



Lattice Nexus 2 Multi-Boot User Guide

Preliminary Technical Note

FPGA-TN-02385-0.81

September 2026

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 all 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 REDUNDANCIES, 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.

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language [FAQ 6878](#) for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents	3
Abbreviations in This Document.....	5
1. Introduction	6
2. Definition of Terms	7
3. Resources.....	8
4. Dual-Boot Mode	9
4.1. Dual-Boot Flow	10
5. Ping-Pong Boot Mode	11
5.1. Ping-Pong Boot Flow	12
6. Multi-Boot Mode	13
6.1. Multi-Boot Flow	14
6.2. Implementation of Static or Dynamic Multi-Boot Mode	15
6.2.1. Static Multi-Boot Mode Host Controller Sequence	16
6.2.2. Dynamic Multi-Boot Mode Host Controller Sequence	16
7. Creating an SPI Storage Device Hex File – For Certus-N2 Only	18
7.1. Using Radiant Deployment Tool to Create a Dual-Boot Hex File	18
7.2. Using Radiant Deployment Tool to Create Ping-Pong Boot Hex File	24
7.3. Using Radiant Deployment Tool to Create Multi-Boot Hex File	30
8. Programming Dual-Boot, Multi-Boot, or Ping-Pong Boot Pattern into an External SPI Storage Device – For Certus-N2 Only	38
9. Programming Dual-Boot or Ping-Pong Boot Pattern into an Internal Memory Storage Device – For Mach-N2 Only	41
10. Pattern Update Guideline	44
References	45
Technical Support Assistance	46
Revision History	47

Figures

Figure 4.1. Dual-Boot Flow Diagram	9
Figure 5.1. Ping-Pong Boot Flow Diagram	11
Figure 6.1. Multi-Boot Flow Diagram.....	13
Figure 6.2. Static Multi-Boot Mode Flow	14
Figure 6.3. Dynamic Multi-Boot Mode Flow	14
Figure 6.4. CONFIG_LMMIC Primitive for Multi-Boot Implementation.....	15
Figure 6.5: Host Controller Sequence for Static Multi-Boot Mode.....	16
Figure 6.6. Host Controller Sequence for Dynamic Multi-Boot Mode.....	17
Figure 7.1. Creating New Deployment for Dual Boot File.....	18
Figure 7.2. Select Input File(s) Window	19
Figure 7.3. Dual Boot Options Window	21
Figure 7.4. Dual Boot Options Window – User Data Files Tab.....	21
Figure 7.5. Select Output File(s) Window	22
Figure 7.6. Generate Deployment Window.....	23
Figure 7.7. Creating New Deployment for Ping-Pong Boot File.....	24
Figure 7.8. Select Input File(s) Window	25
Figure 7.9. Ping-Pong Boot Options Window	27
Figure 7.10. Ping-Pong Boot Options Window – User Data Files Tab.....	27
Figure 7.11. Select Output File(s) Window	28
Figure 7.12. Generate Deployment Window.....	29
Figure 7.13. Configuring MCCLK_FREQ, MULTI_BOOT_MODE, and MULTI_BOOT_SEL in Device Constraint Editor.....	30
Figure 7.14. Creating New Deployment for Multi-Boot File	31
Figure 7.15. Select Input File(s) Window	32
Figure 7.16. Advanced SPI Flash Options Window – Options Tab	33
Figure 7.17. Advanced SPI Flash Options Window – Multiple Boot Tab	34
Figure 7.18. Advanced SPI Flash Options Window – User Data Files Tab.....	35
Figure 7.19. Select Output File(s) Window	36
Figure 7.20. Generate Deployment Window.....	37
Figure 8.1. Radiant Programmer – Getting Started Window.....	38
Figure 8.2. Radiant Programmer – Main Window (with Certus-N2 Device).....	39
Figure 8.3. Radiant Programmer – Device Properties	40
Figure 9.1. Radiant Programmer – Main Window (with Mach-N2 Device).....	42
Figure 9.2. Radiant Programmer – Device Properties for FLASH Configuration Memory	43

Tables

Table 2.1. Definition of Terms	7
Table 3.1. Bitstream Size versus Recommended SPI Storage Size (Single Bitstream Boot Mode)	8
Table 3.2. Bitstream Size versus Recommended SPI Storage Size (Dual-Boot Mode, Ping-Pong Boot Mode).....	8
Table 3.3. Bitstream Size versus Recommended SPI Storage Size (Multi-Boot Mode)	8
Table 4.1. Control Register 1 [2:0] – MSPI Preamble Timer Count.....	10
Table 9.1. Primary Jump Table Structure.....	44

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
BGA	Ball Grid Array
CFG	Configuration
CRC	Cyclic Redundancy Check
EBR	Embedded Block RAM
ECDSA	Elliptic Curve Digital Signature Algorithm
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GUI	Graphic User Interface
ID	Identification
JTAG	Joint Test Action Group
LMMI	Lattice Memory Mapped Interface
MSPI	Controller Serial Peripheral Interface
OSC	Oscillator
PC	Personal Computer
RAM	Random Access Memory
RSA	Rivest-Shamir-Adleman (cryptosystem)
SFDP	Serial Flash Discoverable Parameter
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
UFM	User Flash Memory
xSPI	Expanded Serial Peripheral Interface

1. Introduction

The Lattice Nexus™ 2 platform supports various booting options for loading configuration SRAM from non-volatile memory. These booting options provide configuration flexibility and facilitate fail-safe configuration. Lattice Certus™-N2 devices require an external memory for storage of the configuration bitstreams. Only Lattice Mach™-N2 devices contain internal (on-chip) flash memory.

Multiple configuration boot modes mitigate risk during the field upgrade process and allow flexibility of executing different patterns. Field upgrade disruptions might occur due to power disruption, communication interruption, or bitstream pattern corruption. The Nexus 2 platform supports the following boot modes:

- Single bitstream boot mode
- Dual-boot mode
- Ping-pong boot mode
- Multi-boot mode

The Nexus 2 platform combines multiple bitstream patterns into a single boot image stored in a single external SPI storage device or internal flash memory. This solution decreases cost, reduces board space, and simplifies field upgrades.

2. Definition of Terms

Table 2.1 lists the terms used in this document to describe common functions, features, or concepts.

Table 2.1. Definition of Terms

Term	Definition
Alternative Boot Pattern	In multi-boot mode, after the FPGA device has been configured, this pattern is loaded after PROGRAMN pin pulsing or REFRESH command execution. Up to 16 alternative boot patterns are possible. An alternative boot pattern is also referred to as an alternate pattern in this document.
Binary Hex Data File (.bin File)	The data image of the Hex data file in binary format. All Hex data files are converted into this format prior to consumption.
Bitstream Data File (.bit File)	The configuration data file, for a single FPGA device, in the format that can be loaded directly into the FPGA device to configure the SRAM cells. The file is expressed in binary format.
Configuration	A change in the state of the SRAM memory cells.
Dual-Boot	This feature allows the FPGA device to support two configuration images that reside in a non-volatile memory storage device. Whenever loading failure occurs with the primary image, the FPGA device searches for and loads the secondary image. Both images come from non-volatile memory.
Golden Pattern	The guaranteed good pattern loaded into the FPGA device when booting failure occurs. Only one Golden pattern is allowed.
Hex Data File, Hex File (.exo, .mcs, .xtek Files)	A data record file commonly in a format such as Intel Hex, Motorola Hex, or Extended Tektronix Hex. This file is also known as an addressed record file. It is used for programming configuration memory or flash devices.
Multi-Boot	The FPGA device determines and triggers the loading of the next pattern after a prior successful configuration. Multiple patterns (that is, two patterns or more) are available for the FPGA device to choose to load on demand. All patterns are stored in non-volatile memory.
Ping-Pong Boot	This feature allows the FPGA device to use the jump table to select a pattern for booting without changing the location of the pattern in the non-volatile memory storage device.
Primary Pattern	At power up, the FPGA device loads this pattern first. Only one Primary pattern is allowed.
Programming	The process used to alter the contents of the external SPI storage device or internal flash memory.
Refresh	The process of triggering a configuration data load operation. It is activated by PROGRAMN pin pulsing or REFRESH command execution (which emulates PROGRAMN pin pulsing).

3. Resources

Nexus 2 devices are SRAM-based FPGAs. The SRAM configuration memory must be loaded from an external or internal non-volatile memory that can store all the configuration data. The size of the configuration data varies. It is dependent on the amount of logic available in the FPGA and the number of pre-initialized Embedded Block RAM (EBR) components. A design using the largest Nexus 2 device, with every EBR pre-initialized with unique data values and generated without compression enabled, requires the largest amount of storage.

Table 3.1. Bitstream Size versus Recommended SPI Storage Size (Single Bitstream Boot Mode)

Device	Scenario	Bitstream Size (Mb) ¹	Recommended SPI Storage Device Size (Mb)
LN2-CT-20	No EBR	44	64
	Maximum EBR	64.1	128
LN2-MH-20 ²	No EBR	44	—
	Maximum EBR	64.1	—

Notes:

- Both unencrypted and encrypted bitstreams are the same size. Bitstream compression ratio varies depending on the bitstream so only uncompressed bitstream sizes are shown.
- The LN2-MH-20 device contains 256 Mb of internal flash memory.

Table 3.2. Bitstream Size versus Recommended SPI Storage Size (Dual-Boot Mode, Ping-Pong Boot Mode)

Device	Scenario	Bitstream Size (Mb) ¹		Recommended SPI Storage Device Size (Mb)
		1 Bitstream Pattern	2 Bitstream Patterns	
LN2-CT-20	No EBR	44	88	128
	Maximum EBR	64.1	128.2	256
LN2-MH-20 ²	No EBR	44	88	—
	Maximum EBR	64.1	128.2	—

Notes:

- Both unencrypted and encrypted bitstreams are the same size. Bitstream compression ratio varies depending on the bitstream so only uncompressed bitstream sizes are shown.
- The LN2-MH-20 device contains 256 Mb of internal flash memory.

