MachXO5-NX Device Provisioning User Guide

Preliminary Technical Note

FPGA-TN-02333-0.80

December 2023

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults and associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. 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. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. 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

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
ECC384	Elliptic Curve Cryptography with 384-bit key
ESFB	Embedded Security and Function Block
FPGA	Field Programmable Gate Array
FTDI	Future Technology Devices International
IP	Intellectual Property
ISK	Image Signing Key
JTAG	Joint Test Action Group
KAK	Key Authentication Key
UFM	User Flash Memory

1. Introduction

The LFMXO5-15D device in the MachXO5™-NX family is a continuation of the MachXO3D™ product line with additional FPGA fabric resources and a larger flash size. It features an enhanced security capability of 384-bit key strength compared to 256-bit in MachXO3D. The MachXO5-NX also maintains enhanced bitstream security and user mode security functions, similar to MachXO3D.

Unlike the conventional FPGA, the MachXO5-NX devices require you to provision the booting image into the internal flash as part of the provisioning process to run the user design. This document provides a step-by-step guide to provision the security policy data, security key data, central locks data, booting image and optionally the UFM data into the internal flash of the MachXO5-NX device. For the complete information on the definition of the security policy data, security key data and central locks data, refer to Embedded Security and Function Block User Guide for MachXO5-NX Devices (FPGA-TN-02320) and Advanced Key Management User Guide for MachXO5-NX (FPGA-TN-02321).

This document also walks you through the process of generating the security policy data, security key data and the central locks data using Lattice Customer Key Lock Editor in the Lattice Propel™ Software for provisioning purposes.

2. Generating Programming Files for Provisioning

The user needs to generate several programming files to provision the MachXO5-NX device. This is shown in Table 2.1.

Table 2.1. Programming files for MachXO5-NX device provisioning

Programming Files	File extension	Generated using
Security Policy	.bin	Lattice Customer Policy Key Lock Editor in Lattice Propel™ Software.
Security Keys	.bin	
Central Locks	.bin	
Booting Image (at least 1, optional 2)	.bit	Lattice Radiant™ software.
Optional UFM sector data, up to 8	.bin	Any binary editor tool by users.

Lattice Semiconductor provides the Lattice Customer Policy Key Lock Editor in the Lattice Propel Software to generate or modify the Security Policy, Security Keys, and Central Locks binary files. The editor is available in the Lattice Propel version 2023.2 or later versions.

2.1. Using the Policy Key Lock Editor

The Policy Key Lock Editor is part of the Lattice MachXO5-NX RoT tools in the Lattice Propel software.

To open the Policy Key Lock Editor from the Lattice Propel main interface, select **Lattice Tools-> Lattice MachXO5-NX RoT Tools-> Policy Key Lock Editor**, as shown in Figure 2.1.

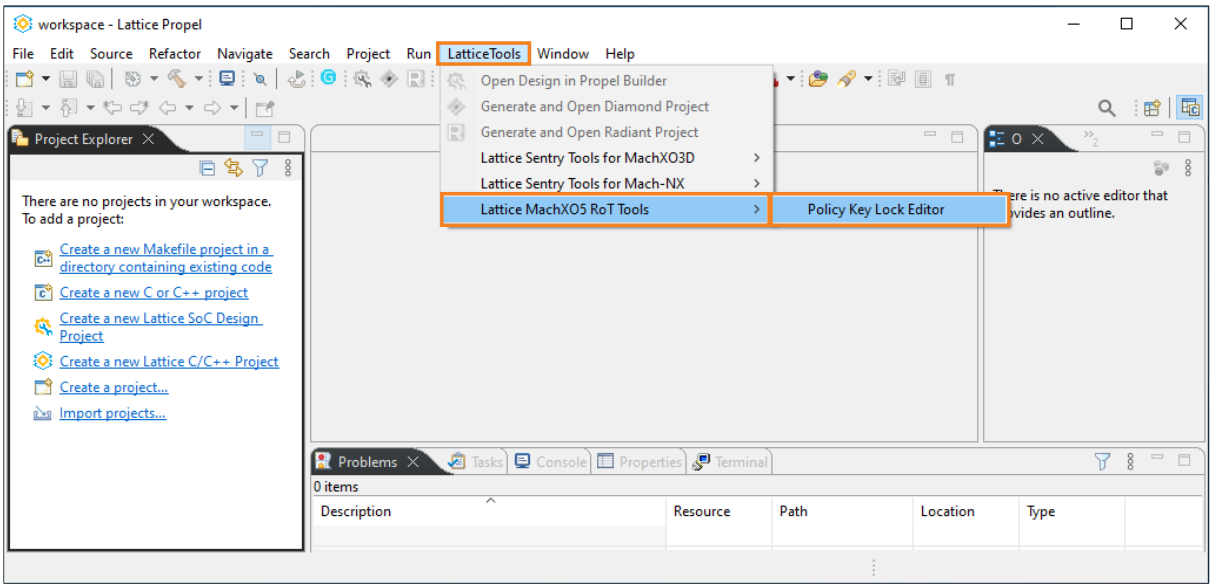

Figure 2.1. Launch Policy Key Lock Editor from Lattice Propel

The **Lattice Customer Policy Key Lock Editor** interface opens. There are three tabs, the **Policy Data**, **Key Data**, and **Lock Data**, as shown in Figure 2.2.

