 - Target Memory – External SPI Flash Memory (SPI FLASH)
 - Port Interface – JTAG2SPI
 - Access Mode – Direct Programming
 - Operation – Erase all
- b. Under SPI Flash Options, select the options below:
 - Family – SPI Serial Flash
 - Vendor – Macronix
 - Device – MX25L51245G
 - Package – 8-land WSON

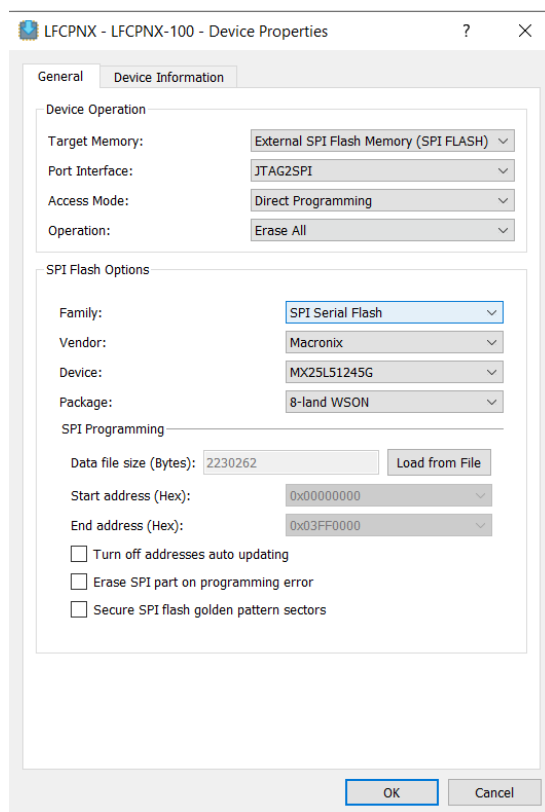


Figure 8.76: Erase all (Radiant Programmer)

7. Click OK and then click the Program Device icon or the menu item Run ->Program Device. This will erase the flash memory if any other data is already present in it.
8. After erasing the flash, power cycle the board and apply the settings below:
 - a. Under Device Operation, select the options below:
 - Target Memory – External SPI Flash Memory (SPI FLASH)
 - Port Interface – JTAG2SPI
 - Access Mode – Direct Programming
 - Operation – Erase, Program, Verify
 - b. Under SPI Flash Options, select the options below:
 - Family – SPI Serial Flash
 - Vendor – Macronix
 - Device – MX25L51245G
 - Package – 8-land WSON
9. To program the **bitstream file**, select the options as shown in below Fig.

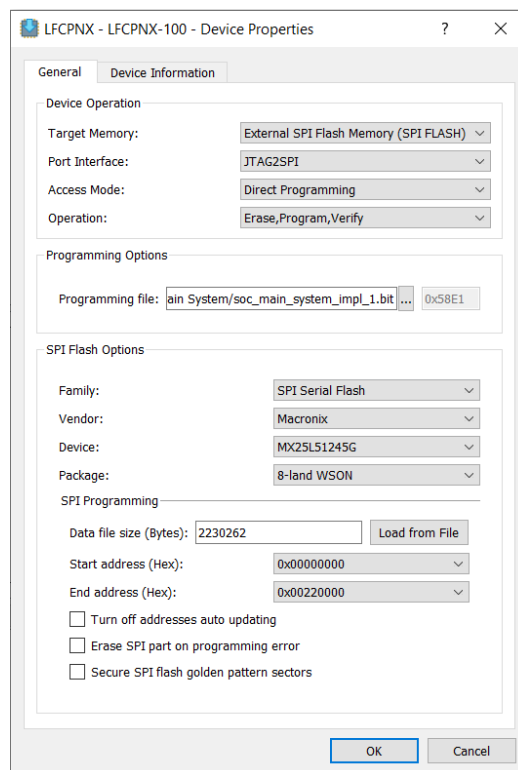


Figure 8.77: Radiant Programmer- Bitstream Flashing Settings

- a. Under **Programming Options**, select the **soc_main_system_impl_1.bit** bitstream file in Programming file.
 - b. Click **Load from File** to update the **Data file size (Bytes)** value.
 - c. Ensure that the following addresses are correct:
 - Start Address (Hex) – 0x00000000
 - End Address (Hex) – 0x00200000
10. Then click the Program Device icon or the menu item Run ->Program Device.
11. Now Power cycle the CertusPro NX Versa Board.
12. To program the **firmware**, select the options as shown in below Figure.
- a. Under **Programming Options**, select the **riscv-pdm.bin** binary file.
 - b. Ensure that the following addresses are correct:
 - Start Address (Hex) – 0x00240000
 - End Address (Hex) – 0x003D0000

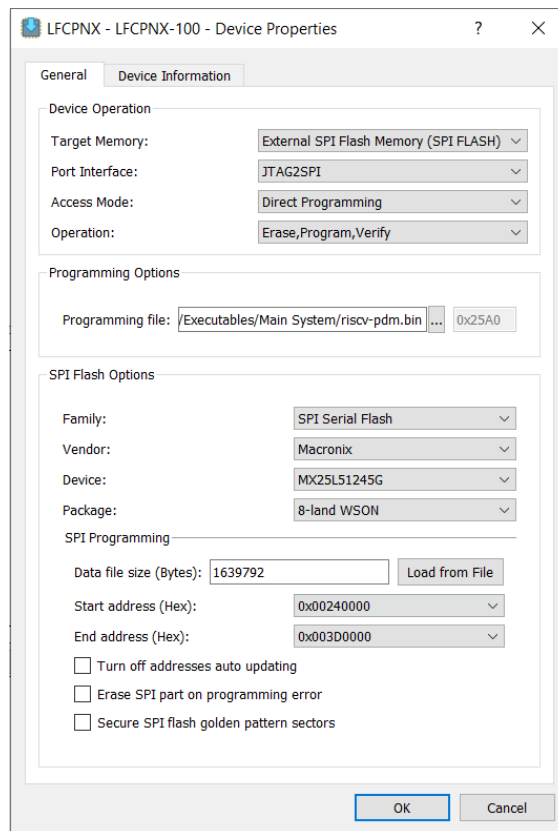


Figure 8.78: Radiant Programmer- Binary Flashing Settings

13. Then click the Program Device icon or the menu item Run ->Program Device.

E.2 Node

This section provides the procedure for programming the SPI Flash on the Certus-NX Versa board for node. There are two different files that should be programmed into the SPI Flash. These files are programmed to the same SPI Flash, but at different addresses:

- Bitstream
- Binary

Board Jumper Connections

Ensure that following jumpers are connected on board-

1. Pin 1 and 2 of JP25 and JP26 should be shorted to select UART.
2. Pin 1 and 2 of J47 should be shorted to select the 1.8 V as Flash IO.

To program the SPI Flash in Radiant Programmer:

1. Connect the Certus-NX Versa board to the PC using a USB cable.
2. Start Radiant Programmer. In the **Getting Started** dialog box, select **Create a new blank project**.

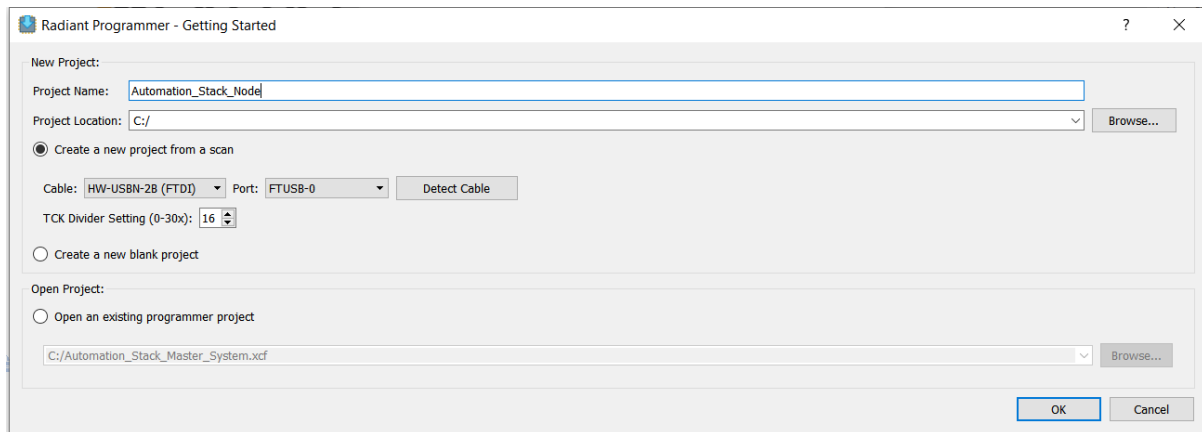


Figure 8.79: Radiant Programmer Default Screen

3. Click **OK**.

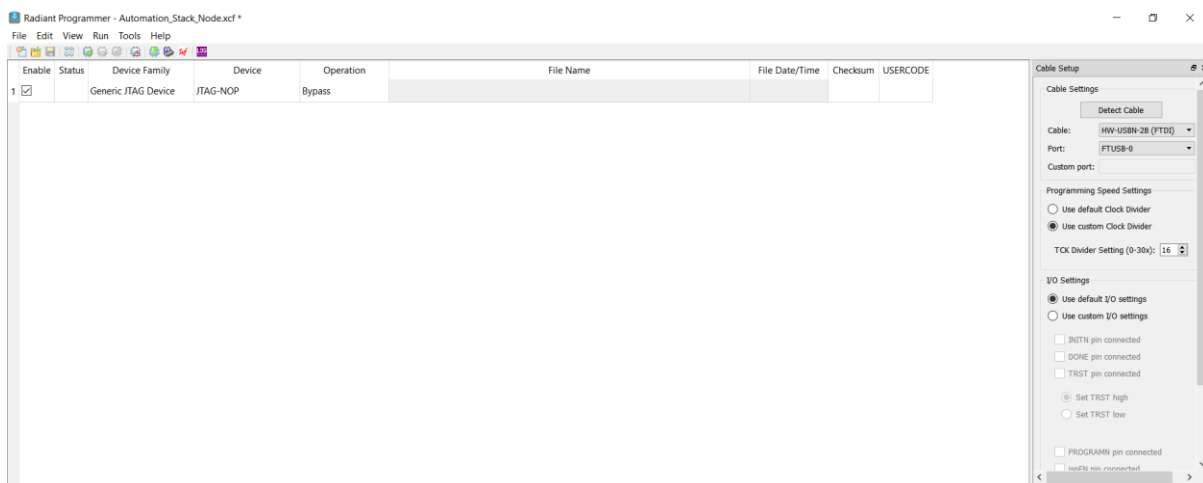


Figure 8.80: Radiant Programmer- Initial Project Window

4. In the Radiant Programmer main interface, select **LFD2NX** for **Device Family** and **LFD2NX-40** for **Device** as shown in below Figure

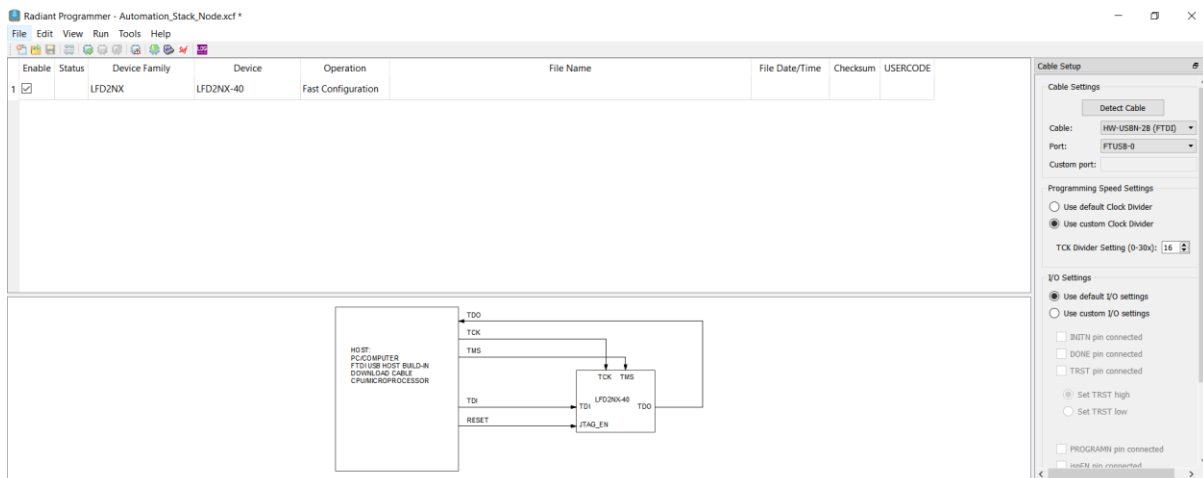


Figure 8.81: Right-click and select Device Properties.