Table 3.3. Bitstream Size versus Recommended SPI Storage Size (Multi-Boot Mode)

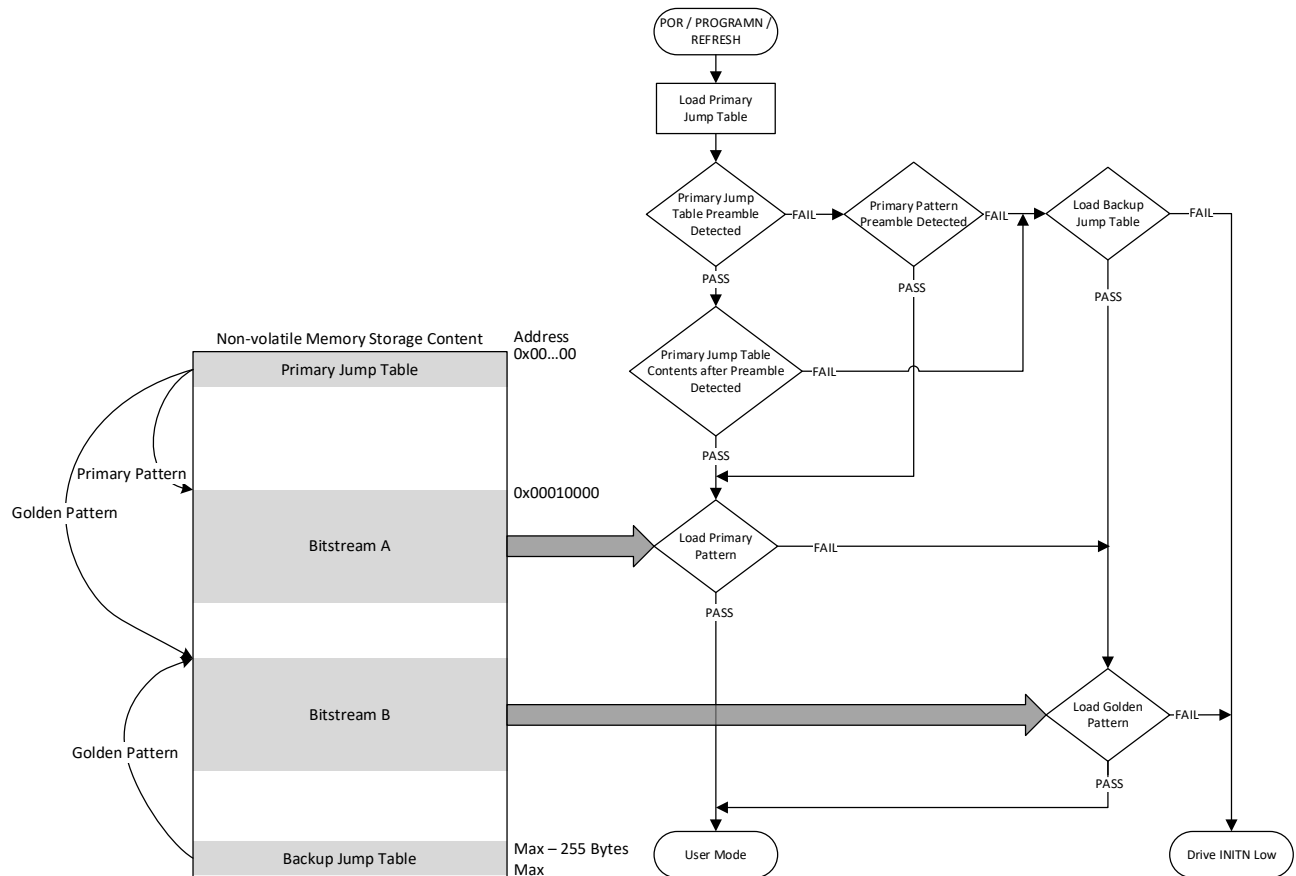
Device	Scenario	Bitstream Size (Mb) ¹					Recommended SPI Storage Device Size (Mb)			
		1 Bitstream Pattern	3 Bitstream Patterns	4 Bitstream Patterns	5 Bitstream Patterns	6 Bitstream Patterns	3 Bitstream Patterns	4 Bitstream Patterns	5 Bitstream Patterns	6 Bitstream Patterns
LN2-CT-20	No EBR	44	132	176	220	264	256	256	256	512
	Maximum EBR	64.1	192.3	256.4	320.5	384.6	256	512	512	512
LN2-MH-20 ²	No EBR	44	132	—	—	—	—	—	—	—
	Maximum EBR	64.1	192.3	—	—	—	—	—	—	—

Notes:

- Both unencrypted and encrypted bitstreams are the same size. Bitstream compression ratio varies depending on the bitstream so only uncompressed bitstream sizes are shown.
- The LN2-MH-20 device contains 256 Mb of internal flash memory.

4. Dual-Boot Mode

The Nexus 2 device dual-boot mode supports booting from two configuration bitstream patterns that reside in an external SPI storage device or internal flash memory. These patterns are designated the primary pattern and golden pattern. The external SPI storage device or internal flash memory contains a primary jump table at address 0x00000000, the primary pattern at address 0x00010000, and the golden pattern at a higher offset address. During initialization, the device executes the commands in the primary jump table to boot from the primary pattern. If the primary pattern fails to load, the device attempts to boot from the golden pattern using the golden pattern offset address specified in the primary jump table. The external SPI storage device or internal flash memory can also store a backup jump table, in case the primary jump table becomes corrupted. Figure 4.1 shows the dual-boot flow diagram.



Notes:

1. The primary jump table points to the addresses of the primary and golden patterns.
2. The backup jump table points to the golden pattern.
3. Primary jump table contents include the *Program Secondary Boot Address* and JUMP command.
4. The primary pattern is loaded either through execution of the JUMP command in the primary jump table or directly when its preamble is detected but the primary jump table preamble is not detected.
5. The golden pattern is loaded either through execution of the JUMP command in the backup jump table or execution of the *Program Secondary Boot Address* command from the primary jump table.
6. The primary pattern (bitstream A) is fixed at address 0x00010000.

Figure 4.1. Dual-Boot Flow Diagram

4.1. Dual-Boot Flow

The dual-boot mode flow is triggered either at power up, PROGRAMN pin pulsing, or REFRESH command execution.

In dual-boot mode, the bitstream pattern might fail to load for one of the following reasons:

- Time-out check — Search for a 32-bit preamble code (0xFFFFBxB3) in the primary pattern, as part of the configuration process, fails.
- Device ID check — ID check at the beginning of the bitstream fails.
- Data integrity check — CRC checks detect data corruption in the bitstream.
- Command validity check — An illegal command is encountered.

Table 4.1 shows the time-out period when loading the primary pattern stored in the external SPI storage device or internal flash memory.

Table 4.1. Control Register 1 [2:0] – MSPI Preamble Timer Count

CR1[2:0]	Timer Value
0	200 ms (default setting)
1	100 ms
2	50 ms
3	40 ms
4	20 ms
5	1 ms
6	500 μ s
7	100 μ s

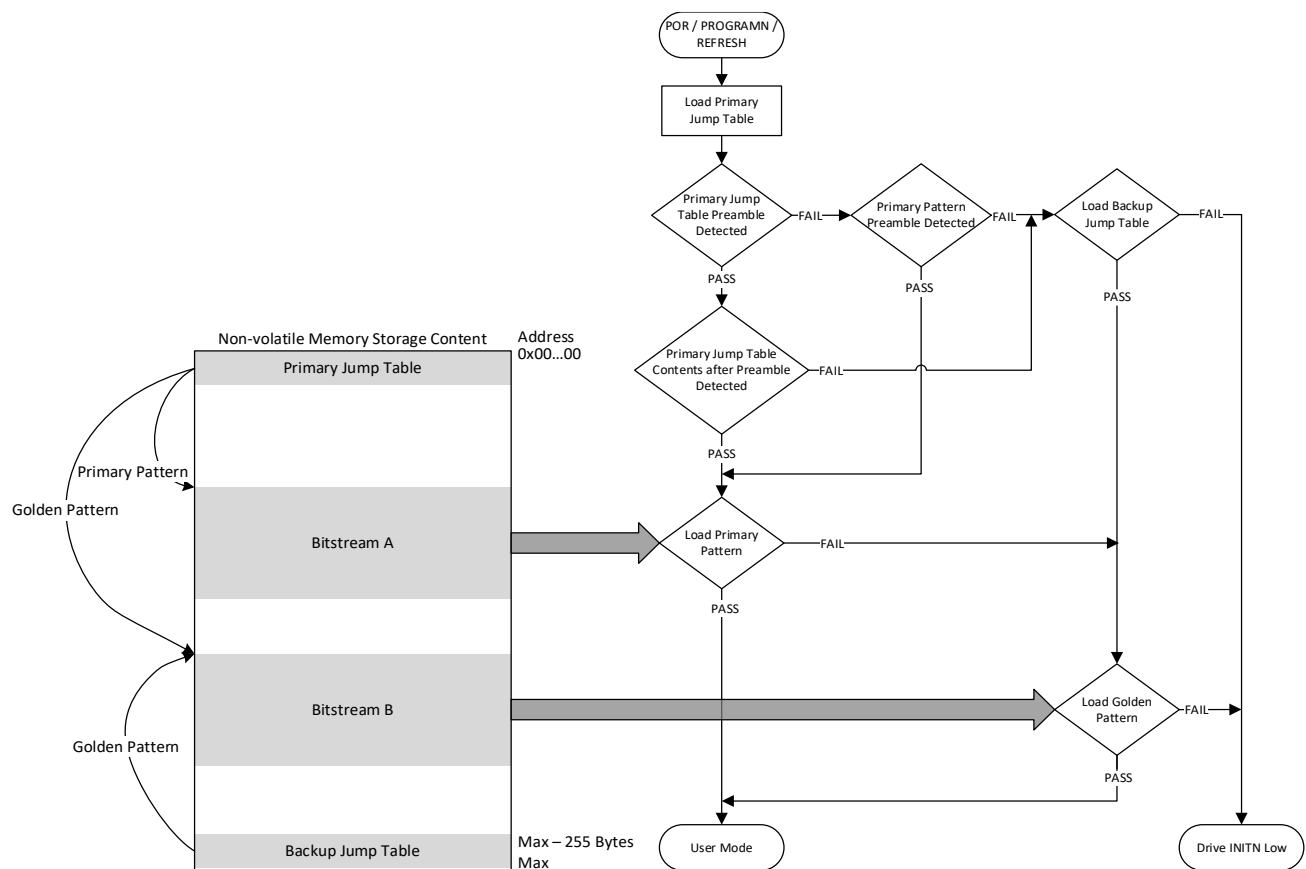
If the primary pattern fails to load, the Nexus 2 device drives the INITN pin low to indicate an error and resets the configuration engine. After clearing the configuration RAM, the device drives the INITN pin high and uses the golden pattern offset address in the primary jump table to load the golden pattern in the non-volatile memory storage device. If the backup jump table is corrupted or the golden pattern configuration fails to load, the Nexus 2 device stops driving the SPI clock and drives the INITN pin low.