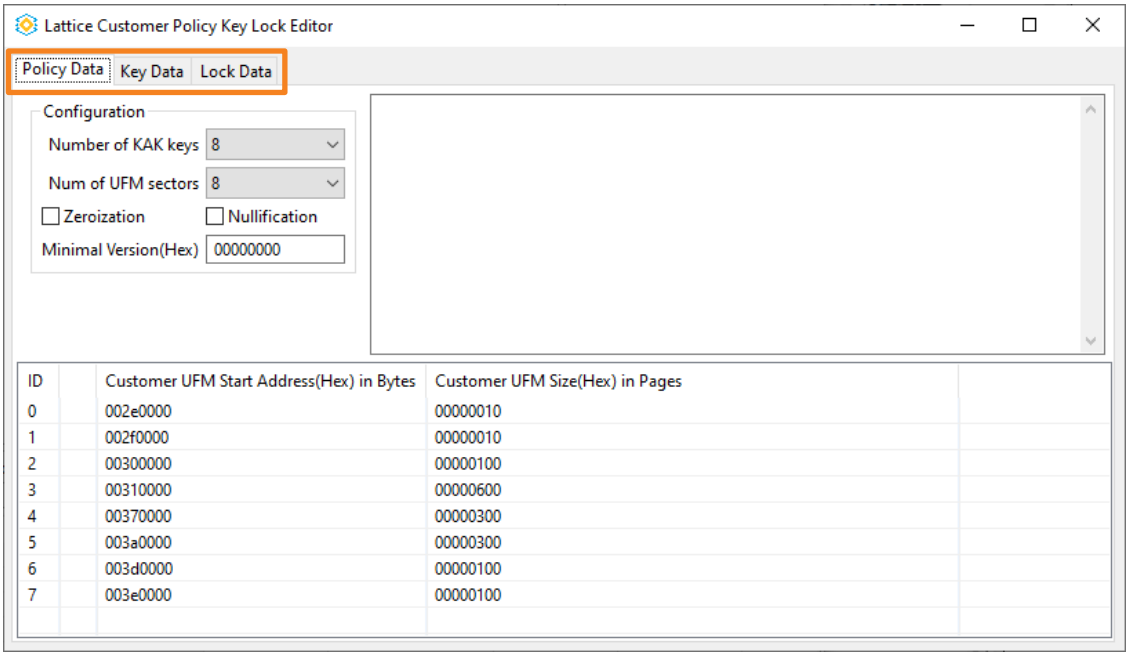

Figure 2.2. Policy Key Lock Editor

2.1.1. Generating the Policy Data

The user can generate a new policy data or modify the existing policy data.

1. Select the **Policy Data** tab.
2. If the user needs to modify the existing policy data, click the **Open** button to open the policy data file to get the current setting. Figure 2.3 shows the example settings in the Policy Data tab.

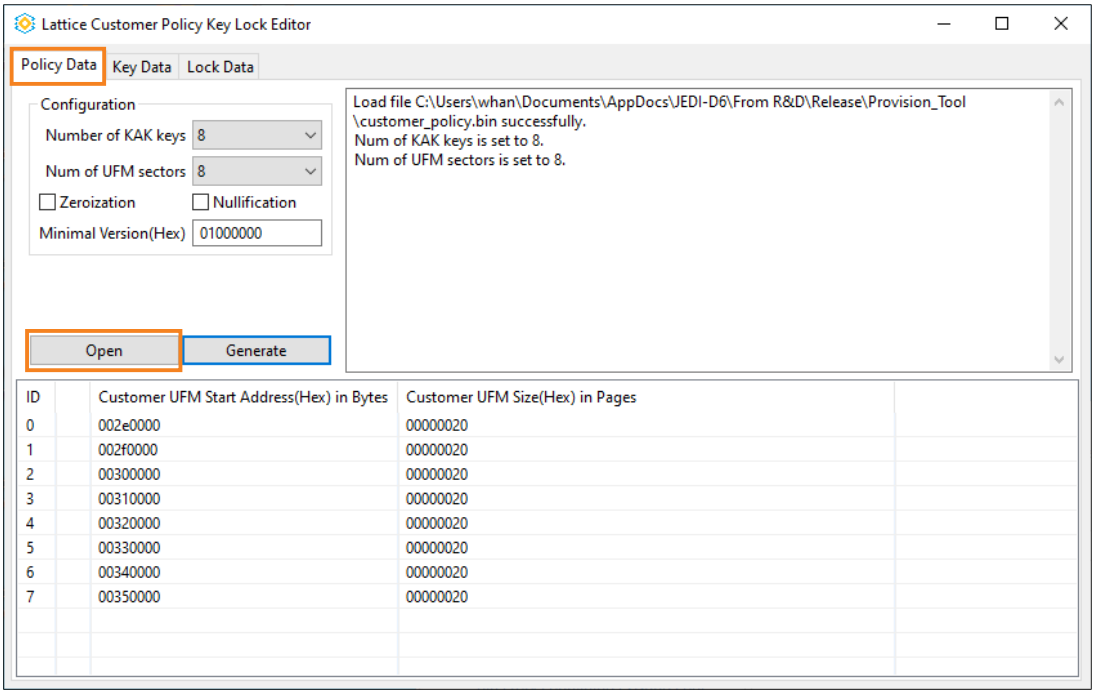

Figure 2.3. Generating the Policy Data

3. Set the values in the **Number of KAK keys** and the **Number of UFM sectors**.
4. Check or leave unchecked the **Zeroization** and **Nullification** options.

Table 2.2 provides the definitions for Configuration Items in the **Policy Data** tab.

Table 2.2. Policy Configuration Definition

Configuration Item	Default Value	Selectable Range	Definition
Number of KAK keys	8	1 ~ 8	Number of KAK keys for MachXO5-NX provisioning
Number of UFM sectors	8	1 ~ 8	Number of UFM sectors for MachXO5-NX provisioning
Zeroization	Uncheck	Check Uncheck	If checked, device zeroization is allowed through the *authorize_and_perform_zeroization* API, which clears all provisioning setup data and keeps only the immutable image.
Nullification	Uncheck	Check Uncheck	If checked, allow the device to be nullified through *authorize_and_perform_nullification* API, which totally disables the device from functioning.
Minimal Version (Hex)	00000000	00000000 FFFFFFFE	Minimal Bitstream Version Value (32 bits) for VRBP (Version Roll Back Protection).

5. Adjust the UMF arrangement by specifying the start address and size for each UFM (maximum 8 UFMs) under **Customer UFM Start Address (Hex) in Bytes** and **Customer UFM Size (Hex) in Pages**.

   Note: Only the UFM setting with the ID less than the **Number of UFM sectors** specified in the policy configuration is processed by the provisioning flow.

   After finalizing the settings, click the **Generate** button.
6. Choose the file name for the *.bin* file.

   The generated binary file is the input for the *program_customer_policy* API inside the Security Policy Programming flow during provisioning.

2.1.2. Generating the Key Data

To add or modify the Key Data:

1. Select the **Key Data** tab.
2. If the user needs to modify the existing key data, click the **Open** button to open the Key data file to get the current Key data settings. Figure 2.4 shows the example settings in the Key Data tab.