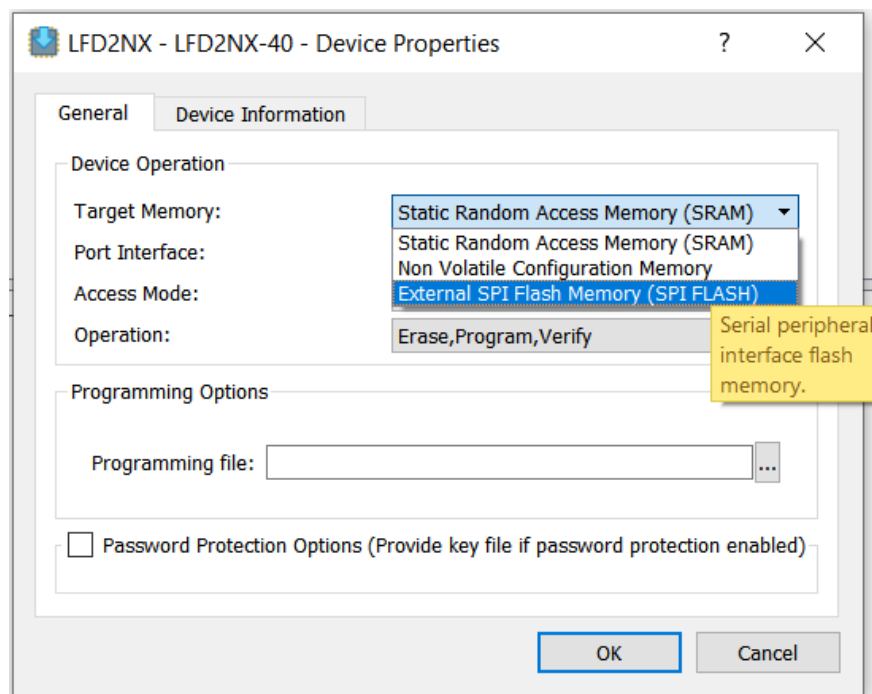


Figure 8.82: Radiant Programmer- Device Operation

5. Before programming, it is necessary to erase the flash memory. For this, Apply the settings below:
 - a. Under Device Operation, select the options below:
 - Target Memory – External SPI Flash Memory (SPI FLASH)
 - Port Interface – JTAG2SPI
 - Access Mode – Direct Programming
 - Operation – Erase all
 - b. Under SPI Flash Options, select the options below:
 - Family – SPI Serial Flash
 - Vendor – Micron
 - Device – MT25QU128
 - Package – 8-pin SOP2

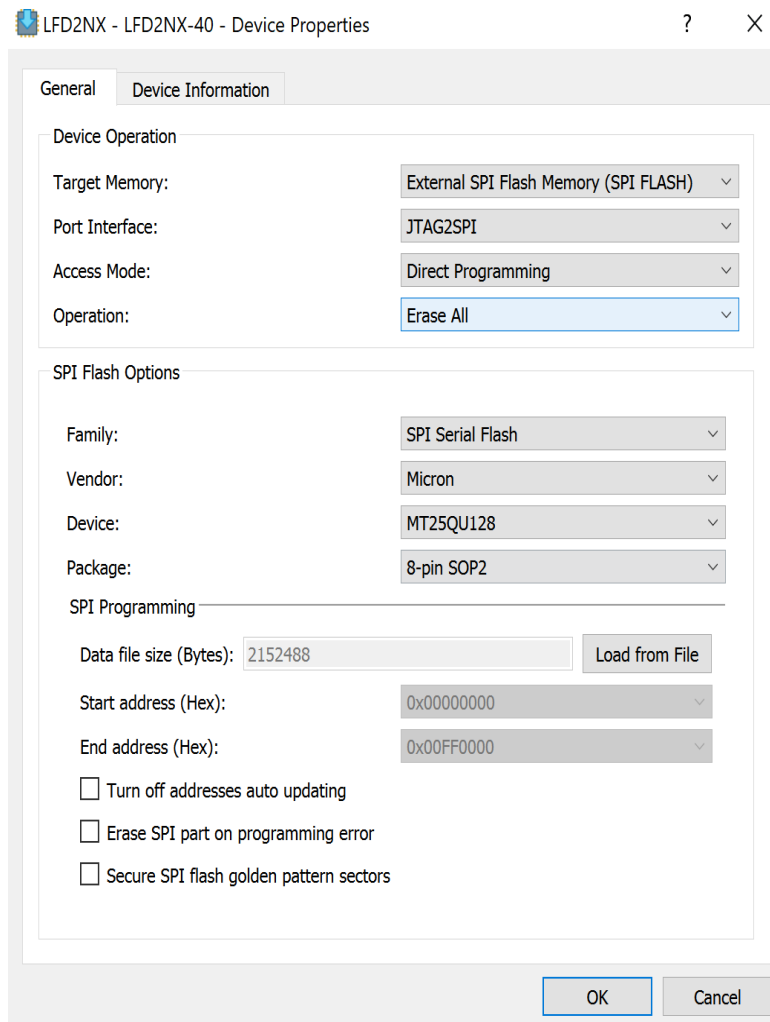


Figure 8.83: Erase all (Radiant Programmer)

6. Click OK and then click the Program Device icon or the menu item Run ->Program Device. This will erase the flash memory if any other data is already present in it.
7. After erasing the flash, power cycle the board and apply the settings below:
 - a. Under Device Operation, select the options below:
 - Target Memory – External SPI Flash Memory
 - Port Interface – SPI
 - Access Mode – Direct Programming
 - Operation – Erase, Program, Verify
 - b. Under SPI Flash Options, select the options below:
 - Family – SPI Serial Flash
 - Vendor – Micron
 - Device – MT25QU128
 - Package – 8-pin SOP2
8. To program the bitstream file, select the options as shown in below Figure

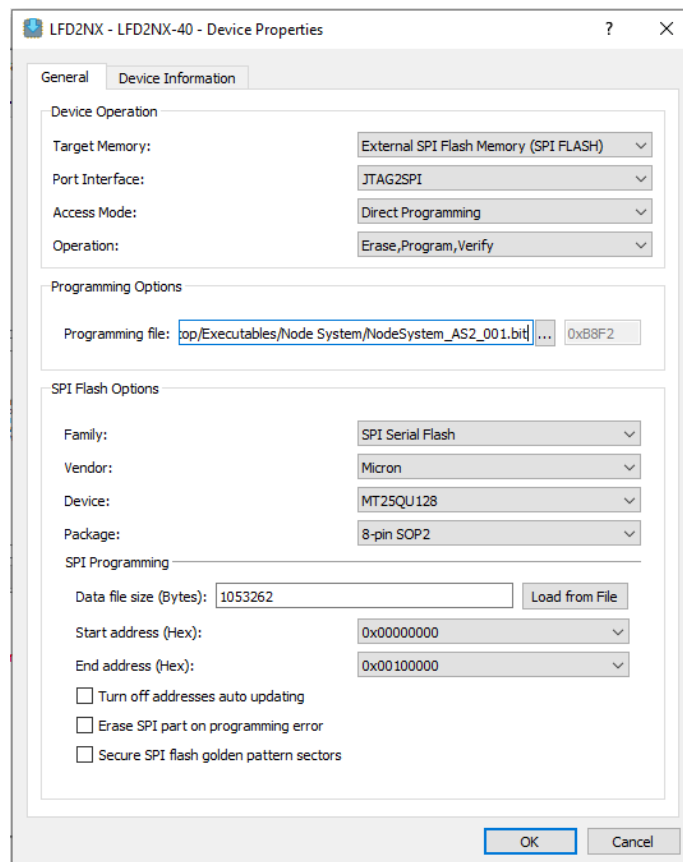


Figure 8.84: Radiant Programmer- Bit stream Flashing Settings

- a. Under **Programming Options**, select the **NodeSystem_AS2_001.bit** bitstream file in Programming file.
 - b. Click **Load from File** to update the **Data file size (Bytes)** value.
 - c. Ensure that the following addresses are correct:
 - Start Address (Hex) – 0x00000000
 - End Address (Hex) – 0x00100000
9. Then click the Program Device icon or the menu item Run ->Program Device.
10. To program the firmware, select the options as shown in below Figure.
- a. Under **Programming Options**, select the **NodeSystem_AS2_001.bin** binary file.
 - b. Ensure that the following addresses are correct:
 - Start Address (Hex) – 0x00140000
 - End Address (Hex) – 0x00240000
11. Then click the Program Device icon or the menu item Run ->Program Device.

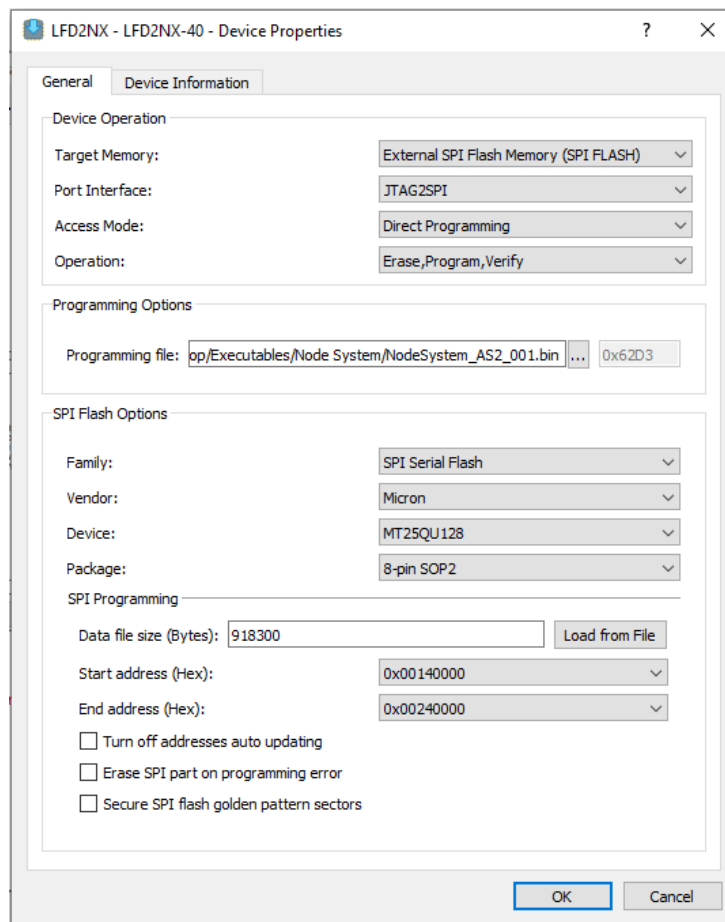


Figure 8.85: Radiant Programmer- Binary Flashing Settings

Note: After programming the Boards, do power cycle the CertusPro NX Versa board and each Certus NX versa boards and then press the system Reset button SW 3.



Figure 8.86: Reset Button SW3 of CertusPro NX Versa Board



Figure 8.87: Reset Button SW3 of Certus NX Versa Board

Appendix F. Troubleshooting (Main System Board)

If you are programming the Main System board for the first time then follow the following one time procedure to avoid the firmware booting issue. It is because, when the boards are delivered from the foundry it is programmed in the quad mode. But SPI flash Controller starts with serial mode (default) for commands and then it selects quad mode for read operation.

To program the SPI Flash in Quad mode :

1. Connect the CertusPro-NX Versa board to the PC using a USB cable.
2. Start Radiant Programmer. In the **Getting Started** dialog box, select **Create a new blank project**.