Notes:

- The backup jump table is located at the last 256 bytes of the non-volatile memory storage device. When the configuration engine attempts to read the backup jump table, it issues a read offset address at 0xFFFFF00 for 24-bit addressing, 0xFFFF_FF00 for 32-bit addressing, or 0xFFFF_FFFF_FF00 for 45-bit addressing. The non-volatile memory truncates the most significant bit or bits of the address based on its density. For example, a 32-Mb flash receives the read offset address as 0x3FFF00 even though the configuration engine sends 0xFFFFF00.
- For the Certus-N2 device, by default, the 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row are set to 0. The MSPI controller operates in 3-byte command and addressing mode. This enables the device to support dual-boot, multi-boot, or ping-pong boot mode with configuration memory sizes up to 128 Mb. If your configuration memory size is 256 Mb or larger, the 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row must be set to 1.
- For the Mach-N2 device, by default, the 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row are set to 1. The Mach-N2 device contains 256 Mb of internal flash memory.

5. Ping-Pong Boot Mode

The Nexus 2 device ping-pong boot mode supports booting from one of two configuration bitstream patterns that reside in an external SPI storage device or internal flash memory. These patterns are designated the primary bitstream and secondary bitstream. The Nexus 2 device boots from the bitstream pattern based on the assignment in the jump table. The jump table allows the device to boot from either bitstream without changing the physical location of the two bitstreams within the external SPI storage device or internal flash memory. Only the jump table needs to be updated to change the boot bitstream pattern. When one bitstream is selected as the primary bitstream pattern, by default, the secondary bitstream becomes the golden bitstream pattern. The external SPI storage device or internal flash memory can also store a backup jump table, in case the primary jump table becomes corrupted. Figure 5.1 shows the ping-pong boot flow diagram.



Notes:

1. The primary jump table points to the addresses of the primary and golden patterns.
2. The backup jump table points to the golden pattern.
3. To change the designation of which bitstream is primary and which is golden, only the jump tables need to be updated.
4. Primary jump table contents include the *Program Secondary Boot Address* and JUMP command.
5. The primary pattern is loaded either through execution of the JUMP command in the primary jump table or directly when its preamble is detected but the primary jump table preamble is not detected.
6. The golden pattern is loaded either through execution of the JUMP command in the backup jump table or execution of the *Program Secondary Boot Address* command from the primary jump table.

Figure 5.1. Ping-Pong Boot Flow Diagram

5.1. Ping-Pong Boot Flow

The ping-pong boot flow is triggered at power up, PROGRAMN pin pulsing, or REFRESH command execution. When a JUMP command in the primary jump table is executed, the device stops the SPI clock, drives the INITN pin low, resets the configuration engine, and clears the configuration RAM. After the configuration RAM is cleared, the device drives the INITN pin high, restarts the SPI clock, and reads the primary pattern from the non-volatile memory storage device as addressed by the jump table.

In ping-pong boot mode, the bitstream pattern selected might fail to load for one of the following reasons:

- Time-out check — Search for a 32-bit preamble code (0xFFFFBxB3) in the bitstream pattern, as part of the configuration protocol, fails. Refer to [Table 4.1](#) for the time-out values.
- Device ID check — ID check at the beginning of the bitstream fails.
- Data integrity check — CRC checks detect data corruption in the bitstream.
- Command validity check — An illegal command is encountered.

If the primary pattern fails to load, the Nexus 2 device drives the INITN pin low to indicate an error and resets the configuration engine. After clearing the configuration RAM, the device drives the INITN pin high and reads the address of the golden pattern.

For the golden pattern, if the primary jump table and primary pattern are corrupted (both primary jump table preamble and primary pattern preamble are not detected), the device loads the backup jump table, which then loads the golden pattern as addressed by the jump table. If only the primary pattern is corrupted, the device loads the golden pattern as addressed through the *Program Secondary Boot Address* command.

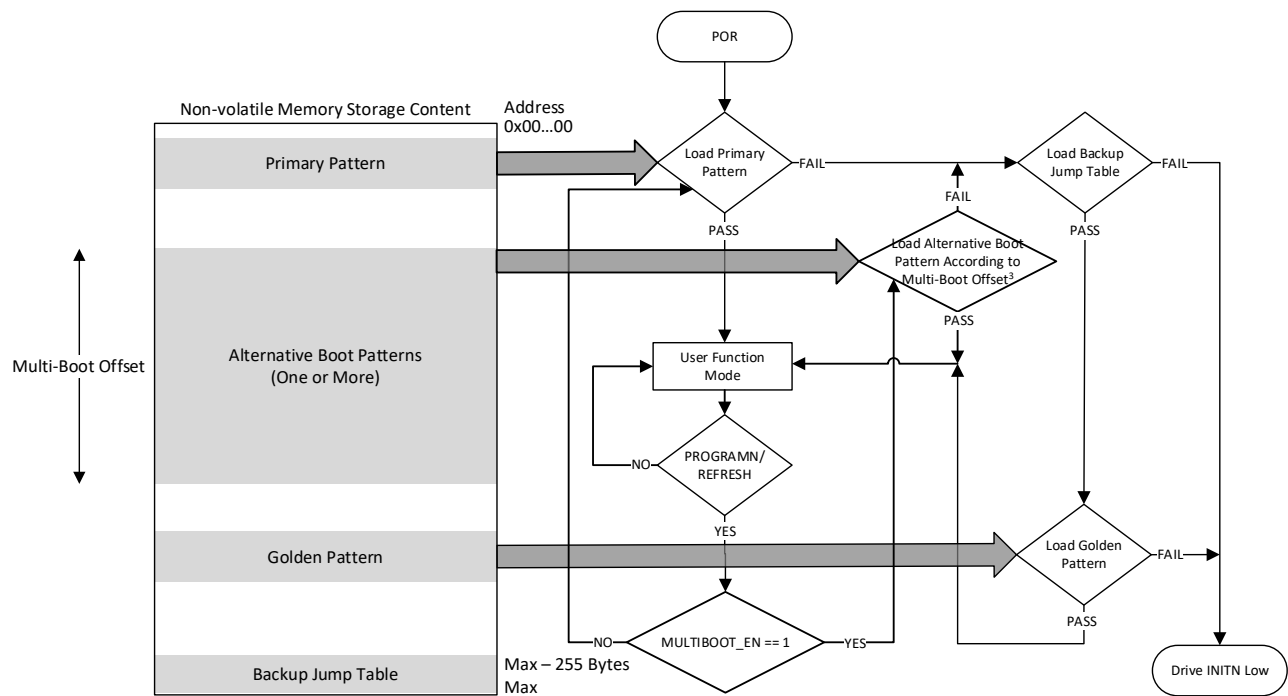
If the backup jump table is corrupted or the golden pattern fails to load, the Nexus 2 device stops driving the SPI clock and drives the INITN pin low.

Notes:

- The backup jump table is located at the last 256 bytes of the non-volatile memory storage device. When the configuration engine attempts to read the backup jump table, it issues a read offset address at 0xFFFFF00 for 24-bit addressing, 0xFFFF_FF00 for 32-bit addressing, or 0xFFFF_FFFF_FF00 for 45-bit addressing. The non-volatile memory truncates the most significant bit or bits of the address based on its density. For example, the 32-Mb flash receives the read offset address as 0x3FFF00 even though the configuration engine sends 0xFFFFF00.
- For the Certus-N2 device, by default, the 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row are set to 0. The MSPI controller operates in 3-byte command and addressing mode. This enables the device to support dual-boot, multi-boot, or ping-pong boot mode with configuration memory sizes up to 128 Mb. If your configuration memory size is 256 Mb or larger, the 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row must be set to 1.
- For the Mach-N2 device, by default, the 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row are set to 1. The Mach-N2 device contains 256 Mb of internal flash memory.

6. Multi-Boot Mode

The Nexus 2 device multi-boot mode supports booting from at least three and up to 18 configuration bitstream patterns that reside in an external SPI storage device or internal flash memory. These patterns are designated the primary pattern, golden pattern, and alternative boot patterns. When a Nexus 2 device boots up, it attempts to boot from the primary pattern. If the primary pattern fails to load, the device attempts to boot from the golden pattern. After successful configuration, the MULTI_BOOT_SEL attribute determines the subsequent loading of alternative boot patterns. Figure 6.1 shows the multi-boot flow diagram.



Notes:

1. The default primary boot offset is at address 0.
2. When MULTI_BOOT_SEL == STATIC, the multi-boot offset uses the value stored in the Multiboot Offset register. When MULTI_BOOT_SEL == DYNAMIC, the multi-boot offset uses the multi-boot address register programmed through the CONFIG_LMMI interface.
3. Load the primary pattern if the multi-boot offset points to the start address of the primary pattern in the SPI storage device.

Figure 6.1. Multi-Boot Flow Diagram

6.1. Multi-Boot Flow

If MULTI_BOOT_SEL == STATIC, after the primary pattern loads, reprogramming of the bitstream can be done by performing PROGRAMN pin pulsing or REFRESH command execution, which then triggers the loading of alternative boot pattern 1. Subsequent PROGRAMN pin pulsing or REFRESH command execution events load the next pattern defined in the multi-boot configuration. The bitstream pattern sequence, target address of the golden pattern, and target addresses of the alternative boot patterns are defined during the multi-boot configuration process in the Lattice Radiant™ Deployment Tool.

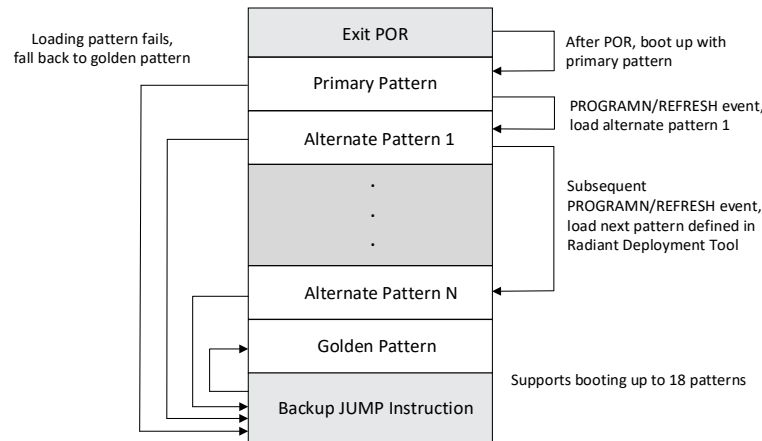


Figure 6.2. Static Multi-Boot Mode Flow

If MULTI_BOOT_SEL == DYNAMIC, after the primary pattern loads, you can switch between all the different patterns stored in the external or internal storage device freely.