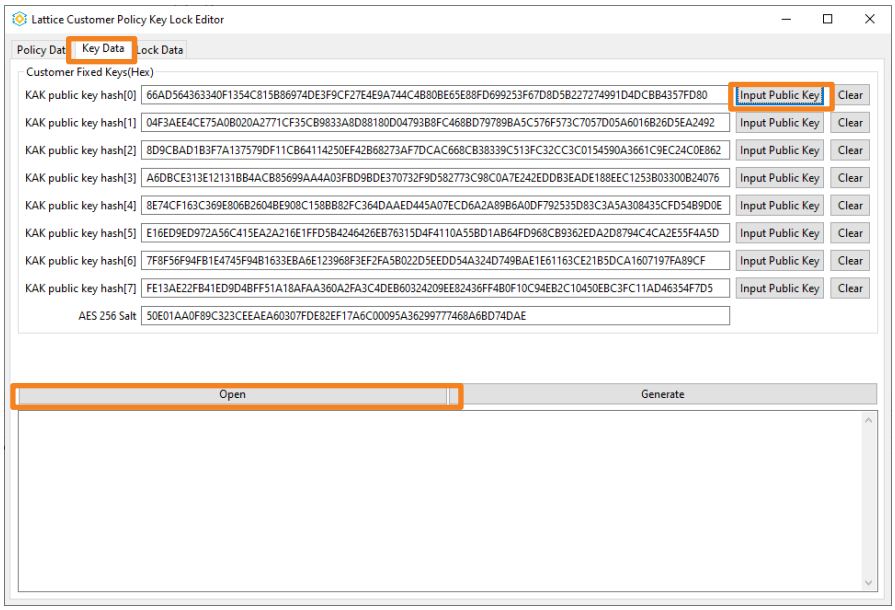

Figure 2.4. Generating the Key Data

3. Add/update the **KAK Public Key Hash** (384 bits each) for the maximum 8 keys.
    - To add the KAK Public Key Hash, you can key in the 384-bit hexadecimal value into the text field directly, or click the **Input Public Key** button to open up the KAK Public Key input dialog box as shown in Figure 2.5.
    - Fill in the 768-bit hexadecimal KAK Public Key.
    - Click the **OK** button to close the dialog box and to populate the KAK Public Key hash field with the calculated SHA-384 hash value.

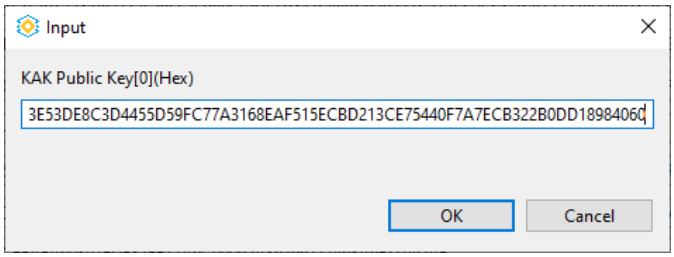

Figure 2.5. Input KAK Public Key to generate KAK public key hash

Note: A KAK Public Key Hash index less than the **Number of KAK keys** specified in the policy configuration is processed only by the provisioning flow.

4. Add or update the **AES 256 Salt** for the device's secret AES key generation.
5. After finalizing the settings, click the **Generate** button.
6. Choose the file name for the *.bin* file.

   The generated binary file is the input for the *program_customer_keys* API inside the Security Key Programming flow during provisioning.

2.1.3. Generating the Central Lock Data

To add or modify the Lock Data:

1. Select the **Lock Data** tab.
2. If the user needs to modify the existing Central Lock Data, click the **Open** button to open the Central Lock Data file to get the current Central Lock Data setting. Figure 2.6 shows the example settings in Lock Data tab.

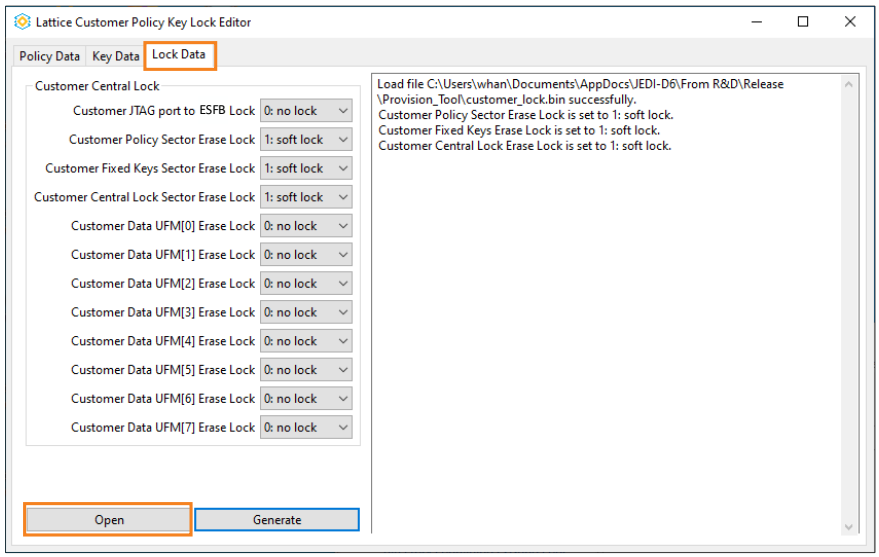

Figure 2.6. Modify the Central Lock Data

3. Update the lock settings for the MachXO5 provisioning.

   This internal lock state defines whether the corresponding sector can be erased/written or not.

   Note: Regardless of the policy setting, *Read* API is always available.

   Table 2.3 shows the Central Lock selection and its effects.