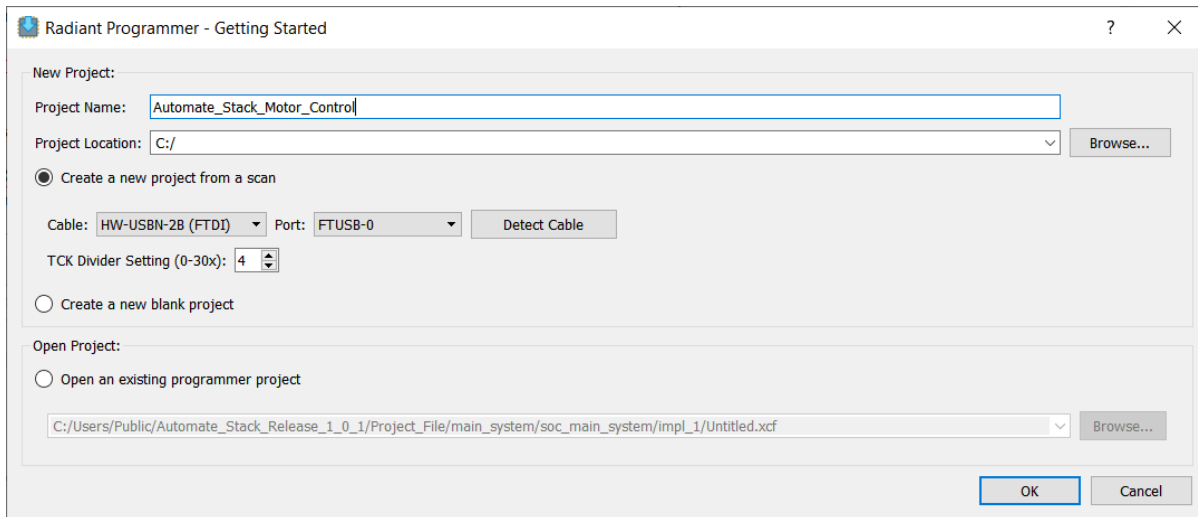


Figure 8.88: Radiant Programmer Default Screen

3. Click OK.

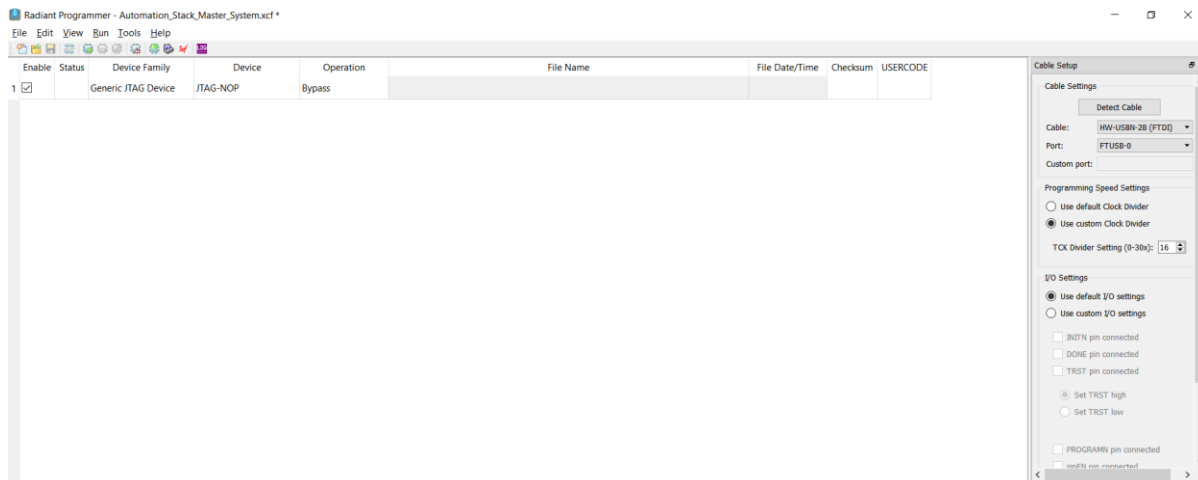


Figure 8.89: Radiant Programmer- Initial Project Window

4. In the Radiant Programmer main interface, select **LFCPNX** for **Device Family** and **LFCPNX-100** for **Device** or detect automatically as shown in below Figure.

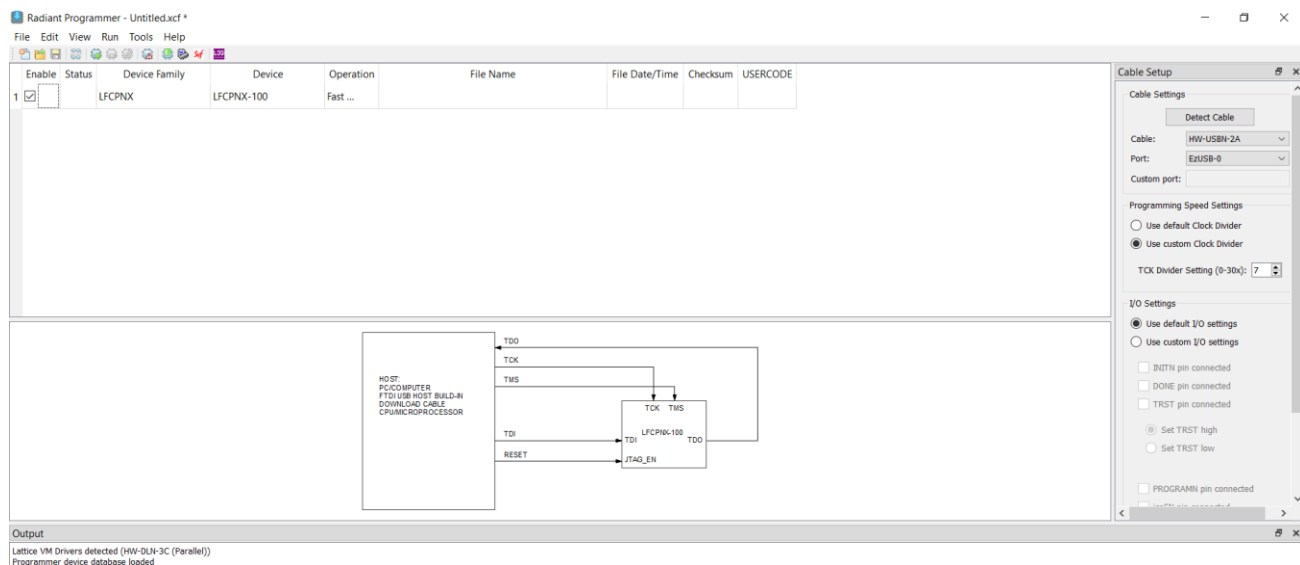


Figure 8.90: Radiant Programmer- Device Selection

5. Right-click and select **Device Properties**.

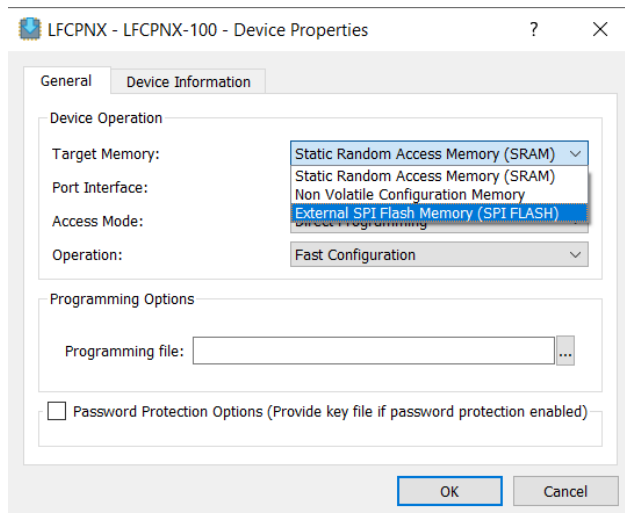


Figure 8.91: Radiant Programmer- Device Operation

6. To program the binary file, apply the settings below:
 - a. Under Device Operation, select the options below:
 - Target Memory – External SPI Flash Memory (SPI FLASH)
 - Port Interface – JTAG2SPI
 - Access Mode – Direct Programming
 - Operation – Erase, Program, Verify Quad 1
 - b. Under SPI Flash Options, select the options below:
 - Family – SPI Serial Flash
 - Vendor – Macronix
 - Device – MX25L51245G
 - Package – 8-land WSON

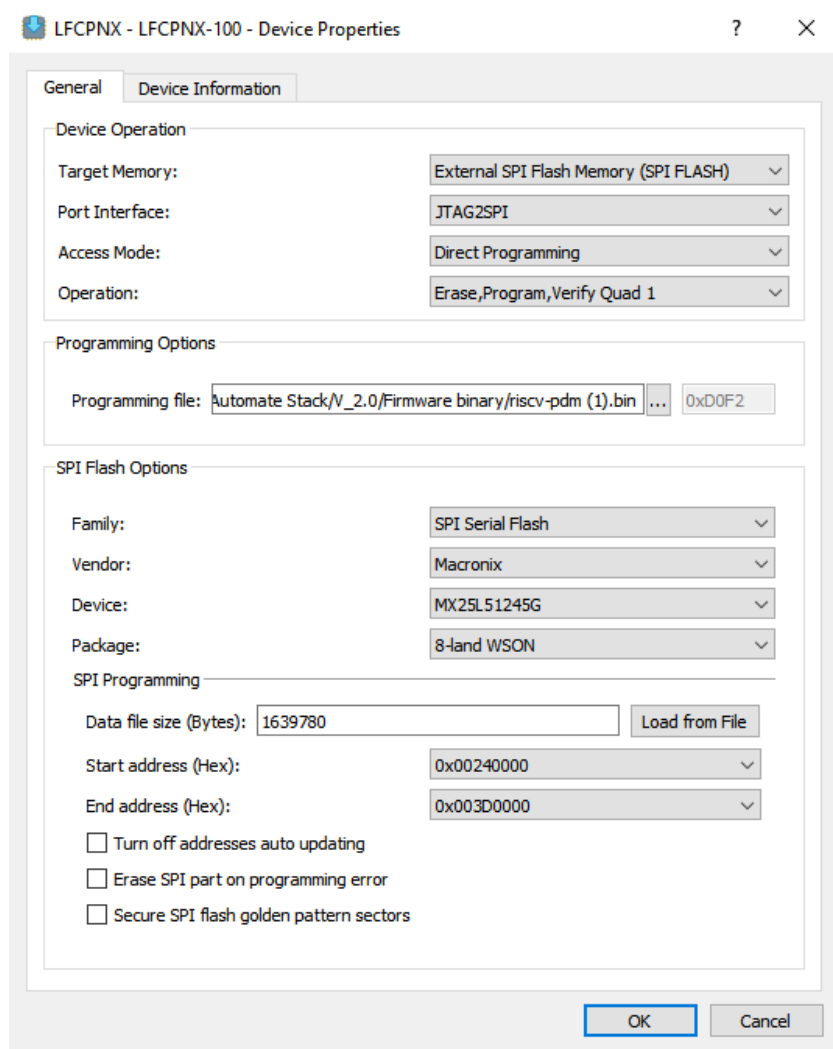


Figure 8.92: Quad Mode Programming (Radiant Programmer)

7. Under **Programming Options**, select the **riscv-pdm.bin** binary file.
 - a. Ensure that the following addresses are correct:
 - Start Address (Hex) – 0x00240000
 - End Address (Hex) – 0x003D0000
8. Click OK and then click the Program Device icon or the menu item Run ->Program Device. After that power cycle the board.
9. To program the **bitstream file**, select the options as shown in below Figure.
10. Under **Programming Options**, select the **soc_main_system_impl_1.bit** bitstream file in Programming file.