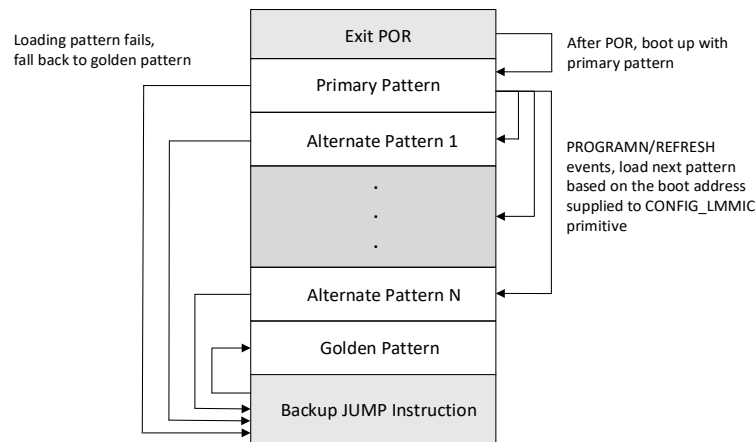


Figure 6.3. Dynamic Multi-Boot Mode Flow

If the primary pattern or alternative boot pattern fails to load, the device drives the INITN pin low to indicate an error and resets the configuration engine. After clearing the configuration RAM, the device drives the INITN pin high and reads the JUMP command in the backup jump table that directs it to load the golden pattern in the non-volatile memory storage device. If the golden pattern configuration fails, the Nexus 2 device stops driving the SPI clock and drives the INITN pin low. Once golden pattern configuration fails, you must reprogram the external SPI storage device or internal flash memory through the JTAG interface (using the JTAG-to-Controller SPI Bridge) or target SPI port (using the Target SPI-to-Controller SPI Bridge).

Notes:

- The backup jump table is located at the last 256 bytes of the non-volatile memory storage device. When the configuration engine attempts to read the backup jump table, it issues a read offset address at 0xFFFFF00 for 24-bit addressing, 0xFFFF_FF00 for 32-bit addressing, or 0xFFFF_FFFF_FF00 for 45-bit addressing. The non-volatile memory truncates the most significant bit or bits of the address based on its density. For example, the 32-Mb flash receives the read offset address as 0x3FFF00 even though the configuration engine sends 0xFFFFF00.
- For the Certus-N2 device, by default, the 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row are set to 0. The MSPI controller operates in 3-byte command and addressing mode. This enables the device to support dual-boot, multi-boot, or ping-pong boot mode with configuration memory sizes up to 128 Mb. If your configuration memory size is 256 Mb or larger, the 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row must be set to 1.
- For the Mach-N2 device, by default, the 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row are set to 1. The Mach-N2 device contains 256 Mb of internal flash memory.

The software selection options for the multi-boot mode can be set through the Global tab of the Device Constraint Editor in the Lattice Radiant software.

- **MULTI_BOOT_MODE:** Enables multi-boot mode.
- **MULTI_BOOT_SEL:** Selects the booting address for the multiple boot event when MULTI_BOOT_MODE is set to ENABLE.
 - **STATIC (default):** Use the value stored in the MultiBoot Offset register as the booting address. This bitstream pattern booting address value can be preset through the Global tab of the Device Constraint Editor. Set these values for the bitstream patterns of your boot sequence before you use the Lattice Radiant Deployment Tool to create the multi-boot hex file. Refer to the MultiBoot Offset section in the [Lattice Nexus 2 sysCONFIG User Guide \(FPGA-TN-02370\)](#) for more information about the MultiBoot Offset register.
 - **DYNAMIC:** Use the multi-boot address register programmed through the CONFIG_LMMI interface.
- **MultiBoot Offset:** A 48-bit register stores the boot address for the next pattern to load from the non-volatile memory. When MULTI_BOOT_SEL is set to STATIC, the device uses the value stored in this register as the booting address for the next pattern.

Note: If you use the STATIC setting for the MULTI_BOOT_SEL option, whenever updating a particular alternate pattern in the non-volatile memory, ensure the corresponding bitstream pattern booting address value is updated accordingly, if necessary, through the Device Constraint Editor. A booting address value update is necessary when the bitstream pattern location changes.

6.2. Implementation of Static or Dynamic Multi-Boot Mode

To implement static or dynamic multi-boot mode, the CONFIG_LMMIC primitive as shown in [Figure 6.4](#) must be incorporated into your design.

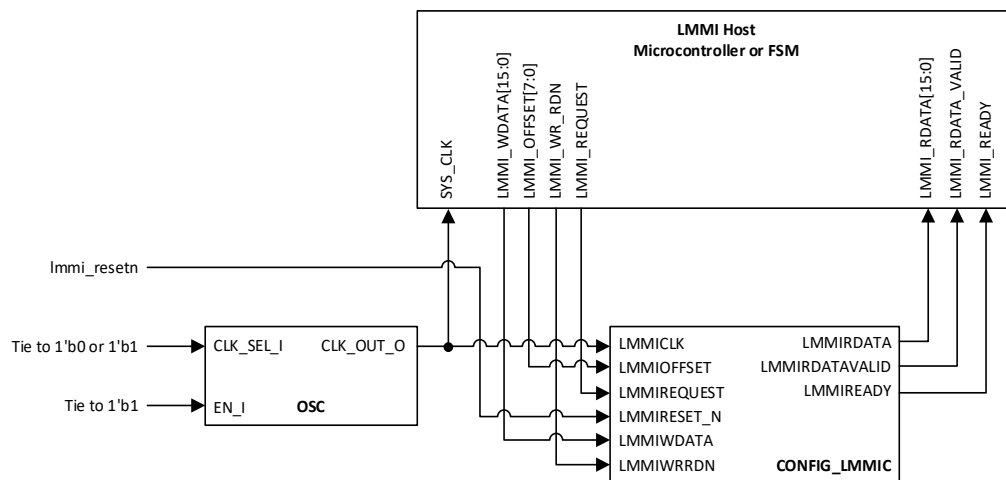


Figure 6.4. CONFIG_LMMIC Primitive for Multi-Boot Implementation

The following are the blocks or modules required for this implementation:

- **OSC:** The oscillator module generates the clock source for the LMMI Host and CONFIG_LMMIC primitive.
- **CONFIG_LMMIC:** The CONFIG_LMMIC primitive adds the Lattice memory mapped interface (LMMI) to the configuration block. Refer to the CONFIG_LMMIC page in the Lattice Radiant Software Help for more information.
- **LMMI Host:** The LMMI Host implements a state machine controller to send the necessary commands to the CONFIG_LMMIC primitive to boot the desired alternate pattern stored in an external SPI flash or internal flash memory. The controller performs the sequence to load the pattern for the static multi-boot mode or dynamic multi-boot mode.

6.2.1. Static Multi-Boot Mode Host Controller Sequence

The following is the controller sequence for loading the pattern in static multi-boot mode, with the flow as shown in Figure 6.5:

1. **LMMI_CFG_PORT_REQUEST[15:0]** and **LMMI_CFG_PORT_REQUEST[31:16]**: Executes the port request command to enable exclusive LMMI access to the configuration engine. Write **LMMI_CFG_PORT_REQUEST[15:0]** first followed by **LMMI_CFG_PORT_REQUEST[31:16]**. Failure to follow this sequence might cause the LMMI port to become inactive, thereby requiring power cycling of the device to recover the LMMI port.
2. **LMMI_CFG_PORT_STATUS[15:0]**: Reads the port status for the CONFIG_LMMI interface and proceeds to the next state once exclusive access is granted.
3. **REFRESH:** Equivalent to pulsing the PROGRAMN pin. Once this command is executed, the device starts loading the desired alternate pattern from the external SPI flash or internal flash memory according to the configuration sequence defined in the multi-boot configuration process in the Lattice Radiant Deployment Tool. If loading of the pattern fails, the device falls back to loading the golden pattern.

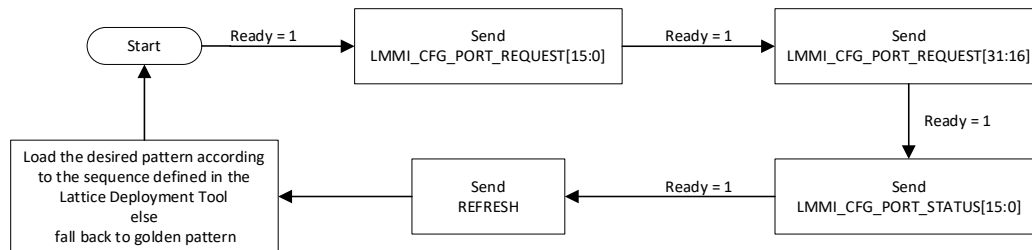


Figure 6.5: Host Controller Sequence for Static Multi-Boot Mode

For more information on configuration access through LMMI and target configuration commands, refer to the LMMI CFG Offset List and Target Configuration Commands tables, respectively, in the [Lattice Nexus 2 sysCONFIG User Guide \(FPGA-TN-02370\)](#).

6.2.2. Dynamic Multi-Boot Mode Host Controller Sequence

The following is the controller sequence for loading the pattern in dynamic multi-boot mode, with the flow as shown in Figure 6.6:

1. **LMMI_CFG_PORT_REQUEST[15:0]** and **LMMI_CFG_PORT_REQUEST[31:16]**: Executes the port request command to enable exclusive LMMI access to the configuration engine. Write **LMMI_CFG_PORT_REQUEST[15:0]** first followed by **LMMI_CFG_PORT_REQUEST[31:16]**. Failure to follow this sequence might cause the LMMI port to become inactive, thereby requiring power cycling of the device to recover the LMMI port.
2. **LMMI_CFG_PORT_STATUS[15:0]**: Reads the port status for the CONFIG_LMMI interface and proceeds to the next state once exclusive access is granted.
3. **LMMI_CFG_MSPI_MULTIBOOT_ADDR [15:0]** and **LMMI_CFG_MSPI_MULTIBOOT_ADDR [31:16]**: Sends the start address through the CONFIG_LMMIC primitive. The start address specifies the location in the external SPI flash or internal flash memory where the desired bitstream is stored for booting and configuring the device.