Table 2.3. Central Lock Definition

Central Lock Setting	Default	No Lock	Soft Lock	Hard Lock
Customer JTAG Port to ESFB Lock	Unlocked	*jtag_interface_lock_unlock* API can change lock status without a KeyBlob.	*jtag_interface_lock_unlock* API can change lock status with a valid KeyBlob.	Cannot change the lock status. Always locked.
Customer Policy Sector Erase Lock	Locked	*authorize_and_erase_ customer_policy* API can change status from locked to unlocked without a KeyBlob. *audit and lock* API can change the state to locked if audit success.	*authorize_and_erase_ customer_policy* API can change state from locked to unlocked with a valid KeyBlob. *audit and lock* API can change the state to locked if audit success.	*authorize_and_erase_ customer_policy* API can never success.
Customer Fixed Keys Sector Erase Lock	Locked	*authorize_and_erase_ customer_keys* API can change state from locked to unlocked without a KeyBlob. *audit and lock* API can change the state to locked if audit success	*authorize_and_erase_ customer_keys* API can change state from locked to unlocked with a valid KeyBlob. *audit and lock* API can change the state to locked if audit success	*authorize_and_erase_ customer_keys* API can never success
Customer Central Lock Sector Erase Lock	Locked	*authorize_and_erase_custo mer_central_locks* API can change state from locked to unlocked without a KeyBlob. *audit and lock* API can change the state to locked if audit success.	*authorize_and_erase_ customer_central_locks* API can change state from locked to unlocked with a valid KeyBlob. *audit and lock* API can change the state to locked if audit success.	*authorize_and_erase_ customer_central_locks* API can never success.
Customer Data UFM [0:7] Erase Lock	Unlocked	*ufm_lock_unlock* API can change the lock status successfully.	*ufm_lock_unlock* API can change the lock status successfully.	Cannot change the lock status. Always locked.

4. After finalizing the setting, click the **Generate** button.
5. Choose the file name for the *.bin* file.

   The generated binary file is the input for the *program_customer_central_locks* API inside the Central Locks Programming flow during provisioning.

3. Device Provisioning

There are several ways to provision the MachXO5-NX device. This section provides guidelines on provisioning using the following:

- Radiant Programmer
- Standard Test and Programming Language (STAPL) file
- Software APIs (in C source code)

Figure 3.1 shows the general provisioning flow during device provisioning. The provisioning flow is the same regardless of how the device is provisioned.

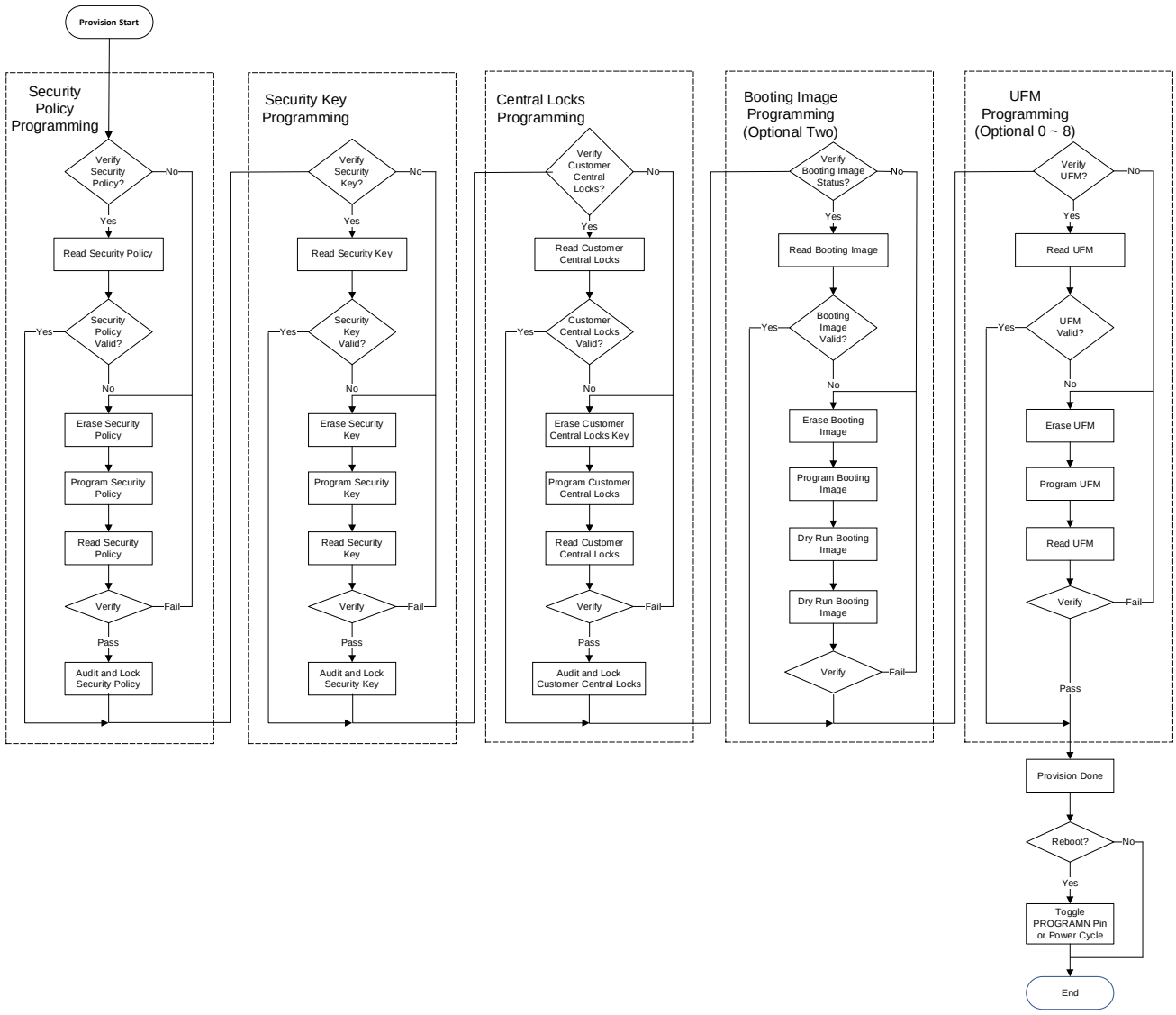

Figure 3.1. MachXO5-NX Provisioning Flow Chart

3.1. Provisioning using Radiant Programmer

The user may provision the MachXO5-NX device using the Radiant Programmer. Follow the instructions below:

1. The user can launch the Radiant Programmer from the Lattice Radiant Software by choosing the **Tool** menu. Click **Programmer**, or click the Programmer icon from the menu bar. Alternatively, the user can launch the Programmer from the standalone Radiant Programmer.