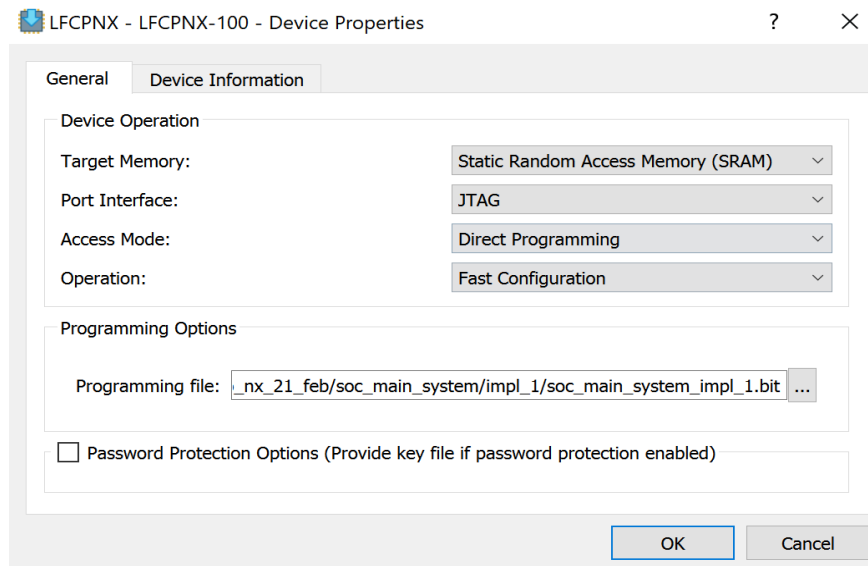


Figure 8.93: Bitfile Programming

11. Then click the Program Device icon or the menu item Run ->Program Device.

12. Now it is required to erase the SPI flash. For that, apply the settings below:

- a. Under Device Operation, select the options below:
 - Target Memory – External SPI Flash Memory (SPI FLASH)
 - Port Interface – JTAG2SPI
 - Access Mode – Direct Programming
 - Operation – Erase All
- b. Under SPI Flash Options, select the options below:
 - Family – SPI Serial Flash
 - Vendor – Macronix
 - Device – MX25L51245G
 - Package – 8-land WSON

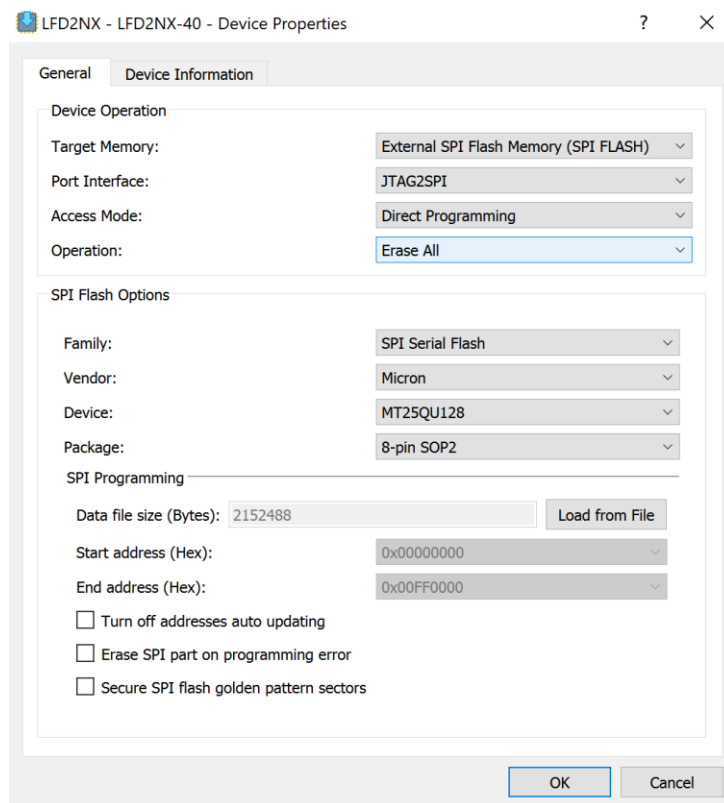


Figure 8.94: Erase all (Radiant Programmer)

13. Then click the Program Device icon or the menu item Run ->Program Device.

14. Now Power cycle the CertusPro NX Versa Board.

Note: This is a one-time process which is needed to follow for every new board.

Appendix G. Debugging Using Dock light

User does not need R-Pi board or any server or client setup to run basic demo from docklight. Docklight tool is available on web so user can download **version of tool**.

Docklight can be used as host for debugging. It supports UART protocol with various standard baud rates.

1. User needs to send each command manually to demonstrate any functionality of the system. It supports same UART packet type as GUI. Brief explanation of UART request and response packets is explained in further sections of this Appendix. **Lattice_Automate_Stack_1_1_Docklight.ptp** basic docklight script is provided with the demo package. Explanation of basic command is given in further sections of this Appendix. User can add more commands in .ptp file based on requirement.

Basic minimal list of procedures are also defined in this Appendix run basic demo from docklight.

Here are basic steps to run demonstration using docklight:

2. HW(A CertusPro-NX Board and Certus-NX Boards) should be programmed with appropriate executable files as given in Appendix E
3. Docklight tool should be installed in host PC.
4. HW connection should be proper as given in below fig.

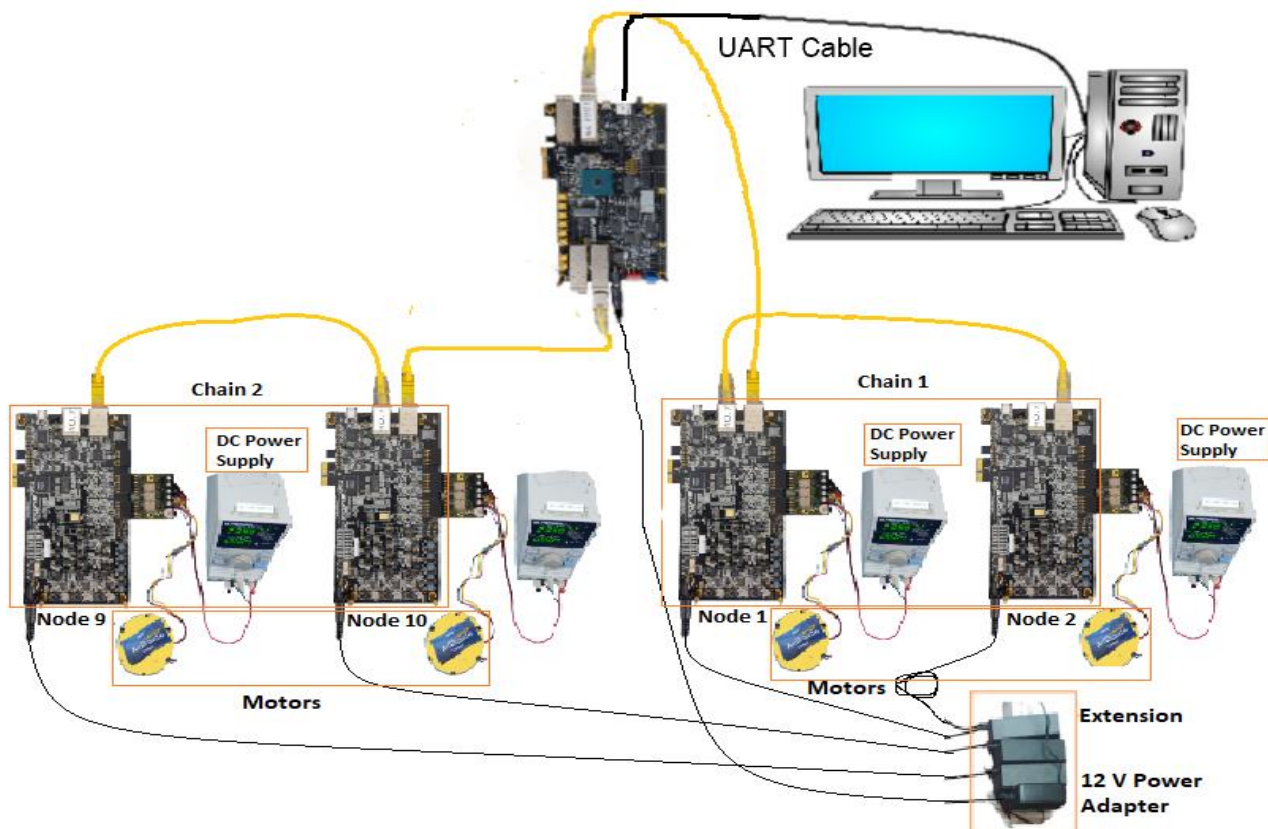


Figure 8.95: Setup for docklight

5. Open the Docklight application

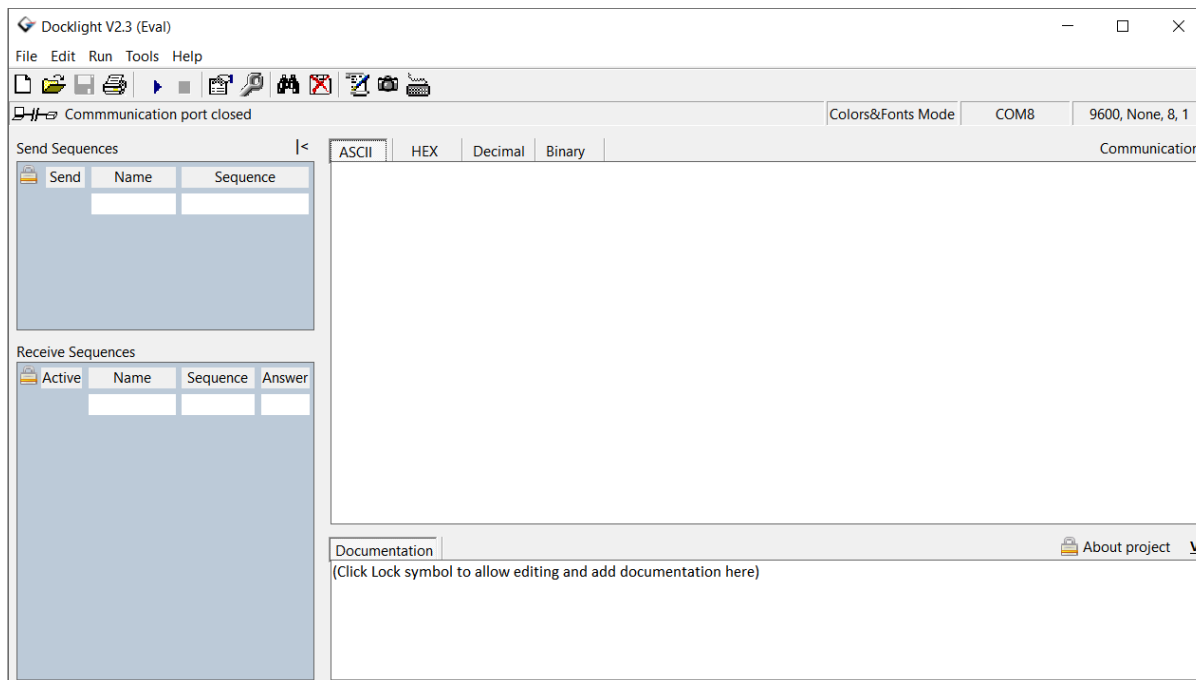
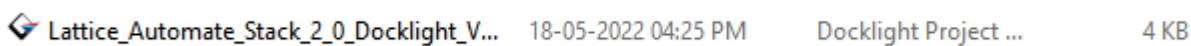


Figure 8.96: Docklight Default Screen

6. Now open the Docklight script **Lattice_Automate_Stack_2_0_Docklight.ptp**



7. Select the correct comm port and baud rate 9600.

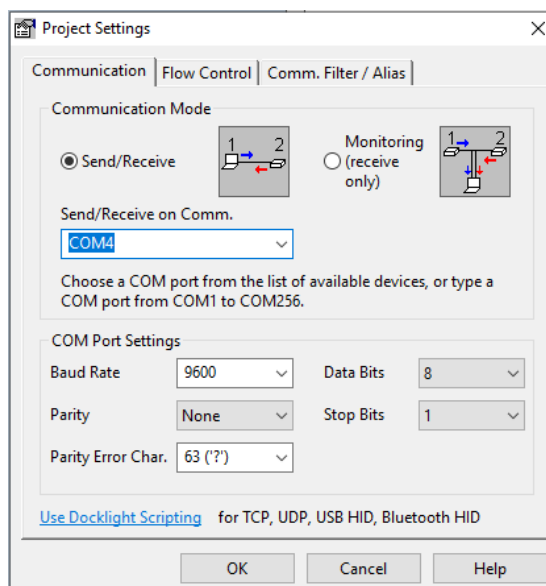


Figure 8.97: Select Com port and Baud rate

8. Click on the Run in the Docklight.

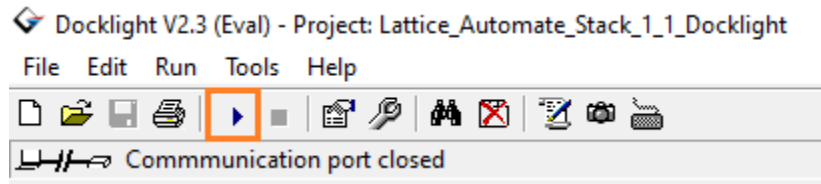


Figure 8.98: Run button

9. Press System Reset of Main System Board.



Figure 8.99: Reset the main board(CertusPro-NX Versa)

10. System initialization will appear in the ASCII tab.

```
Main System Ready!<CR><LF>
Automate Stack2.0 ver 0.1.0!<CR><LF>
Version Date 06/10/2022<CR><LF>
Multi Link for Main System detected<CR><LF>
Node num 16, Node Data length 64<CR><LF>
Broad Cast Start<CR><LF>
Org:detected_nodes 10001<CR><LF>
Chain : 1 ACTIVE NODE: 1<CR><LF>
Chain : 1 FINAL ACTIVE NODE 1<CR><LF>
PHY LINK EXPECTED: 3<CR><LF>
BEFORE: TIMER PHY LINKS MAIN and NODES 3<CR><LF>
After Node Enable: 1<CR><LF>
Chain 1 nodes reg : 1<CR><LF>
INIT_DONE<CR><LF>
Node num 16, Node Data length 64<CR><LF>
Broad Cast Start<CR><LF>
Org:detected_nodes 20101<CR><LF>
Chain : 2 ACTIVE NODE: 1<CR><LF>
Chain : 2 FINAL ACTIVE NODE 1<CR><LF>
PHY LINK EXPECTED: 3<CR><LF>
BEFORE: TIMER PHY LINKS MAIN and NODES 3<CR><LF>
After Node Enable: 1<CR><LF>
Chain 2 nodes reg : 1<CR><LF>
INIT_DONE<CR><LF>
Sys Init Done.<CR><LF>
PU Seq Start.<CR><LF>
Motor Status Value RPM: 0<CR><LF>
Motor Status Value RPM: 0<CR><LF>
PU Seq Done<CR><LF>
PDM INIT DONE. INPUT ADDRESS: e89a0<CR><LF>
<LF>
free up ether ip flag<LF>
```

Figure 8.100: System Initialization

11. User refer section **Steps to demonstrate of basic functionality** to run basic functionalities of the system.

G.1 UART Command's Description

User can refer this section for UART read/write request and response commands.