4. REFRESH: Equivalent to pulsing the PROGRAMN pin. Once this command is executed, the device starts loading the desired alternate pattern from the external SPI flash or internal flash memory according to the boot address sent in the previous state. If loading of the pattern fails, the device falls back to loading the golden pattern.

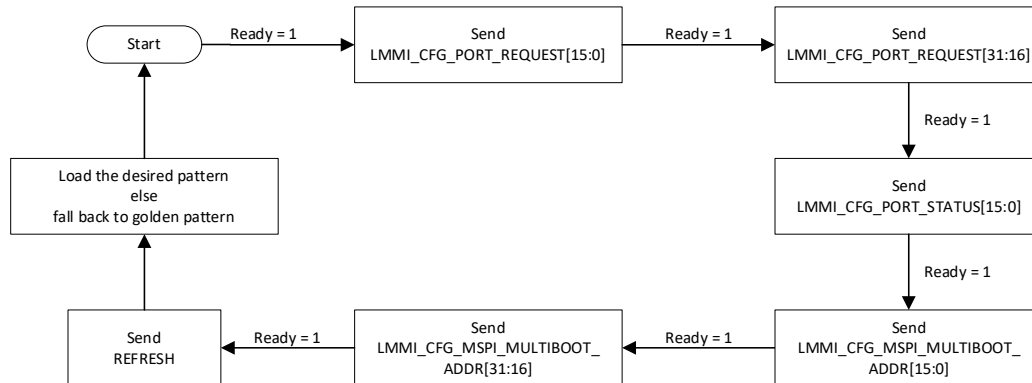


Figure 6.6. Host Controller Sequence for Dynamic Multi-Boot Mode

For more information on configuration access through LMMI and target configuration commands, refer to the LMMI CFG Offset List and Target Configuration Commands tables, respectively, in the [Lattice Nexus 2 sysCONFIG User Guide \(FPGA-TN-02370\)](#).

7. Creating an SPI Storage Device Hex File – For Certus-N2 Only

The Lattice Radiant software provides a turnkey solution to manage the low-level implementation details of the dual-boot, ping-pong boot, and multi-boot features on Certus-N2 devices. The Lattice Radiant Deployment Tool, which is part of Lattice Radiant software, merges the different patterns and the JUMP command into one SPI storage device hex file. The SPI storage device hex file can later be programmed into an external SPI storage device using the Radiant Programmer or a third-party programmer.

7.1. Using Radiant Deployment Tool to Create a Dual-Boot Hex File

This section provides the procedure for generating a dual-boot SPI storage device hex file using the Radiant Deployment Tool.

- Generate the golden and primary bitstream files in the Lattice Radiant software.
 - MCCLK_FREQ (MCLKP frequency) setting of bitstream files should not exceed the external SPI storage device normal or standard read speed.
 - MCCLK_FREQ can be configured using the Global tab of the Device Constraint Editor in the Lattice Radiant software.
- Invoke the Lattice Radiant Deployment Tool by choosing **Tools > Deployment Tool** from the Lattice Radiant Programmer.
- In the Radiant Deployment Tool window, select **External Memory** as the **Function Type** and select **Dual Boot** as the **Output File Type** (Figure 7.1).
- Click **OK**.

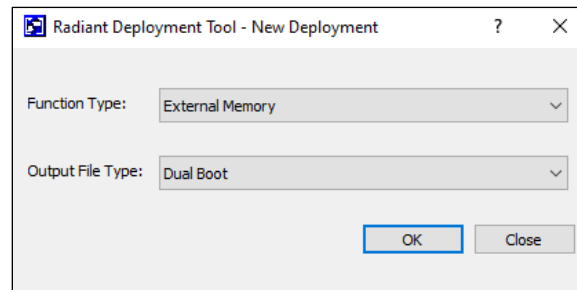


Figure 7.1. Creating New Deployment for Dual Boot File

Step 1 of 4: Select Input File(s) window (Figure 7.2)

- Click on the ... button in the **File Name** fields to browse and select the two bitstream files to be used to create the SPI storage device hex file.
- The **Device Family** and **Device** fields automatically populate based on the bitstream files selected.
- Click **Next**.

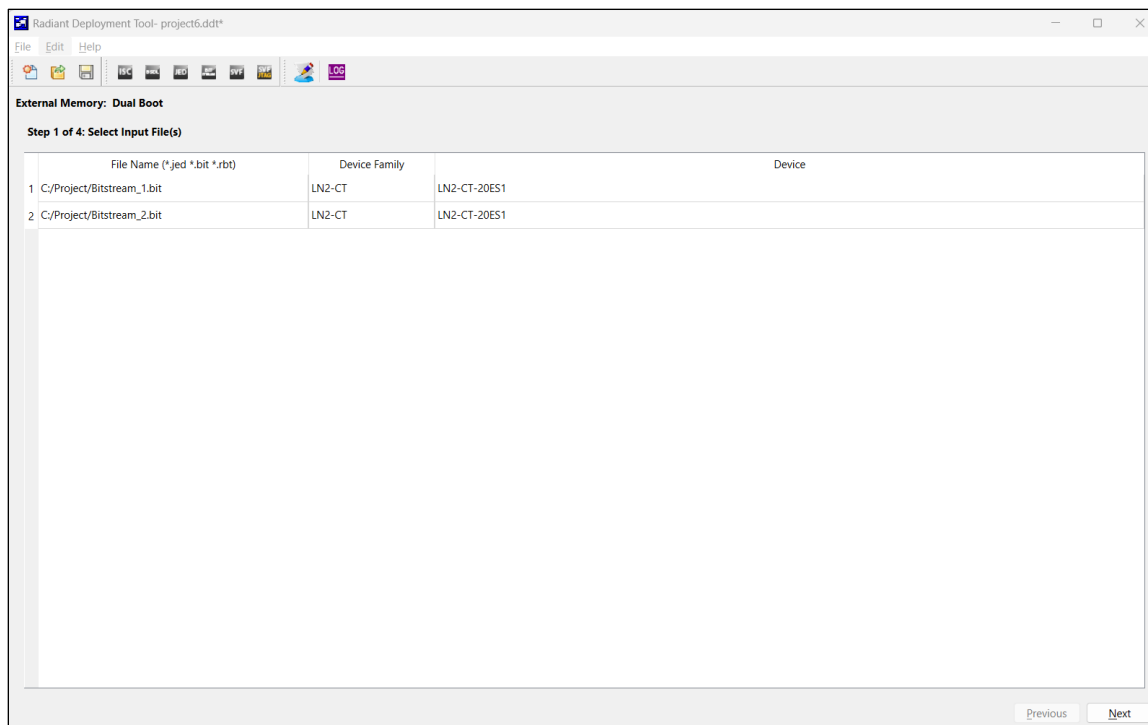


Figure 7.2. Select Input File(s) Window

Step 2 of 4: Dual Boot Options window (Figure 7.3)

- Select the **Output Format** (Intel Hex, Motorola Hex, or Extended Tektronix Hex).
- Select the **SPI Flash Size (Mb)** (4, 8, 16, 32, 64, 128, 256, 512, or 1024 Mb).
- Select **SPI Flash Read Mode** (Standard Read, Fast Read, Dual I/O SPI Flash Read, Quad I/O SPI Flash Read, xSPI Profile 1 Read, and xSPI Profile 2 Read).
- Select the **Clock Mode**.
 - When **SPI Flash Read Mode** is set to Standard Read, Fast Read, Dual I/O SPI Flash Read, or Quad I/O SPI Flash Read, only the Single-ended Clock option is available.
 - When SPI Flash **Read Mode** is set to xSPI Profile 1 Read or xSPI Profile 2 Read, the *Single-ended Clock* and *Differential Clock* options are available.
- The Radiant Deployment Tool automatically assigns the bitstream files selected in Step 1 to be used for the golden pattern and primary pattern.
 - Change the pattern options by clicking on the drop-down menu of the respective fields.
 - The **Starting Address** of the **Golden Pattern** is automatically assigned.
 - Change the **Starting Address** of the **Golden Pattern** by clicking on the drop-down menu.
- Select the following options as required:
 - **Byte Wide Bit Mirror** – Flips each byte in Intel, Extended Tektronix, or Motorola hexadecimal data files. For example, 0xCD (b1100 1101) becomes 0xB3 (b1011 0011) when this is selected.
 - **Retain Bitstream Header** – By default, Radiant Deployment Tool replaces the bitstream header information (name, version number, and date of the file) with 0xFF values. Selecting this option retains the header information that is generated as the header.
 - **Optimize Memory Space** – When checked, it uses the bitstream file size instead of the worst-case bitstream size.
 - **SFDP Enabled** – Enables the checking of Serial Flash Discoverable Parameter (SFDP) signature during controller SPI booting. This option must be enabled for flash devices that support SFDP.
 - **Use Read Mode Settings for Golden Pattern** – Appears when **SPI Flash Read Mode** is set to Dual I/O SPI Flash Read or Quad I/O SPI Flash Read. When checked, the JUMP command in the jump table operates using the SPI flash read mode specified by the **SPI Flash Read Mode** setting instead of the default serial (x1) read mode. For example, if **SPI Flash Read Mode** is set to *Quad I/O SPI Flash Read*, the JUMP command loads the golden pattern using the quad I/O read mode. This allows the golden image to be loaded using the faster dual I/O or quad I/O read mode.
 - **Encryption** – Enables encryption of an unencrypted bitstream. After enabling, click on the **Edit Key** button and select the encryption key to be used.
 - **ECDSA Authentication** – Enables bitstream ECDSA authentication. After enabling, select **Authentication Mode** from the drop-down list, then load the **Public Key** and the **Private Key**.
 - **RSA Authentication** – Enables bitstream RSA Authentication. After enabling, select **Authentication Mode** from the drop-down list, then load the **Public Key** and the **Private Key**.
- (Optional) If programming the flash memory with user data is required, click the **User Data Files** tab (Figure 7.4).
 - Check **User Data Files** to enable the insertion of the user data file.
 - Select the **Number of User Data Files** from the drop-down list.
 - Click the ... (browse) button associated with a user data file field, navigate to, and then select the user data file (*.bit, *.rbit, *.bin, *.mcs, *.exo, *.xtek) to be programmed into the SPI storage device.
 - Select the starting address to store the user data file from the **Starting Address** drop-down list.
- Click **Next**.