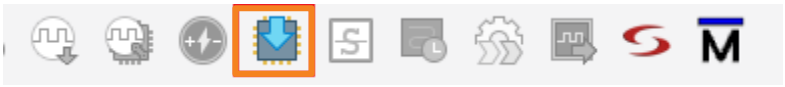

Figure 3.2. Radiant Programmer Menu

2. Fill in the necessary information to start a new project or open an existing project.

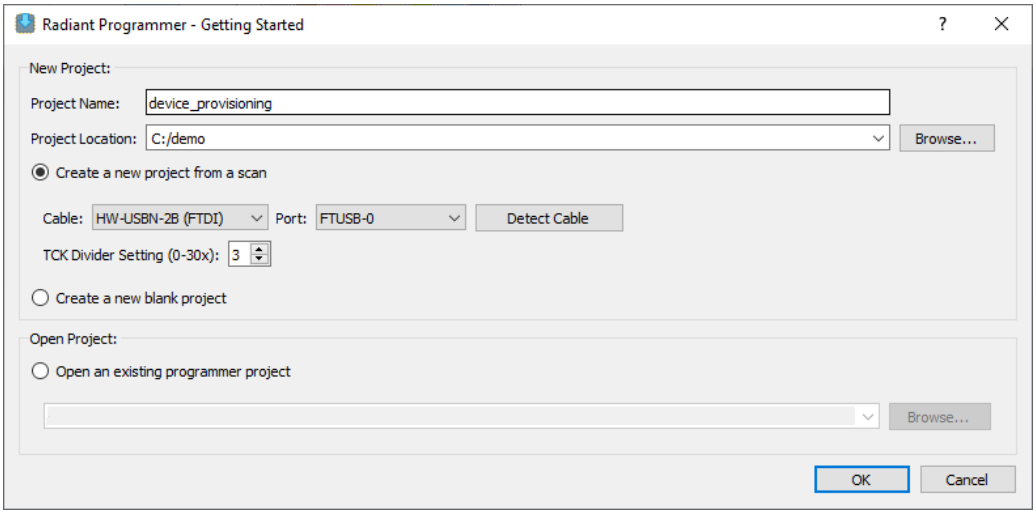

Figure 3.3. Radiant Programmer - Getting Started

3. Click the **Run** menu, and the **Scan Device**.

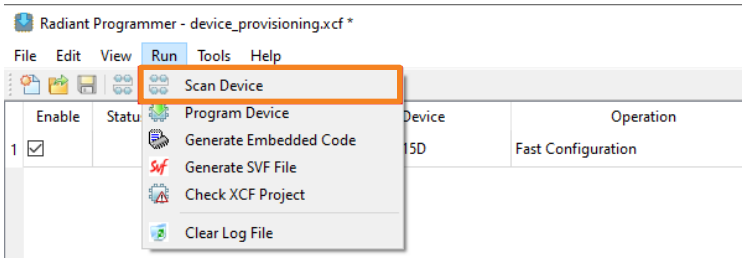

Figure 3.4. Radiant Programmer – Scan Device

4. Make sure the MachXO5-NX secure device is detected in the Programmer as shown in Figure 3.5.

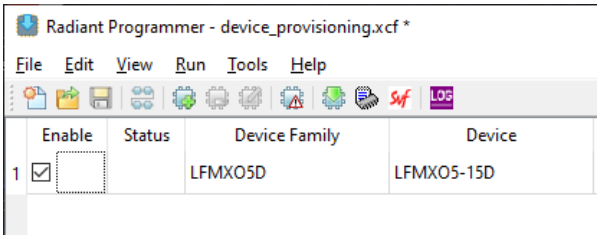

Figure 3.5. MachXO5-NX secure device detected

5. Launch the Device Properties window.

   Click the **Edit** menu, and click **Device Properties.** Alternatively, the user can double click the row 1 at Operation column to launch the Device Properties window.

6. In the General tab of Device Properties window, under Device Operation, select the following options:
   a. **Target Memory: Provision FLASH Configuration Memory**
   b. **Port Interface: JTAG**
   c. **Access Mode: Direct FLASH Programming**
   d. **Operation: Erase, Program, Verify, Refresh**
7. Under the **Provisioning Flash Programming Options**, choose the **Policy, Keys** and **Locks Programming file**.

   Note that all these 3 programming files are mandatory for provisioning.

8. Choose the user image (.bit file) in **CFG0** and **CFG1 Programming Options**.

   Note that CFG0 is mandatory, while CFG1 is optional.

9. For the **User Flash Memory Programming Options**, it has dependency on the **Policy Programming file** selected by the user. The number of available **User Data Programming option** is following the number of UFM sector defined in the **Policy Programming file**. Same goes to the Start and End address of the UFM sector. It follows the start address and UFM size defined in **Policy Programming file**.
10. After the user configures all the options shown in Figure 3.6, click the **OK** button to close the Device Properties window.