Write Request Command (TX):

CC 77 10 00 00 01 00 00 18 70 19 00 01 00 00 00

CMD	LEN	Read	write	Address	Data
CC 77 : Magic No. For Command packet					
CMD Len: 2 byte for packet length					
Data Write: 0x00000100					
Data Read: 0x00000200					
Address: Address to Write or Read					
Data: Data will be present from byte position 12(index 12)					
Write Response Command (RX):					
AA	88	0C	00	00	01 00 00 00 00 00 00 00
Packet Length Response Packet Error / No Error					
AA 88 : Magic Word					
Packet Length: 2 byte of packet length					
Response Packet Code:					
Data Write Response: 0x00000100					
Data Read Response: 0x00000200					
Error/ No Error :					
No Error: 0x00000000					
Error: 0x00000001					
Read Request Command (TX):					
CC	77	0C	00	00	02 00 00 18 00 19 00
				CMD LEN	Read write Address
CC 77 : Magic No. For Command packet					
CMD Len: 2 byte for packet length					
Data Write: 0x00000100					
Data Read: 0x00000200					
Address: Address to Write or Read					
Data: Data will be present from byte position 12 (index 12)					
Read Response Command (RX):					
AA	88	10	00	00	02 00 00 00 00 00 00 00 00 00 00 00
Packet Length Response Packet Error / No Error Data					
AA 88 : Magic Word					
Packet Length: 2 byte of packet length					
Response Packet Code:					
Data Write Response: 0x00000100					
Data Read Response: 0x00000200					
Error/ No Error :					
No Error: 0x00000000					
Error: 0x00000001					

G.2 Commands

The configuration settings for the motor will be applied by the user to control the motor.

1. Main system commands - to enable different mode

- A. **CHAIN_1_SELECT** : Chain 1 of nodes select, set by the user.
Command: CC 77 10 00 00 01 00 00 18 70 19 00 01 00 00 00
- B. **CHAIN_2_SELECT** : Chain 2 of nodes select, set by the user.
Command: CC 77 10 00 00 01 00 00 18 70 19 00 02 00 00 00
- C. **CHAIN_1_NODE_SELECT** : Node select from chain 1, set by the user.
Command: CC 77 10 00 00 01 00 00 04 70 19 00 01 00 00 00
Note: This Command is only for Chain 1 and Node 1
- D. **CHAIN_2_NODE_SELECT** : Node select from chain 2, set by the user.
Command: CC 77 10 00 00 01 00 00 1C 70 19 00 01 00 00 00
- E. **ALL_CHAIN_TX_ENABLE** : Both chain of nodes select, set by the user.
Command: CC 77 10 00 00 01 00 00 20 70 19 00 01 00 00 00
- F. **SINGLE_CHAIN_TX_ENABLE** : one by one chain select mode enable, set by the user.
Command: CC 77 10 00 00 01 00 00 20 70 19 00 00 00 00 00
- G. **PDM_NORMAL_PACKET_ENABLE** : PDM normal packet enable, set by the user.
Command: CC 77 10 00 00 01 00 00 24 70 19 00 00 00 00 00
- H. **PDM_EXTENDED_PACKET_ENABLE** : PDM extended packet enable, set by the user.
Command: CC 77 10 00 00 01 00 00 24 70 19 00 01 00 00 00
- I. **MOTOR_STATUS_READ_MAIN_SYSTEM** : Motor status read from main system register bank, set by the user.
Command: CC 77 0C 00 00 02 00 00 24 70 19 00 01 82 10 00

2. MOTOR Config Command

- A. **NODE_LED_ON** : Node LED ON configuration , set by the user.
Command: CC 77 10 00 00 01 00 00 54 40 18 00 00 00 00 00
- B. **NODE_LED_OFF** : Node LED OFF configuration , set by the user.
Command: CC 77 10 00 00 01 00 00 54 40 18 00 FF FF FF FF
- C. **MIN_RPM** : Minimum rpm limit of motor, set by the user.
Command: CC 77 10 00 00 01 00 00 00 40 18 00 78 00 04 96
- D. **MAX_RPM** : Maximum rpm limit of motor, set by the user.
Command: CC 77 10 00 00 01 00 00 04 40 18 00 10 0E 00 00
- E. **RPM_PI_KI** : Configuration of motor, set by the user.
Command: CC 77 10 00 00 01 00 00 08 40 18 00 10 00 40 00
- F. **RPM_PI_KP** : Configuration of motor, set by the user.
Command: CC 77 10 00 00 01 00 00 0C 40 18 00 80 00 00 08
- G. **TARGET_RPM 120** : Target rpm on which the motor is to run, set by the user.
Command: CC 77 10 00 00 01 00 00 1C 40 18 00 78 00 00 00
- H. **TARGET_RPM 500** : Target rpm on which the motor is to run, set by the user.
Command: CC 77 10 00 00 01 00 00 1C 40 18 00 78 00 00 00
- I. **TARGET_RPM 1000** : Target rpm on which the motor is to run, set by the user.
Command: CC 77 10 00 00 01 00 00 1C 40 18 00 78 00 00 00
- J. **TARGET_RPM 1500** : Target rpm on which the motor is to run, set by the user.
Command: CC 77 10 00 00 01 00 00 1C 40 18 00 78 00 00 00

- K. **TARGET_RPM 1800** : Target rpm on which the motor is to run, set by the user.
Command: CC 77 10 00 00 01 00 00 1C 40 18 00 78 00 00 00
- L. **PI_RESET** : Configuration of motor, set by the user.
Command: CC 77 10 00 00 01 00 00 18 40 18 00 00 00 00 01
- M. **STROBE_FUNCTION** : Strobe function configuration, set by the user.
Command: CC 77 10 00 00 01 00 00 18 40 18 FF 00 00 00 00
- N. **MOTOR_POWER (38%)** : Motor power configuration , set by the user.
Command: CC 77 10 00 00 01 00 00 14 40 18 00 80 01 00 00
- O. **MOTOR_POWER(10%)** : Motor power configuration , set by the user.
Command: CC 77 10 00 00 01 00 00 14 40 18 00 80 01 00 00
- P. **MOTOR_START** : Motor start configuration of motor, set by the user.
Command: CC 77 10 00 00 01 00 00 18 40 18 00 00 00 00 10
- Q. **MOTOR_STOP** : Motor stop configuration of motor , set by the user.
Command: CC 77 10 00 00 01 00 00 18 40 18 00 00 00 00 12
- R. **MOTOR_POWER_OFF** : Motor power off configuration of motor, set by the user.
Command: CC 77 10 00 00 01 00 00 18 40 18 00 00 00 00 00
- S. **PDM_OFFSET** : PDM offset configuration , set by the user.
Command: CC 77 10 00 00 01 00 00 30 40 18 00 40 0F C8 00
- T. **PDM_PEAK_DETECT** : PDM peak detect configuration , set by the user.
Command: CC 77 10 00 00 01 00 00 30 40 18 00 02 0F C8 00
- U. **PDM_START** : PDM start configuration, set by the user.
Command: CC 77 10 00 00 01 00 00 30 40 18 00 09 0F C8 00

Motor Status Command

- A. **PDM_READY_CHECK** : PDM Ready check configuration , set by the user.
Command: CC 77 0C 00 00 02 00 00 38 40 18 00 40 0F C8 00
- B. **PDM_LOCK_CHECK** : PDM lock check configuration , set by the user.
Command: CC 77 0C 00 00 02 00 00 2C 40 18 00 40 0F C8 00

PDM Data Fetched Command

- A. **PDM_DATA_FETCHED** : PDM data fetched configuration , set by the user.
Command: CC 77 10 00 00 01 00 00 0C 70 19 00 00 00 00 00
- B. **PDM Health Status Command Health_Status**: PDM data fetched configuration, set by the user.
Command: CC 77 10 00 00 02 00 00 70 70 19 00

00 - 1st node

01 - 2nd node

02 - 3rd node and so on

Steps to demonstrate of basic functionality.

1. Basic Communication test
 - a. Select the Chain.
 - b. Select the node for Motor Control
 - c. Send LED ON and OFF commands.
2. Both Chain Communication

- d. Select any one chain number
- e. Select the nodes for selected chain.
- f. Send the command for selected node.
- g. Select another chain number
- h. Select node for another chain.
- i. Send command for another chain.
- j. Command will be send to all the both chain

3. Motor Configuration Test

- a. Select the Chain.
- b. Select the node for Motor Control
- c. Send the Motor config command.

4. Motor Status Test

- a. Select the Chain.
- b. Select the node for Motor Control
- c. Send the motor Status command.

5. Motor Start Test

- a. Select the Chain.
- b. Select the node for Motor Control
- c. Send the PDM Offset command.
- d. Send the Min RPM command.
- e. Send the MAX RPM command.
- f. Send the PI KI command.
- g. Send the PI KP command.
- h. Send the Target 120 command.
- i. Send the PI Reset command.
- j. Send the Strobe function command.
- k. Send the Motor Power 10 % command.
- l. Send the Motor Start command
- m. Send the strobe function command.
- n. Send the Motor Stop command.
- o. Send the strobe function command.

6. PDM Data Fetched Test

PDM Data can be fetched from one node at a time.

- a. Select the Chain.
- b. Select the node for Motor Control
- c. Send all the motor start test commands till step “m”.
- d. Send the Motor Power 38% command.
- e. Send the Target RPM 500 command then Strobe function command.

Note: Match the RPM Lock Status

- f. Send the Target RPM 1000 command then Strobe function command.

Note: Match the RPM Lock Status

- g. Send the Target RPM 1500 command then Strobe function command.

Note: Match the RPM Lock Status

h. Send the Target RPM 1800 command then Strobe function command.

Note: Match the RPM Lock Status

i. Send the PDM Peak Detect command then Strobe Function command.

j. Send the PDM Start command then Strobe Function command.

k. Send PDM Ready Check command and check the status

l. Send the PDM call node command.

m. Wait for the PDM data response on docklight

Then bit for the health status, it will take around 2min, do not send any command before getting health status response.

n. Send the Motor Stop command.

Appendix H. OpcUA Modeler (Server End)

H.1 OpcUA Modeler Installation

1. Open the terminal in Raspberry pi.
2. Write the Command **sudo apt install python3.6**
3. Write the Command **sudo apt-get install python3-pyqt5**
4. Write the Command **pip3 install opcua-modeler**
5. Now go to the directory using this command **cd /usr/local/bin**
6. For opening the OPCUA Modeler **./opcua-modeler**
7. OPCUA Modeler will open.

Note: PyQT 5 is required.

Note: Python 3.6+ is required.