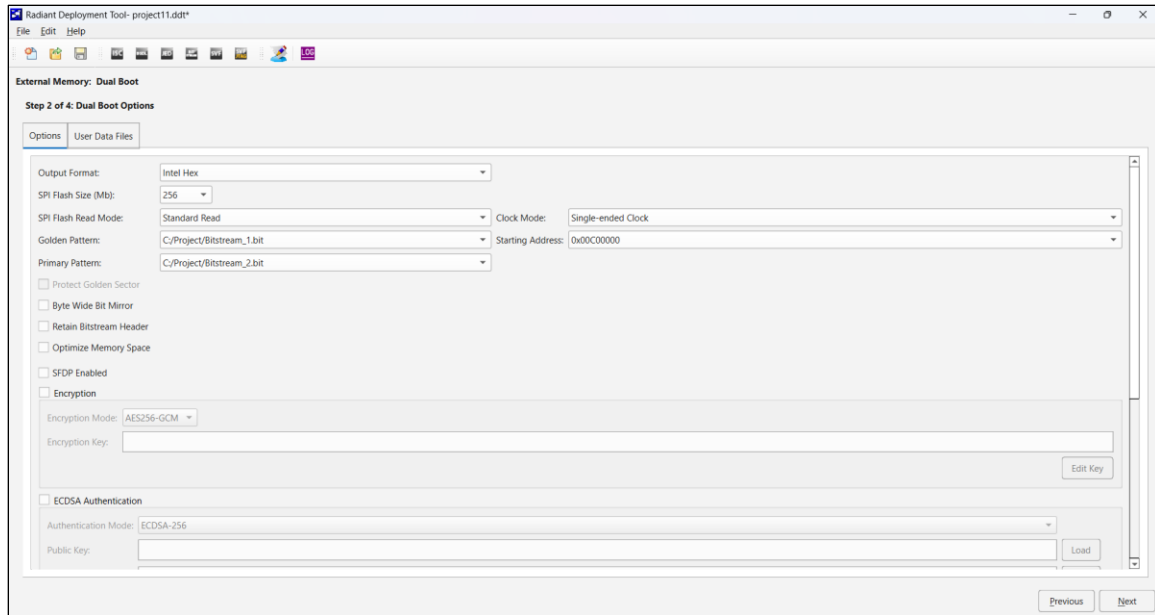


Figure 7.3. Dual Boot Options Window

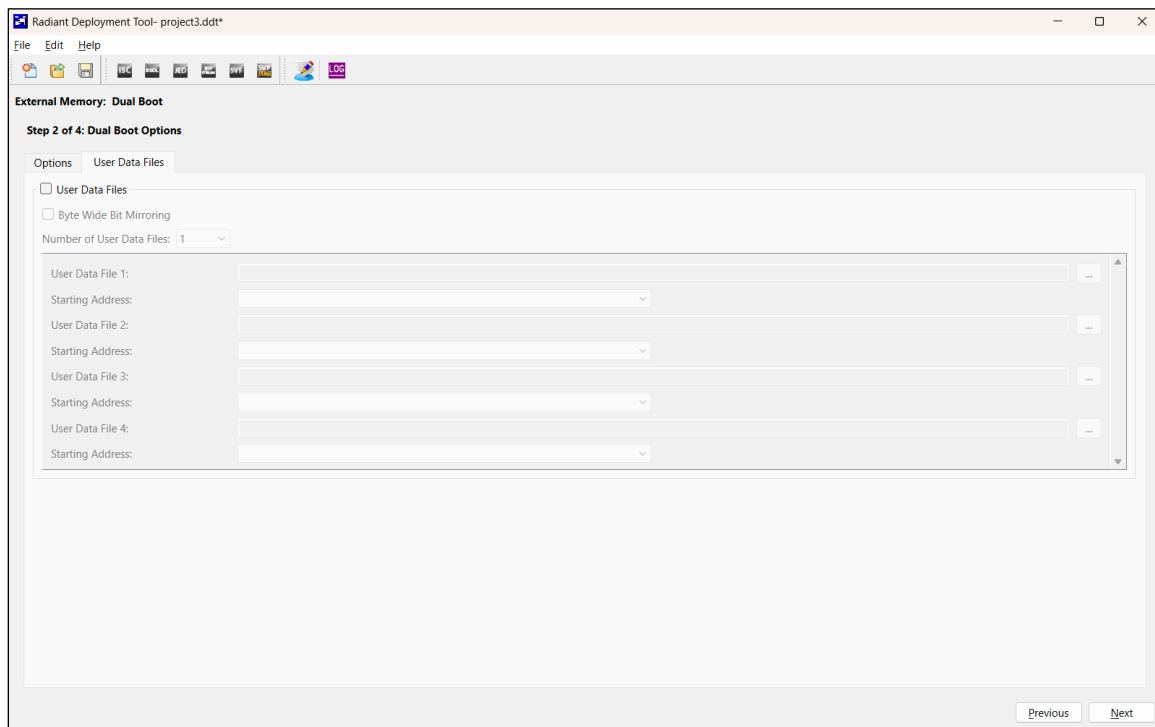


Figure 7.4. Dual Boot Options Window – User Data Files Tab

Step 3 of 4: Select Output File(s) window (Figure 7.5)

- Specify the name and location of the output SPI storage device hex file in the **Output File 1** field.
- Click **Next**.

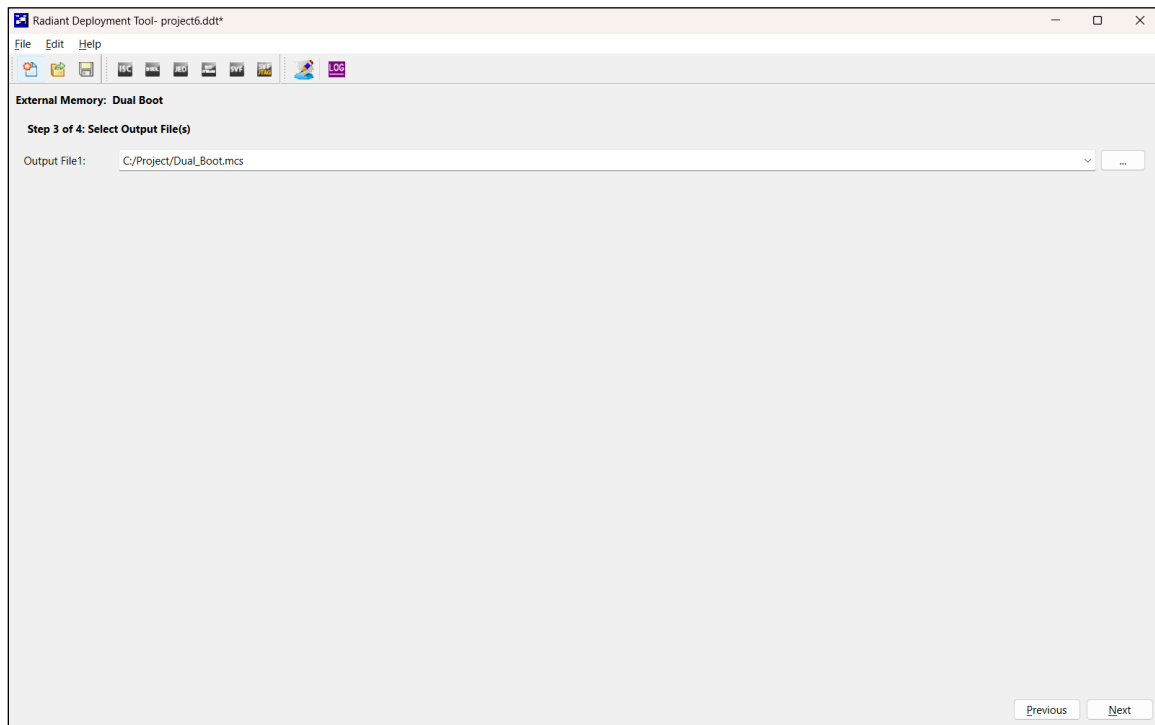


Figure 7.5. Select Output File(s) Window

Step 4 of 4: Generate Deployment window (Figure 7.6)

- Review the summary information.
- If everything is correct, click the **Generate** button.
- The **Generate Deployment** pane should indicate that the SPI storage device file is generated successfully.
- The generated .mcs file is located in the same directory as the two bitstream files.
- Save the deployment settings by selecting **File > Save**.
- To exit, select **File > Exit**.

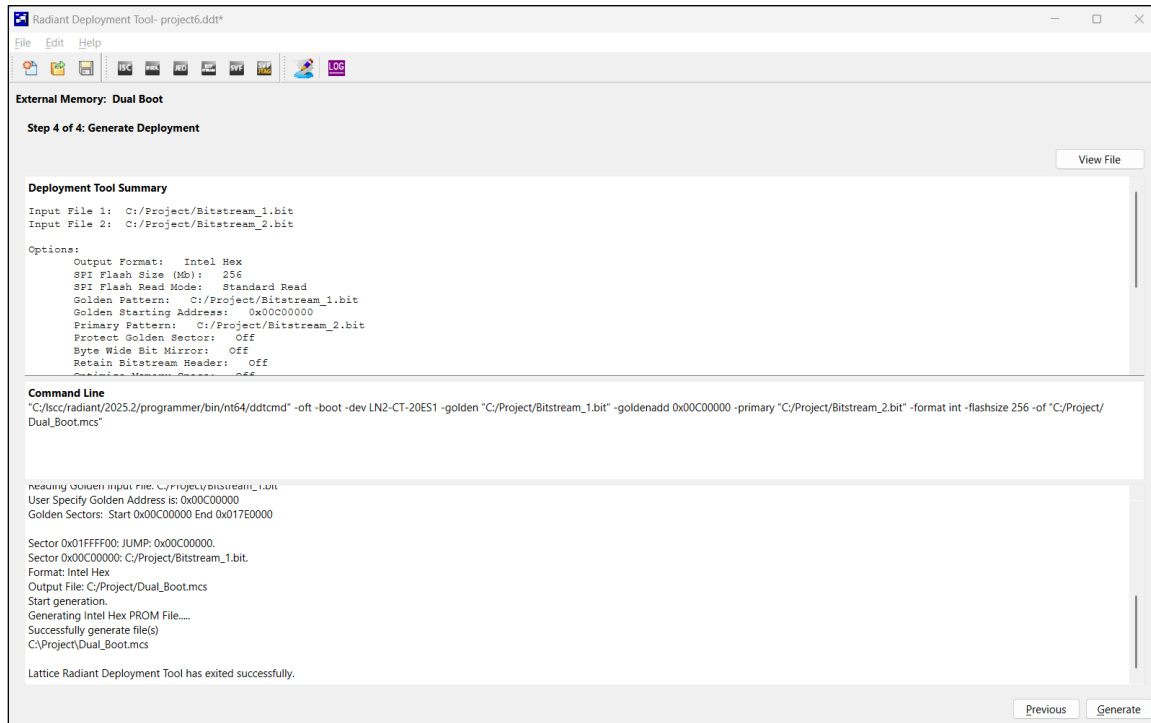


Figure 7.6. Generate Deployment Window

7.2. Using Radiant Deployment Tool to Create Ping-Pong Boot Hex File

This section provides the procedure for generating a ping-pong boot SPI storage device hex file using the Radiant Deployment Tool.