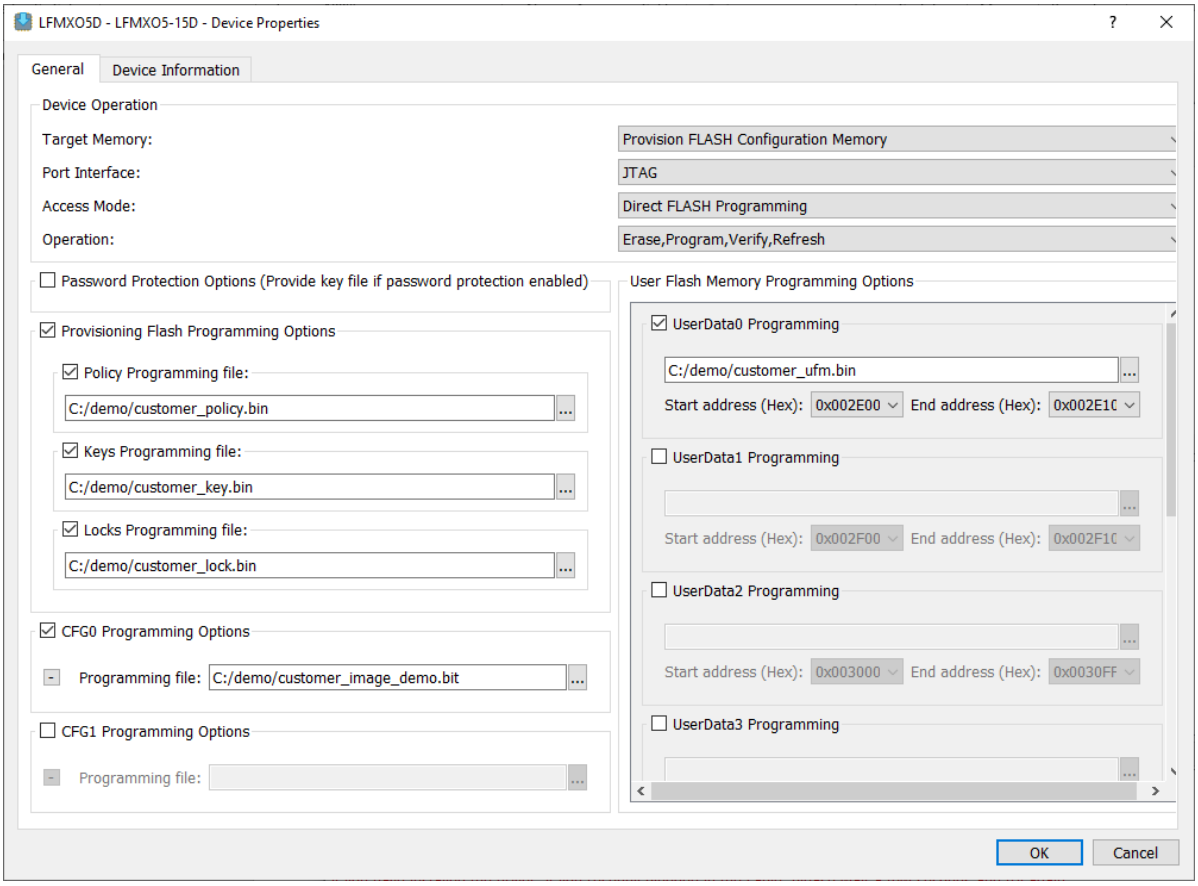

Figure 3.6. Configure Programmer options for device provisioning

11. Click the **Run** menu and click **Program Device** to start the provisioning process. Alternatively, the user can click the **Program Device** button from the menu bar shown in Figure 3.7. The provisioning will proceed. Check the output window of the Programmer to see the provisioning status.

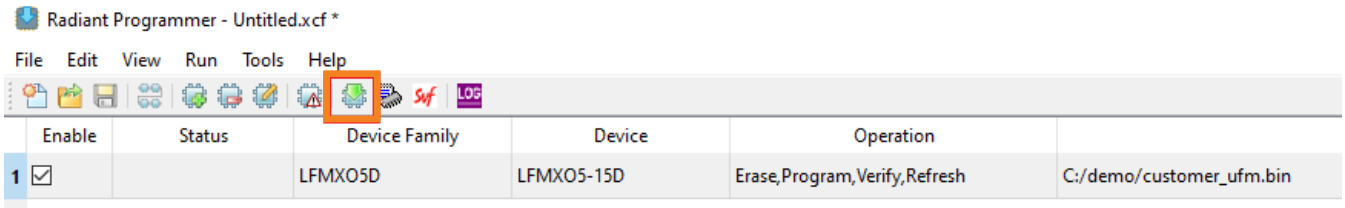

Figure 3.7. Program Device Button

3.2. Provisioning using STAPL File

STAPL is the industry standard for executing the JTAG command. Many vendors who create programming tools and solutions use it extensively. The Radiant Deployment Tool is provided by Lattice, allowing the users to generate a STAPL file for device provisioning. This section walks you through the process of generating the STAPL file and using the Download Debugger to execute the generated STAPL file to do provisioning for MachXO5-NX devices.

3.2.1. Generating STAPL File

To generate the STAPL file, the user must save the Radiant Programmer Project to an XCF file with the.xcf extension. To generate the XCF file, follow the steps below, and then use the Radiant Deployment tool to generate the STAPL file.

1. For more information, refer to Section 3.1 Provisioning using Radiant Programmer. Once all the parameters are configured correctly, click the Radiant Programmer **File,** then click the **Save Project As** option. Save the file in *.xcf* format at the location of your choice.

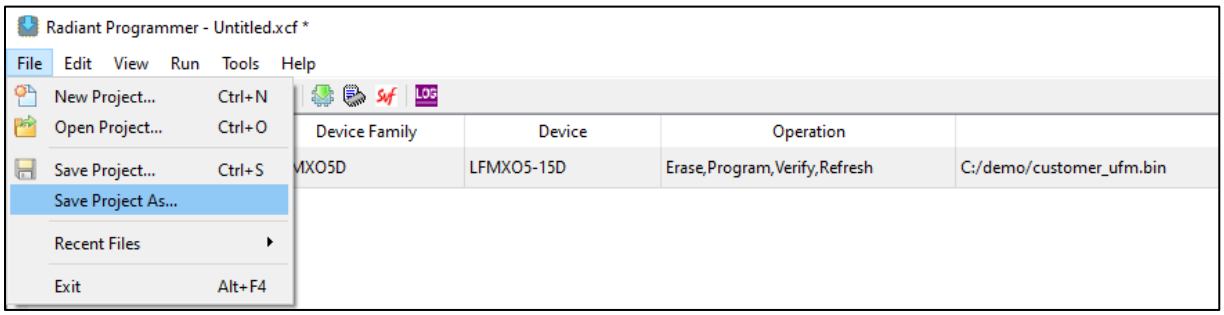

Figure 3.8. Save Project As option

2. Launch the Radiant Deployment Tool.

   Click the **Tool** menu and click **Deployment Tool**.

3. Select **Create New Deployment**, set the **Function Type** to **Tester** and the **Output File Type** to **STAPL – JTAG Chain**. Click the **OK** button to proceed.