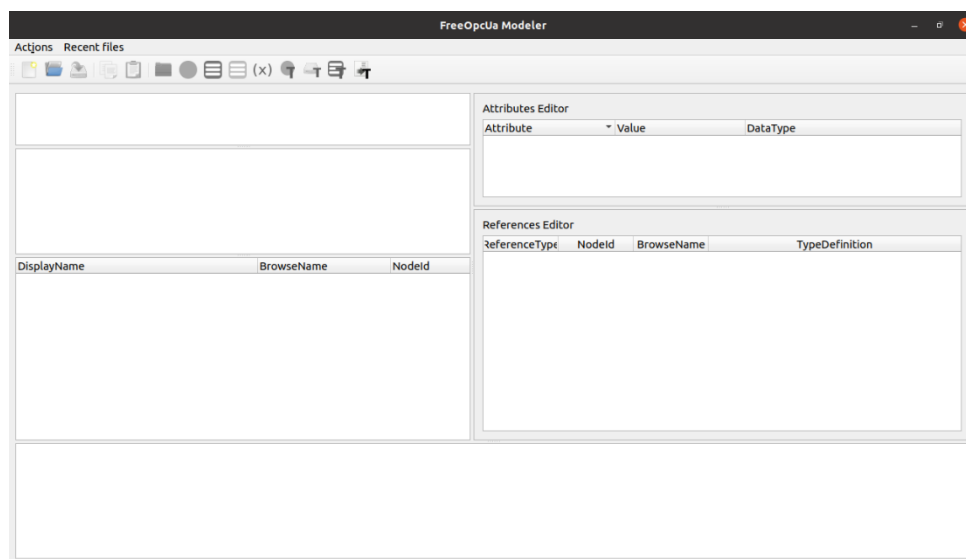


Figure 8.101: OPCUA Modeler

H.2 OPCUA_Modeler : For creating a new Xml (Information model)

1. After OPCUA Modeler installation process start from the below steps.
2. Now go to the **Actions** and Create a **New Model**.
3. Click on the below NamespaceArray and create a new **add space**.
4. Gui contain default information model like root, objects, types, views.

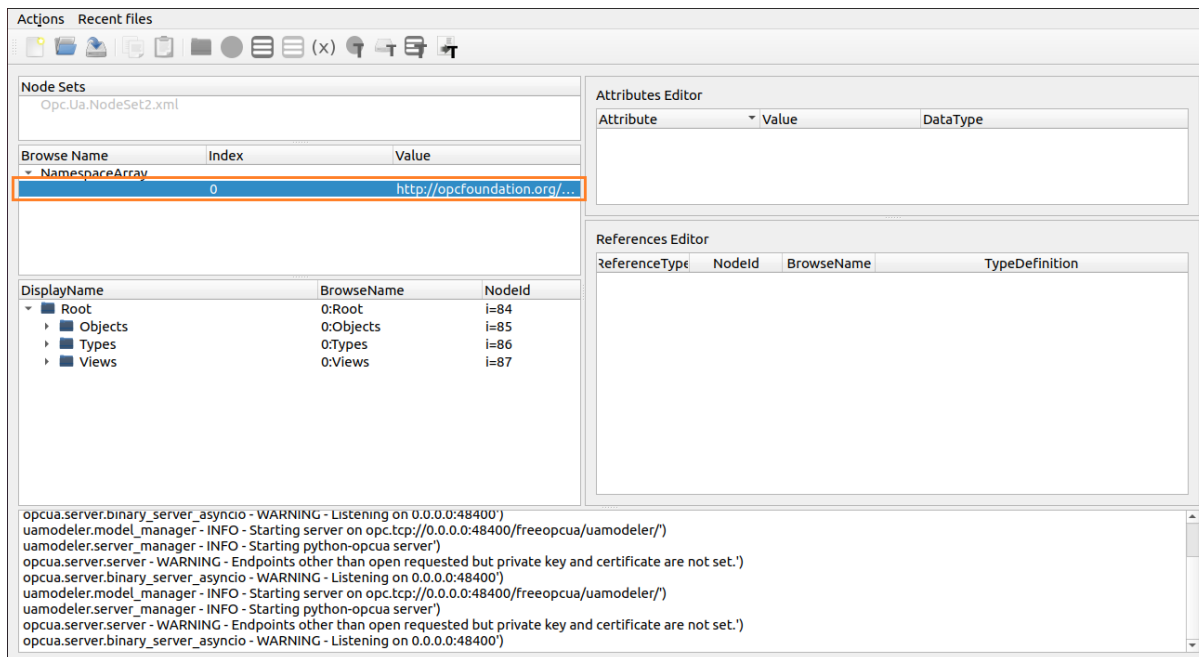


Figure 8.102: Create OpcUA Modeler xml file

- Give the name as **MainSystem** in add space.

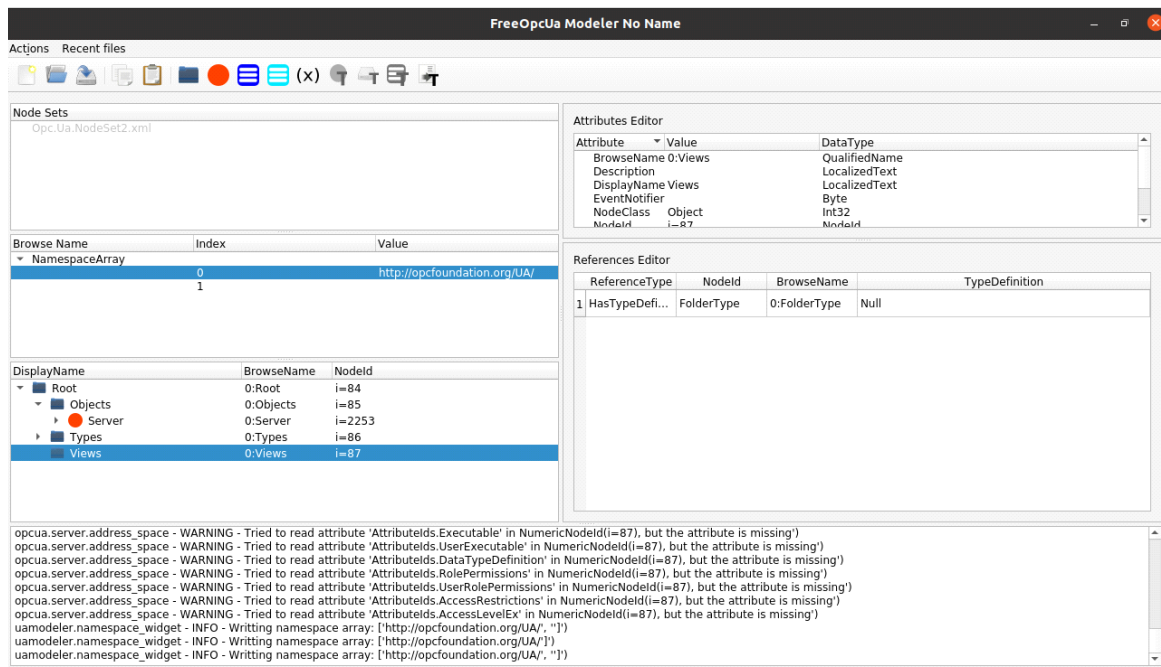


Figure 8.103: Create Main system

- After add space, it will show in Value.
- NamespaceArray are specified as variables, method and structure types information.

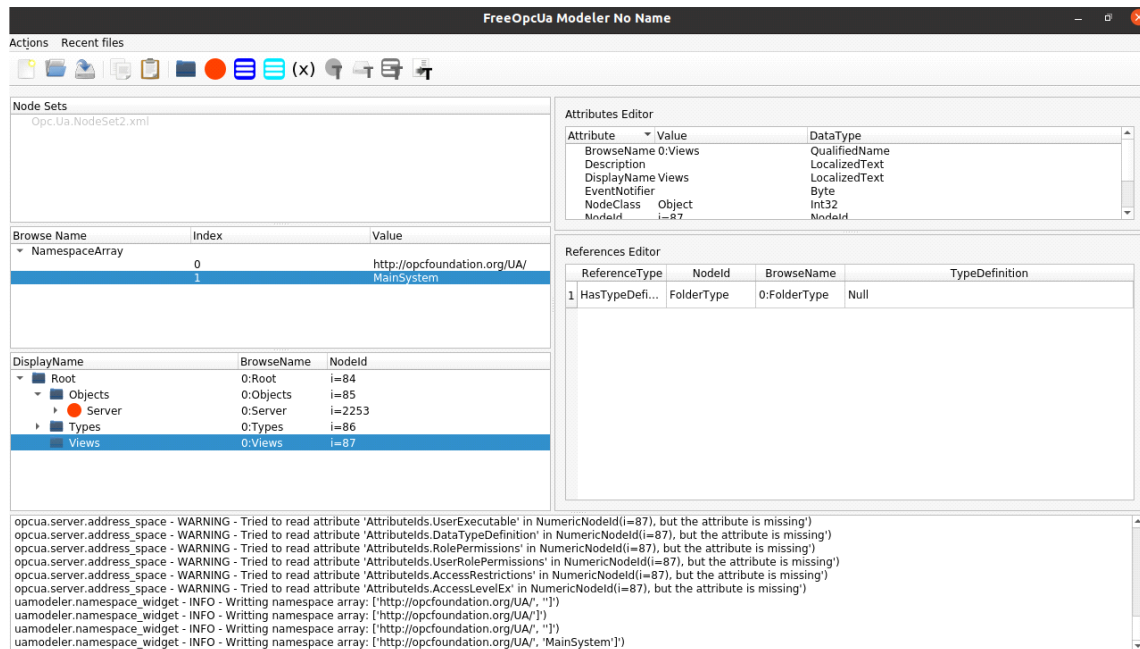


Figure 8.104: Add namespacearray in Browse

8. Press the right click on the Objects and make a new object and give the name: node1
9. In the object, specified name space as Mainsystem and default Object structure type as BaseObjectType.
10. Now check the Auto Nodeid then it will detect automatically.