- Generate the two bitstream files in the Lattice Radiant software.
 - MCCLK_FREQ (MCLKP frequency) setting of bitstream files should not exceed the external SPI storage device normal or standard read speed.
 - MCCLK_FREQ can be configured using the Global tab of the Device Constraint Editor in the Lattice Radiant software.
- Invoke the Lattice Radiant Deployment Tool by choosing **Tools > Deployment Tool** from the Lattice Radiant Programmer.
- In the Radiant Deployment Tool window, select **External Memory** as the **Function Type** and select **Ping-Pong Boot** as the **Output File Type** (Figure 7.7).
- Click **OK**.

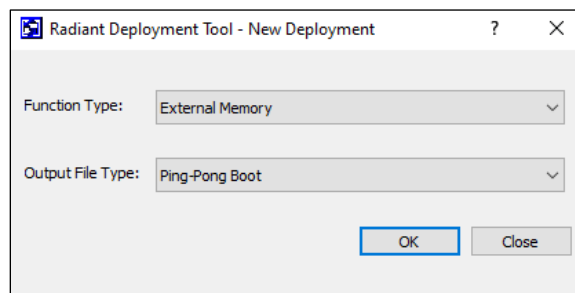


Figure 7.7. Creating New Deployment for Ping-Pong Boot File

Step 1 of 4: Select Input File(s) window (Figure 7.8)

- Click on the ... button in the **File Name** fields to browse and select the two bitstream files to be used to create the SPI storage device hex file.
- The **Device Family** and **Device** fields automatically populate based on the bitstream files selected.
- Click **Next**.

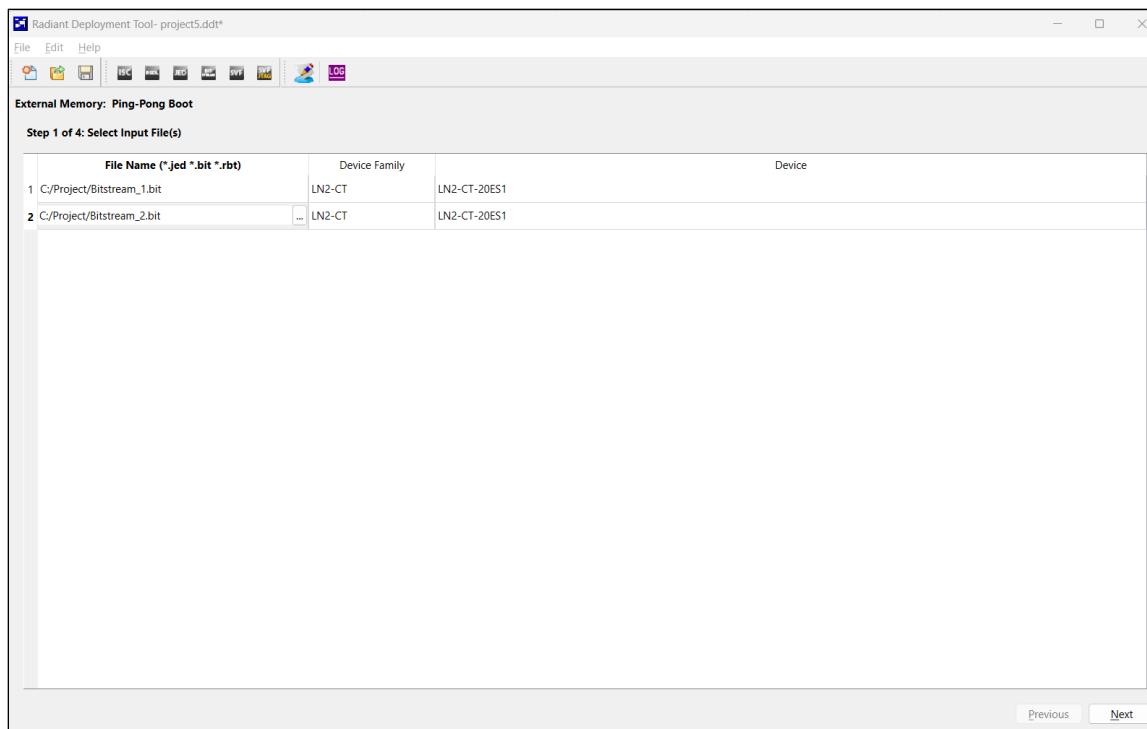


Figure 7.8. Select Input File(s) Window

Step 2 of 4: Ping-pong Boot Options window (Figure 7.9)

- Select the **Output Format** (Intel Hex, Motorola Hex, or Extended Tektronix Hex).
- Select the **SPI Flash Size (Mb)** (4, 8, 16, 32, 64, 128, 256, 512, or 1024 Mb).
- Select **SPI Flash Read Mode** (Standard Read, Fast Read, Dual I/O SPI Flash Read, Quad I/O SPI Flash Read, xSPI Profile 1 Read, or xSPI Profile 2 Read).
- Select the **Clock Mode**.
 - When **SPI Flash Read Mode** is set to Standard Read, Fast Read, Dual I/O SPI Flash Read, or Quad I/O SPI Flash Read, only the Single-ended Clock option is available.
 - When SPI Flash Read Mode is set to xSPI Profile 1 Read or xSPI Profile 2 Read, the Single-ended Clock and Differential Clock options are available.
- The Radiant Deployment Tool automatically assigns the bitstream files selected in Step 1 to be used for the primary pattern and secondary pattern.
 - Change the pattern options by clicking on the drop-down menu of the respective fields.
 - The **Starting Address** fields of the **Primary Pattern** and **Secondary Pattern** are automatically assigned.
 - Change the **Starting Address** of the patterns by clicking on the drop-down menu.
- Select the following options as required:
 - **Generate Jump Table Only** – Generates jump table only.
 - **Byte Wide Bit Mirror** – Flips each byte in Intel, Extended Tektronix, or Motorola hexadecimal data files. For example, 0xCD (b1100 1101) becomes 0xB3 (b1011 0011) when this is selected.
 - **Retain Bitstream Header** – By default, Radiant Deployment Tool replaces the bitstream header information (name, version number, and date of the file) with 0xFF values. Selecting this option retains the header information that is generated as the header.
 - **SFDP Enabled** – Enables the checking of SFDP signature during controller SPI booting. This option must be enabled for flash devices that support SFDP.
 - **Use Read Mode Settings for Golden Pattern** – Appears when **SPI Flash Read Mode** is set to Dual I/O SPI Flash Read or Quad I/O SPI Flash Read. When checked, the JUMP command in the backup jump table operates using the SPI flash read mode specified by the **SPI Flash Read Mode** setting instead of the default serial (x1) read mode. For example, if **SPI Flash Read Mode** is set to *Quad I/O SPI Flash Read*, the JUMP command loads the golden pattern using the quad I/O read mode. This allows the golden image to be loaded using the faster dual I/O or quad I/O read mode.
 - **Encryption** – Enables encryption of an unencrypted bitstream. After enabling, click on the **Edit Key** button and select the encryption key to be used.
 - **ECDSA Authentication** – Enables bitstream ECDSA authentication. After enabling, select **Authentication Mode** from the drop-down list, then load the **Public Key** and the **Private Key**.
 - **RSA Authentication** – Enables bitstream RSA Authentication. After enabling, select **Authentication Mode** from the drop-down list, then load the **Public Key** and the **Private Key**.
- (Optional) If programming the flash memory with user data is required, click the **User Data Files** tab (Figure 7.10).
 - Check **User Data Files** to enable the insertion of the user data file.
 - Select the **Number of User Data Files** from the drop-down list.
 - Click the ... (browse) button associated with a user data file field, navigate to, and then select the user data file (*.bit, *.rbit, *.bin, *.mcs, *.exo, *.xtek) to be programmed into the SPI storage device.
 - Select the starting address to store the user data file from the **Starting Address** drop-down list.
- Click **Next**.