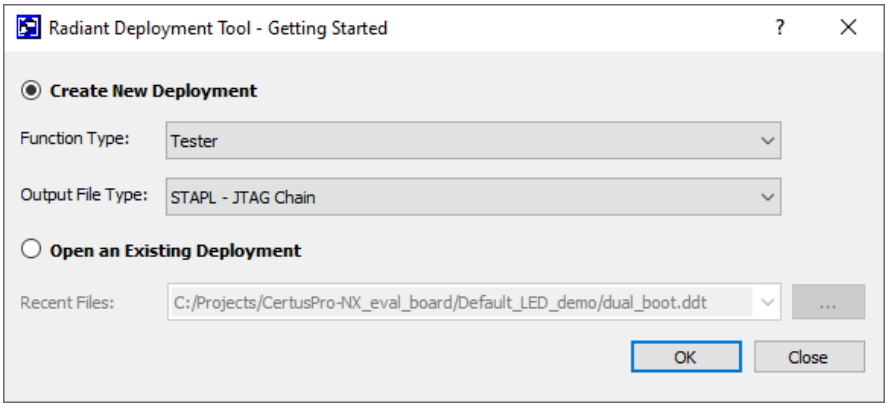

Figure 3.9. Radiant Development Tool

4. Select the **.xcf** file that was previously generated by the Radiant Programmer in the **Input XCF file** field as shown in Figure 3.10. Click the **Next** button to proceed.

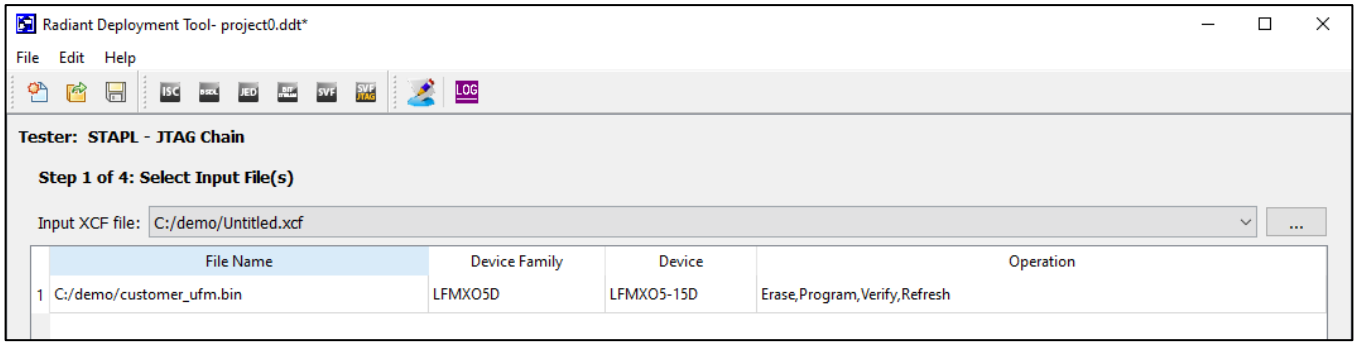

Figure 3.10. Input XCF File

5. Leave the default STAPL parameters and click the **Next** button to proceed.
6. Give the STAPL file name for the .stp output file name. Click the **Next** button to proceed.
7. Ensure all the parameters are correctly configured as shown in Figure 3.11. Click the **Generate** button to generate the .stp file.

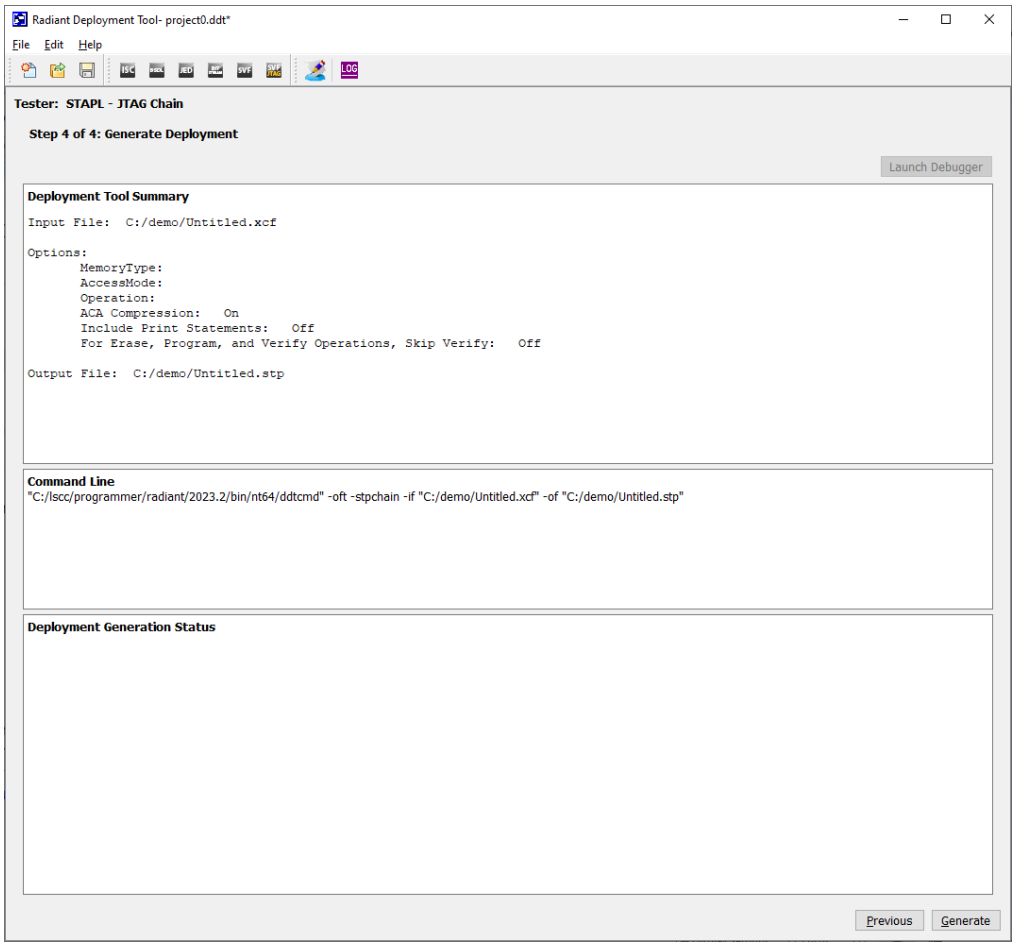

Figure 3.11. Deployment Tool setting to generate the STAPL file.