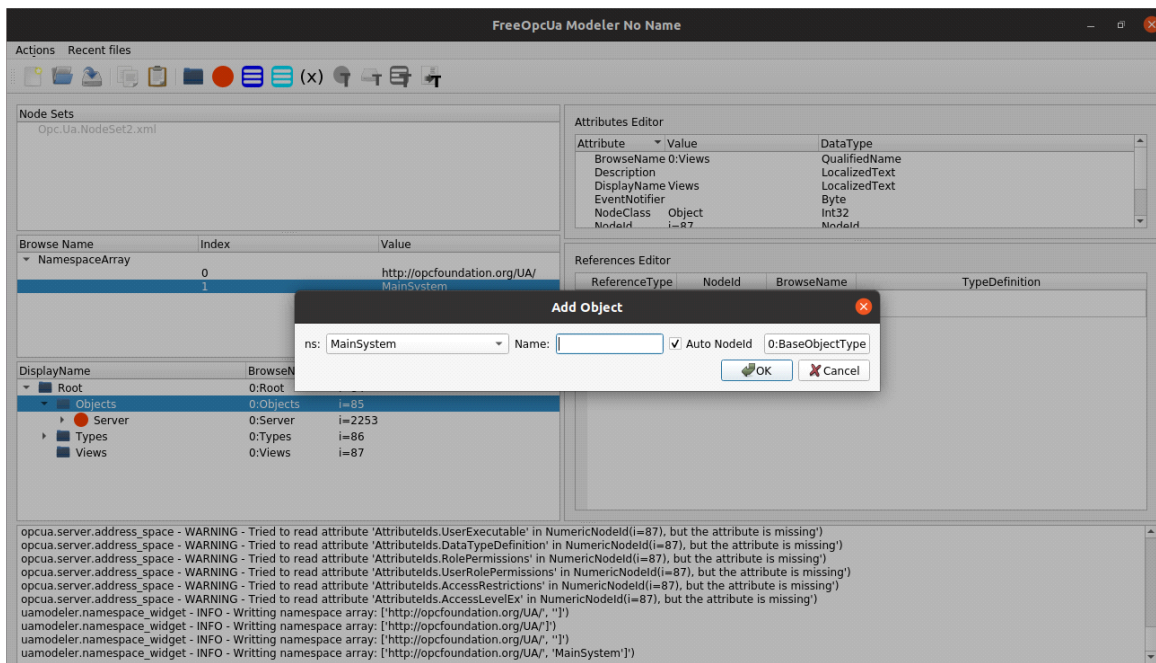


Figure 8.105: Object name

11. If want to change something else then change manually.

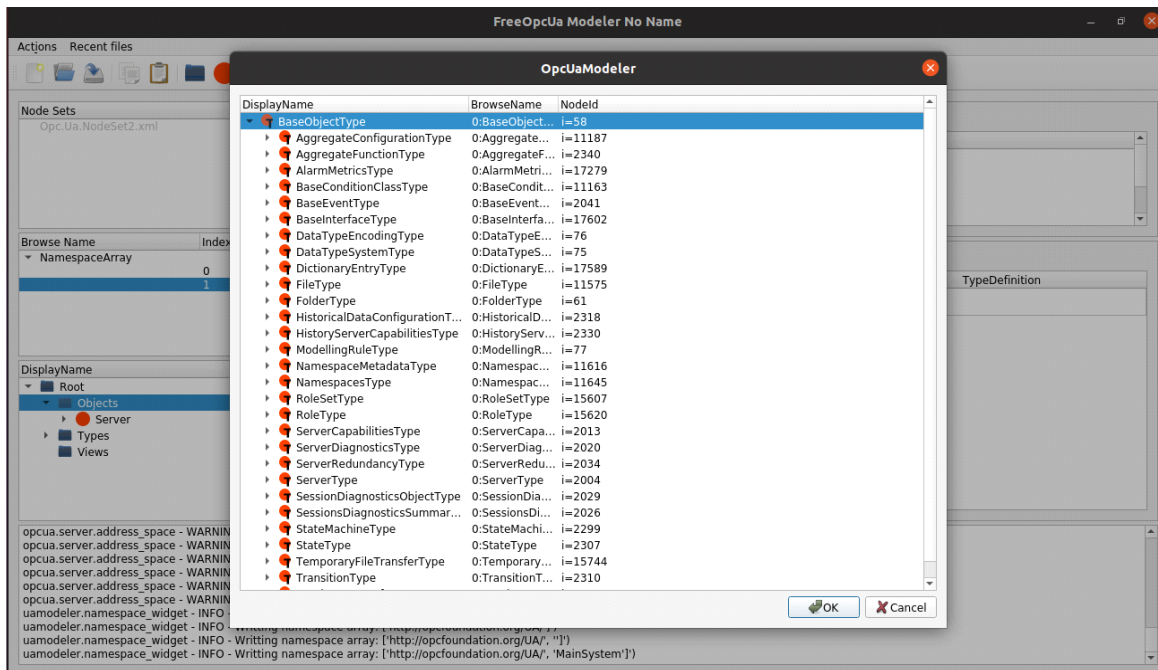


Figure 8.106: manual selection of node id

12. Now click OK.

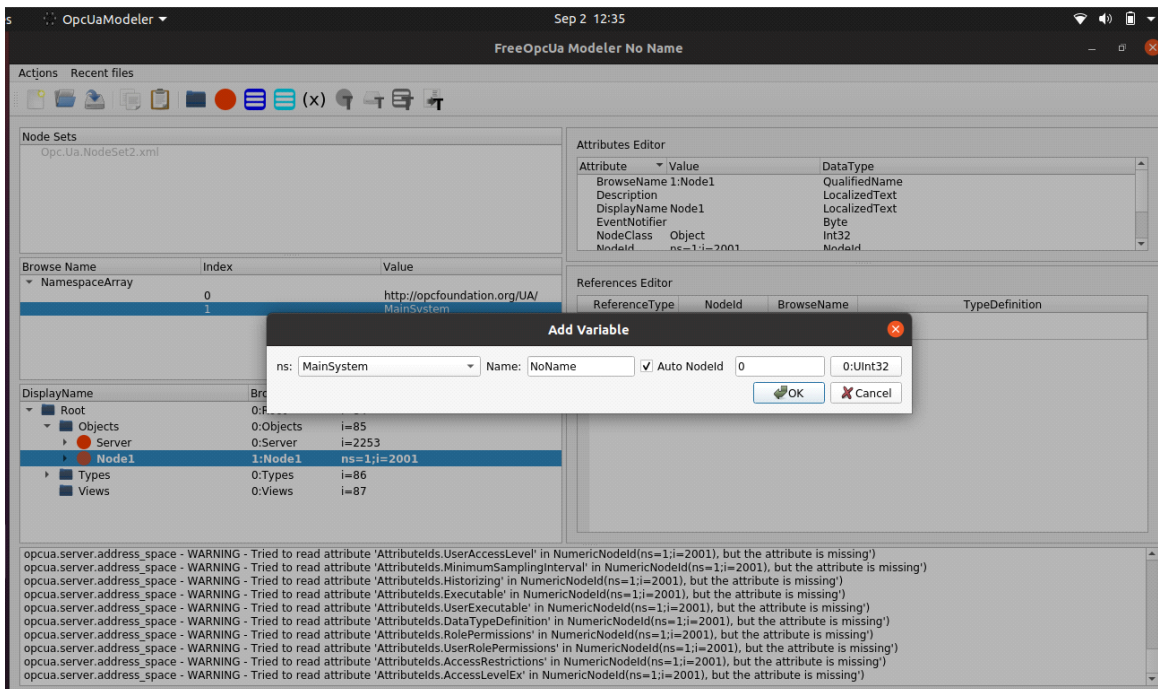


Figure 8.107: Add Variable

13. Now define a variable in Node1 objects then press the right click on Node1 and specify namespace, variable name (TargetRPM) and data type like integer, unsigned integer or boolean.

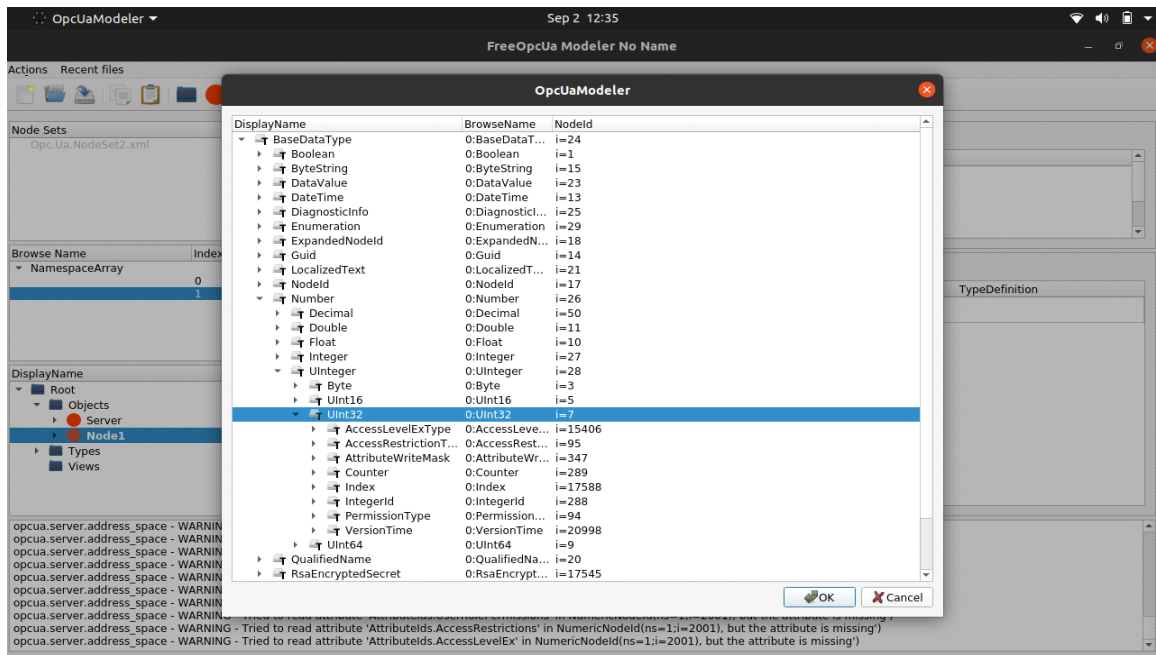


Figure 8.108: Define Variable

14. Now add a method name as **MotorStart** and also define input: Arg Name as **motorInput** and output: Arg Name as **motorOutput**.
15. Click **Ok**.
16. Input arguments send by a client side and server will provide output on the client side.