3.2.2. Executing STAPL File Using the Download Debugger

To execute the STAPL file using the Download Debugger, follow the steps below:

1. Launch the Radiant Programmer. Click the **Tools** menu, and select the **Download Debugger**.
2. Ensure the device is detected by the Download Debugger. Click the **Configuration** menu and select **Cable and I/O Port Setup.** The dialog box shall be configured as shown in Figure 3.12. Click the **OK** button to continue.

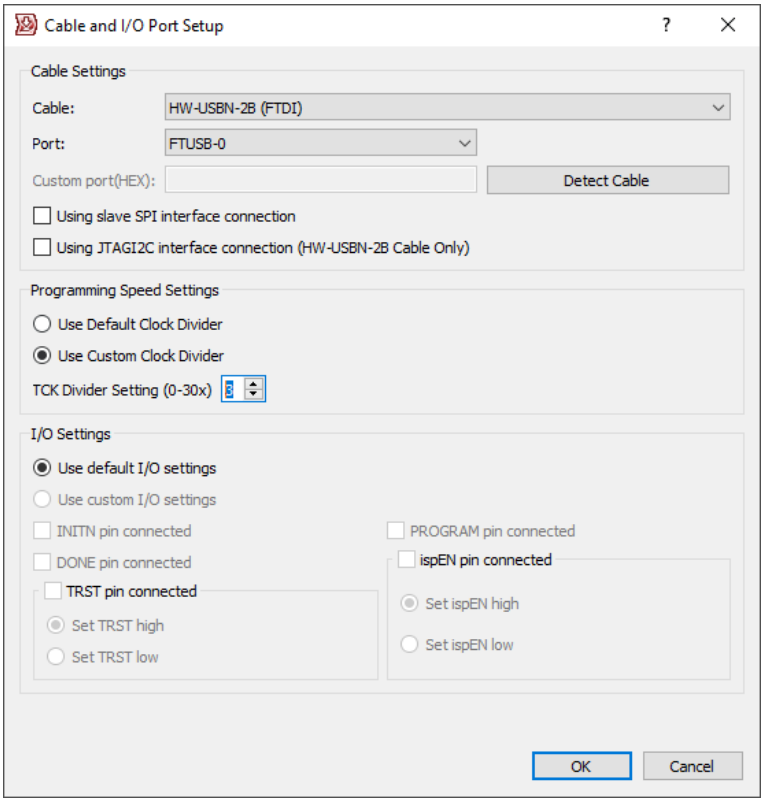

Figure 3.12. Configuration Setup – Cable and I/O Port.

3. Click **File** menu and select **Open.**
4. Select the generated .stp file.

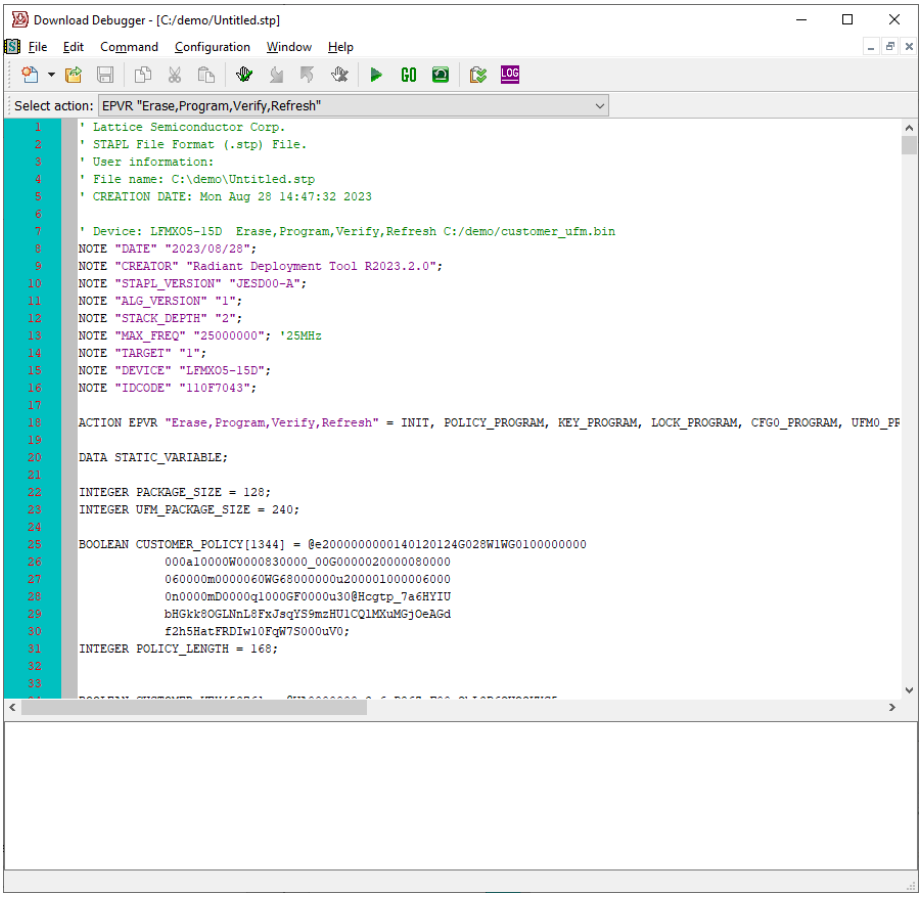

Figure 3.13. Generated STP file

5. Click the **Command** menu and select **Go** to execute the STP file.

3.3. Provisioning using Software APIs

Lattice provides the device provisioning reference code in C source code. The source code provides the user the flexibility to integrate provisioning into the host or embedded system. If the user needs the source code and the documentation to use the Software APIs for device provisioning, please submit a technical support case at www.latticesemi.com/techsupport and request for FPGA-RD-02259 and the related reference code.

References

For more information refer to:

- Embedded Security and Function Block User Guide for MachXO5-NX Devices (FPGA-TN-02320)
- Advanced Key Management User Guide for MachXO5-NX (FPGA-TN-02321)
- MachXO5-NX web page
- Development Kits and Boards for MachXO5-NX
- IP and Reference Designs for MachXO5-NX
- Lattice Radiant FPGA design software
- Lattice Propel FPGA design software
- Lattice Insights for Lattice Semiconductor training courses and learning plans
- Lattice Sales Office

Technical Support Assistance

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

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

Revision History

Revision 0.80, December 2023

Section	Change Summary
All	Preliminary initial release.

www.latticesemi.com