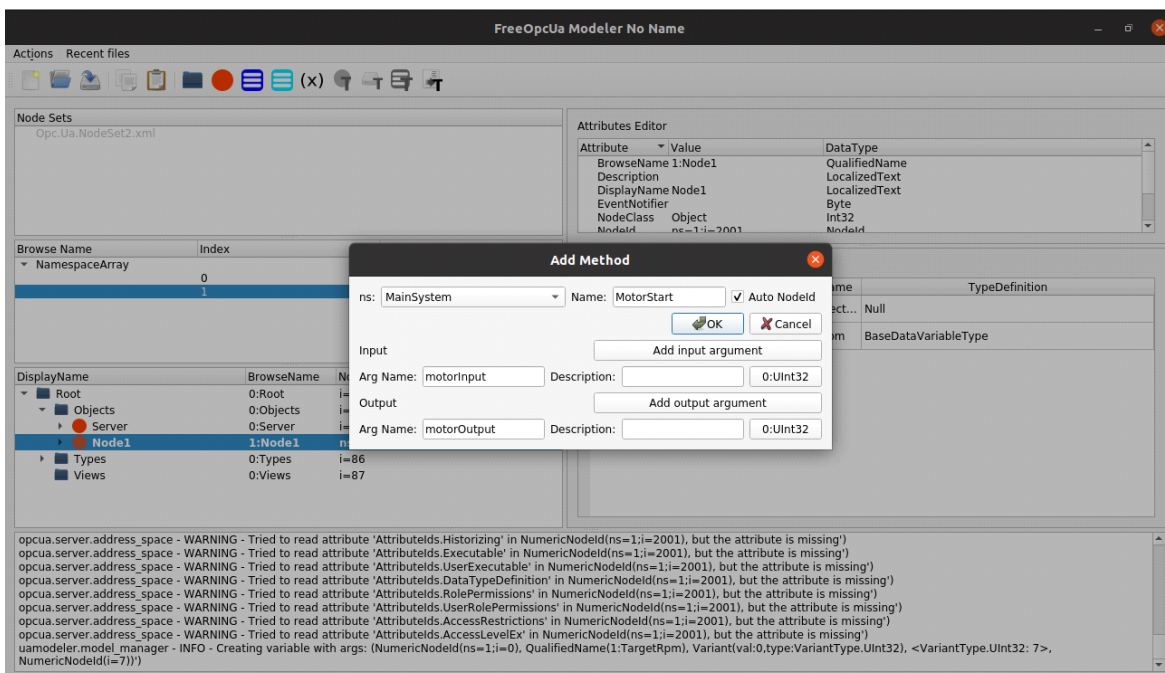


Figure 8.109: Add Method

17. Node1 have structure tree of information model.

18. These steps is to make a new information model for using opcua-modeler.

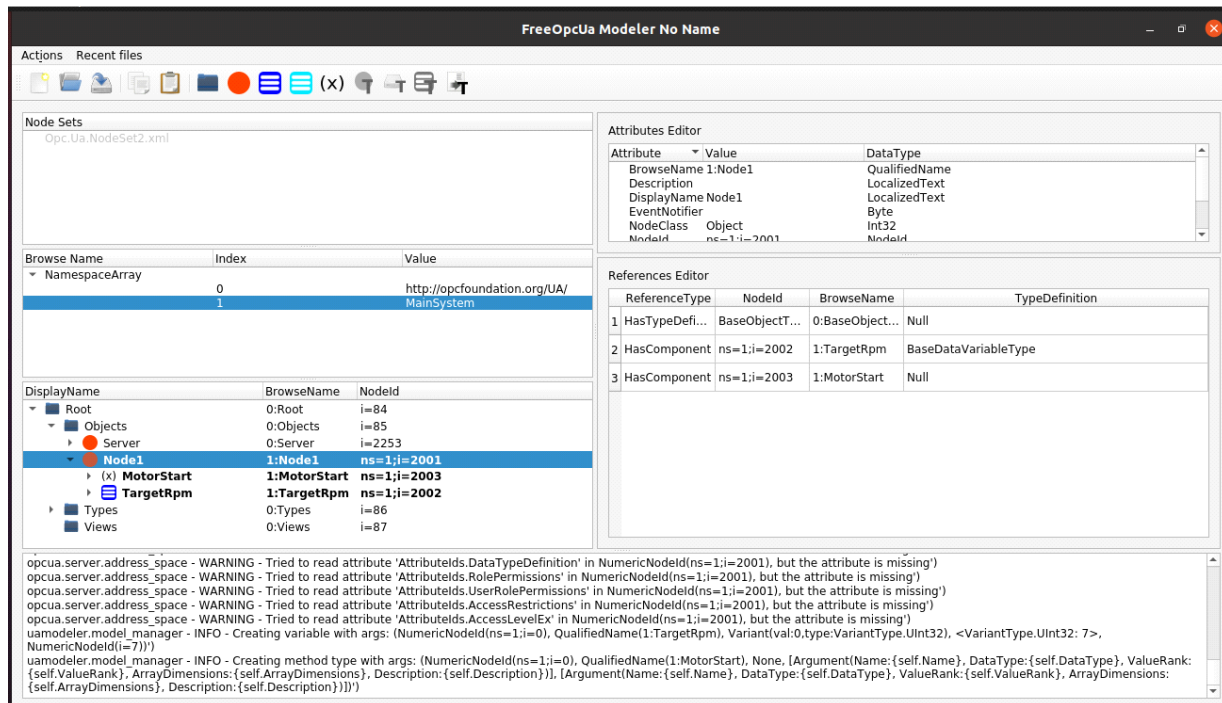


Figure 8.110: opcua modeler

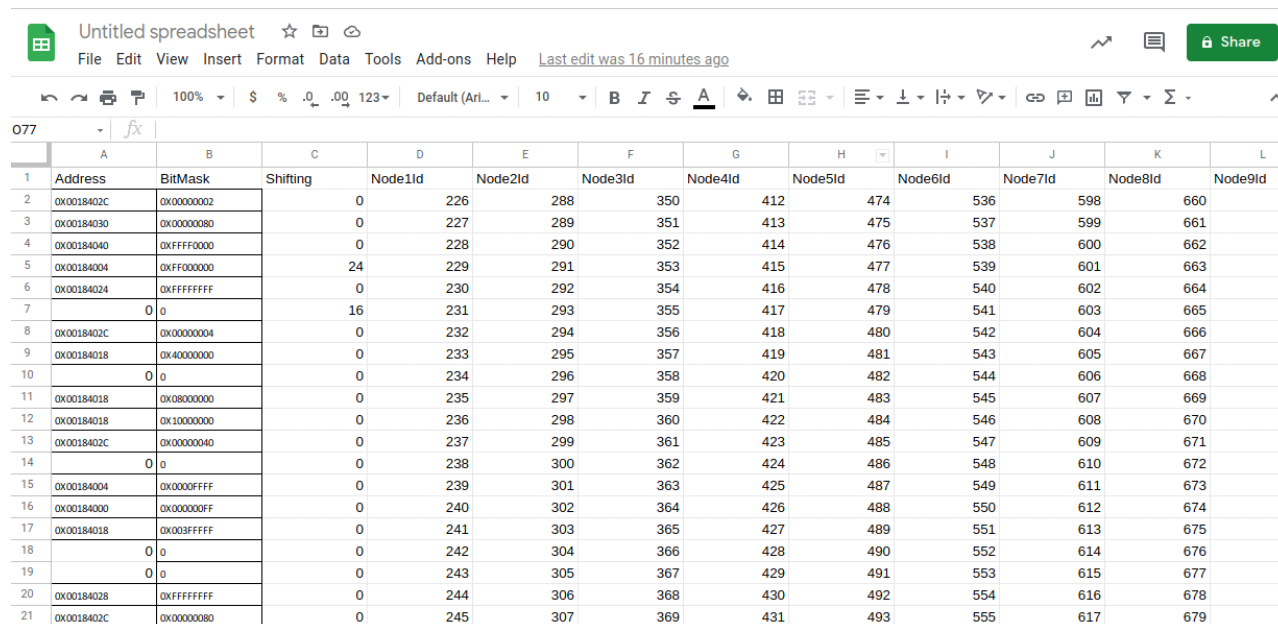
19. If any variables or method in xml information model using opcua-modeler then run ./xmlCmd_script.sh for register all variables and method callbacks in xml .c and .h file.
20. Xml .c and .h file generate Node set compiler.
21. This script is located in Scripts directory.

```
pi@raspberrypi:~/Lattice_Automate_Stack_1_1/OPCUA_Server/OPCUA_ServerApp/Scripts $ ./xmlCmd_script.sh
INFO: _main_:Preprocessing (existing) ../lib/open62541/deps/ua-nodeset/Schema/Opc.Ua.NodeSet2.xml
INFO: _main_:Generating Code for Backend: open62541
INFO: _main_:NodeSet generation code successfully printed
pi@raspberrypi:~/Lattice_Automate_Stack_1_1/OPCUA_Server/OPCUA_ServerApp/Scripts $
```

Figure 8.111: xml script

Appendix I. CSV (Comma separated value) File

1. CSV file is a text file, it contains a list of data separated by commas.



	A	B	C	D	E	F	G	H	I	J	K	L
1	Address	BitMask	Shifting	Node1Id	Node2Id	Node3Id	Node4Id	Node5Id	Node6Id	Node7Id	Node8Id	Node9Id
2	0X0018402C	0X00000002	0	226	288	350	412	474	536	598	660	
3	0X00184030	0X00000080	0	227	289	351	413	475	537	599	661	
4	0X00184040	0XFFFF0000	0	228	290	352	414	476	538	600	662	
5	0X00184004	0XFF000000	24	229	291	353	415	477	539	601	663	
6	0X00184024	0XFFFFFFFF	0	230	292	354	416	478	540	602	664	
7	0	0	16	231	293	355	417	479	541	603	665	
8	0X0018402C	0X00000004	0	232	294	356	418	480	542	604	666	
9	0X00184018	0X40000000	0	233	295	357	419	481	543	605	667	
10	0	0	0	234	296	358	420	482	544	606	668	
11	0X00184018	0X08000000	0	235	297	359	421	483	545	607	669	
12	0X00184018	0X10000000	0	236	298	360	422	484	546	608	670	
13	0X0018402C	0X00000040	0	237	299	361	423	485	547	609	671	
14	0	0	0	238	300	362	424	486	548	610	672	
15	0X00184004	0X0000FFFF	0	239	301	363	425	487	549	611	673	
16	0X00184000	0X000000FF	0	240	302	364	426	488	550	612	674	
17	0X00184018	0X003FFFFFFF	0	241	303	365	427	489	551	613	675	
18	0	0	0	242	304	366	428	490	552	614	676	
19	0	0	0	243	305	367	429	491	553	615	677	
20	0X00184028	0XFFFFFFFF	0	244	306	368	430	492	554	616	678	
21	0X0018402C	0X00000080	0	245	307	369	431	493	555	617	679	

Figure 8.112: CSV file

2. 1st column represents an address, this column contains an Address of (Main/Node) variable.
3. 2nd column represents a Bitmask, this column contains a Bitmask of (Main/Node) variable.
4. 3rd column represents a Shifting, this column contains a Shifting of (Main/Node) variable.
5. Nodeid column represents a Nodeid, this column contains all variable Nodeid of (Main/Node) variables.

Appendix J. Setting up the Auto-Bootable MQTT-Based Client

To set up the auto-bootable MQTT-based client, perform the steps in the sections mentioned below.

Unzip the folder:

1. Open the terminals
2. Unzip the bundle.
3. Run the command: **unzip Mqtt_Lattice_AutomateStack_2_0.zip**.

Perform the steps below after unzip:

1. Run the command: **cd Mqtt_Lattice_AutomateStack_2_0/ lib/paho.mqtt.c/**.
2. Run the command to clean the old package: **sudo make clean**.
3. Run the command for compilation: **sudo make && sudo make install**.

Run the commands below if you get the Open SSL error:

1. **sudo apt-get install libssl-dev**
2. **sudo make && sudo make install**

To make the new server executable, run the following commands:

1. **cd Mqtt_Lattice_AutomateStack_2_0/Scripts/**
2. **sudo make clean**
3. **sudo make**

Run the below command for installing the mosquitto broker:

1. **sudo apt update && sudo apt upgrade**
2. **sudo apt install -y mosquitto mosquitto-clients sudo systemctl enable mosquitto.service**
3. **sudo nano /etc/mosquitto/mosquitto.conf**
4. Mosquitto.conf file will open after this copy the below two command in that file.
5. **listener 1883**
6. **allow_anonymous true**
7. For save and close that mosquitto.cnfg file.
8. **press ctrl+x**
9. **type y and press Enter**

Automate the application:

10. Run the command: **sudo crontab -e**
11. Type 1 and press **Enter** to select the nano editor. The crontab opens.
12. Copy the line below and paste it in the crontab at the end of the file. **@reboot /home/pi/Mqtt_Lattice_AutomateStack_2_0/Scripts/autoapp.sh**
13. Press **ctrl+x**, type **y** and press **Enter**.

Setting Up the IPV4 Address and Router on Raspberry Pi:

1. Right-click on  at the right side of the window, and then click on **Wireless and Wired Network Settings**.

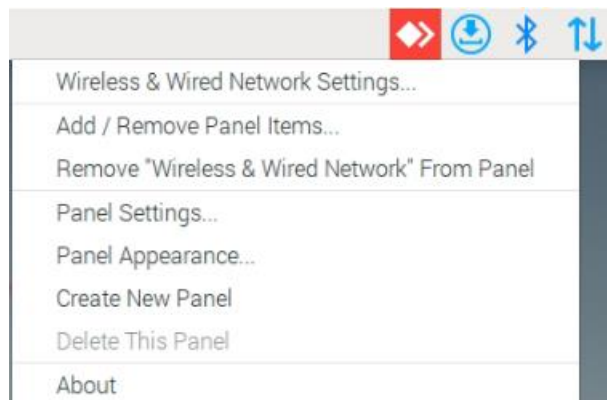


Figure 8.113. IPV4 Address Setting

- The Network Preferences screen is displayed. Select the **eth0** from the drop-down menu.

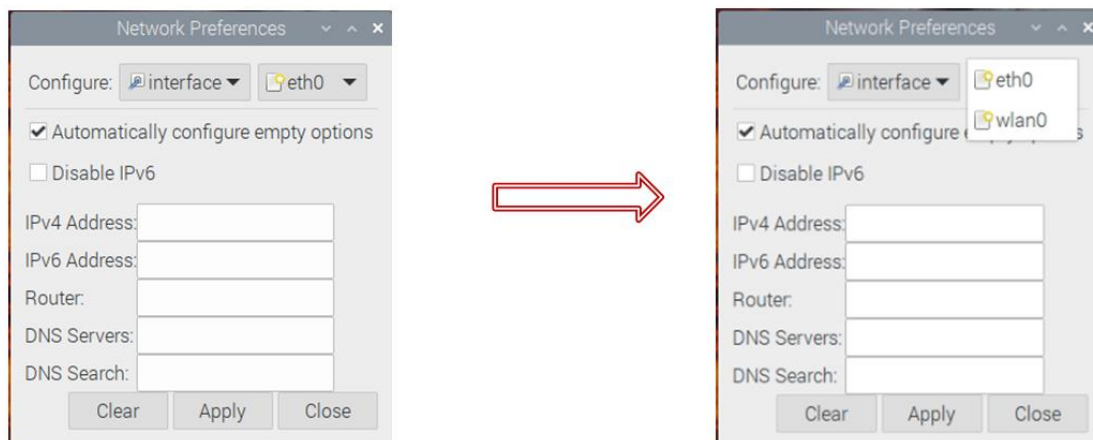


Figure 8.114. Network Preferences Settings

- Untick the Automatically configure empty box.
- Enter the IP address and router as mentioned in below figure and click **Apply**.
 - IPV4 Address – 10.0.1.112**
 - Router – 192.168.1.1**

Note: Router value same as the value of Default gateway in your laptop.

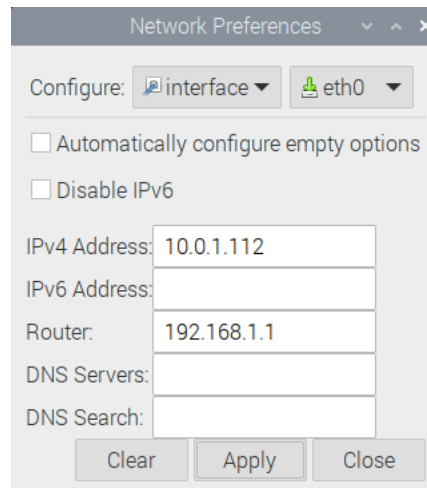


Figure 8.115. IP Configuration

5. Run this command to reboot the Raspberry Pi: **sudo reboot**.

Technical Support Assistance

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

Revision History

Revision 1.0, June 2022

Section	Change Summary
	Initial release.



www.latticesemi.com