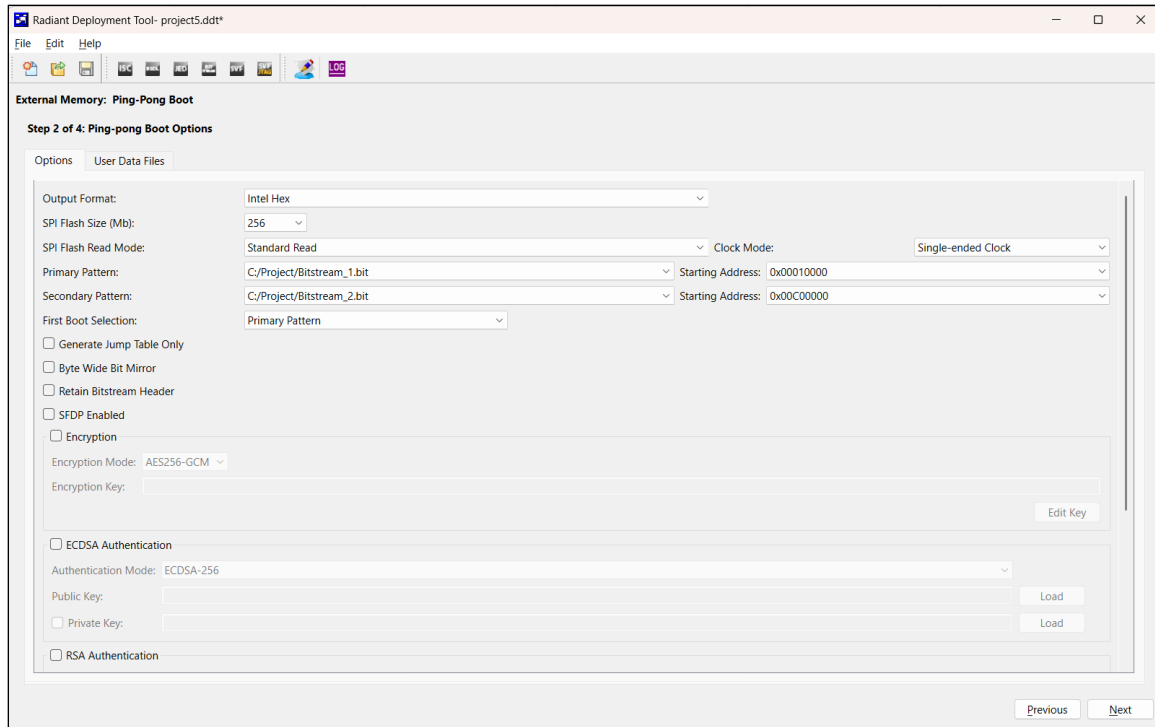


Figure 7.9. Ping-Pong Boot Options Window

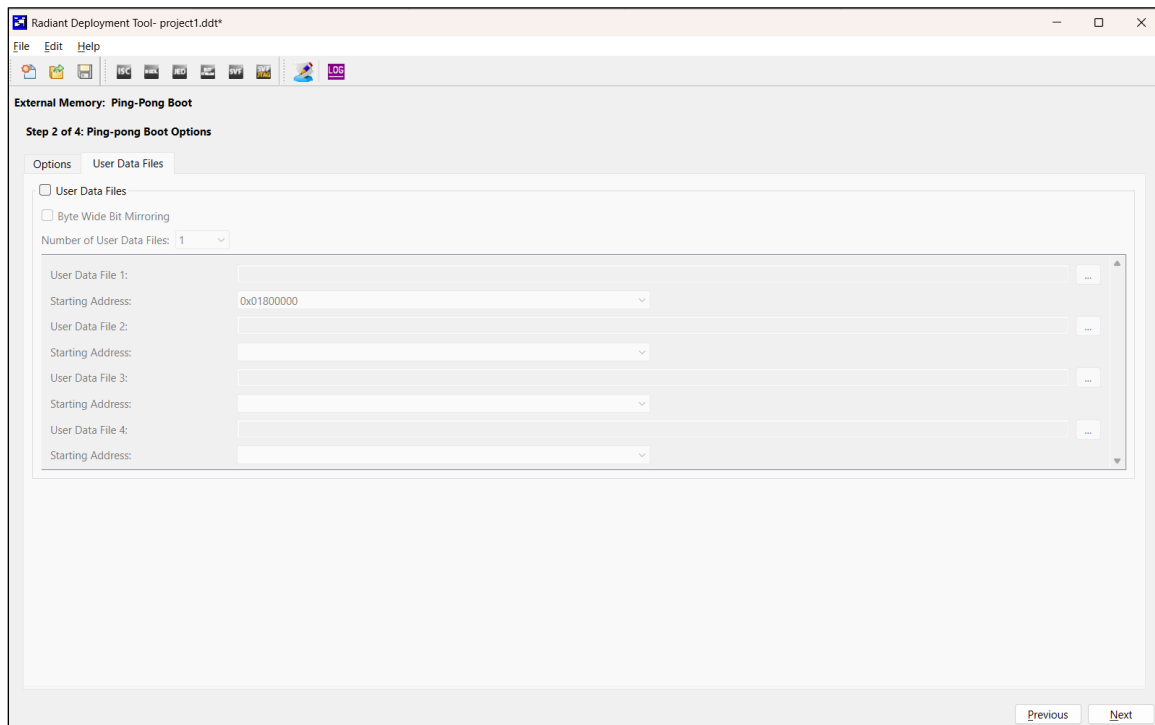


Figure 7.10. Ping-Pong Boot Options Window – User Data Files Tab

Step 3 of 4: Select Output File(s) window (Figure 7.11)

- Specify the name and location of the output SPI storage device hex file in the **Output File 1** field.
- Click **Next**.

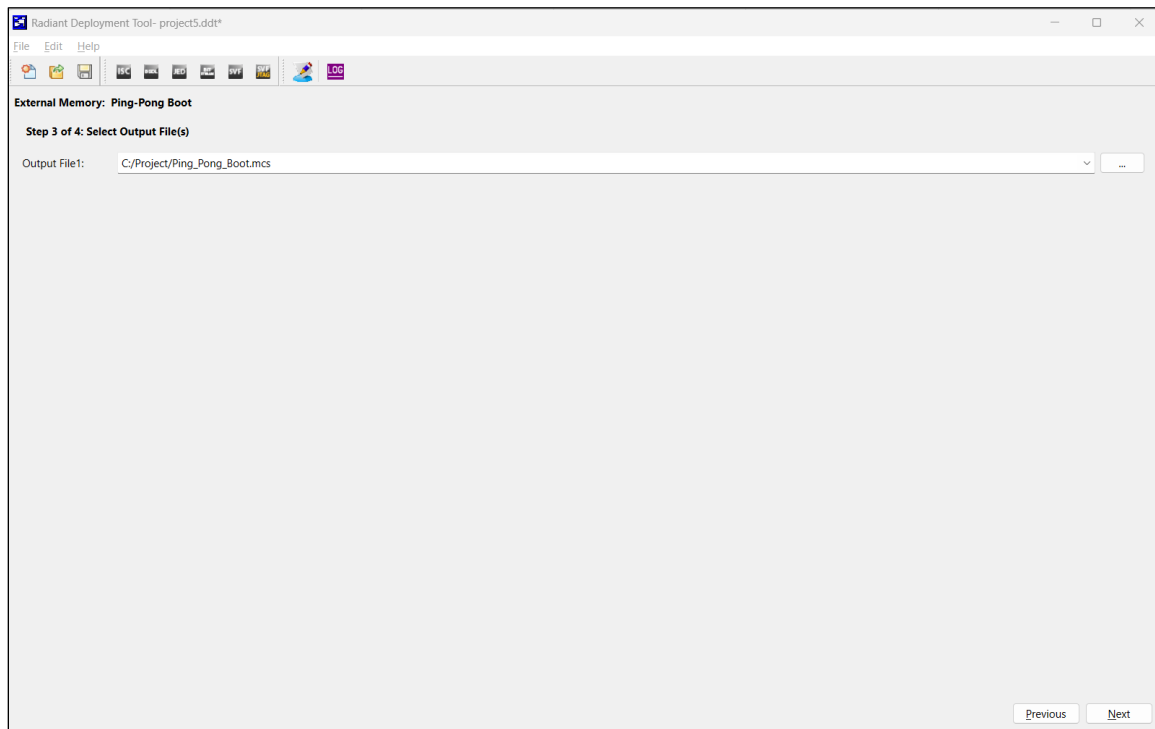


Figure 7.11. Select Output File(s) Window

Step 4 of 4: Generate Deployment window (Figure 7.12).

- Review the summary information.
- If everything is correct, click the **Generate** button.
- The **Generate Deployment** pane should indicate that the SPI storage device file is generated successfully.
- The generated .mcs file is located in the same directory as the two bitstream files.
- Save the deployment settings by selecting **File > Save** or **File > Save As....**
- To exit, select **File > Exit**.

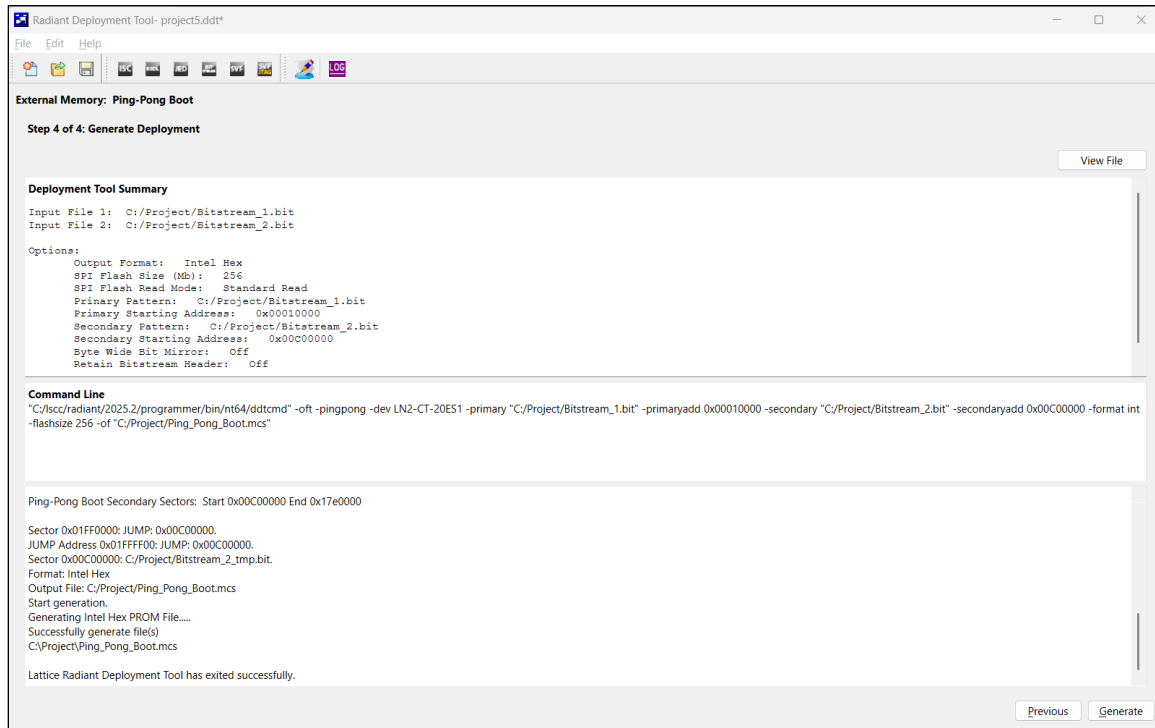


Figure 7.12. Generate Deployment Window

7.3. Using Radiant Deployment Tool to Create Multi-Boot Hex File

This section provides the procedure for generating a multi-boot SPI storage device hex file using the Radiant Deployment Tool.

- Generate the number of bitstream files needed in the Lattice Radiant software.
 - MCCLK_FREQ (MCLKP frequency) setting of bitstream files should not exceed the external SPI storage device normal or standard read speed.
 - MULTI_BOOT_MODE must be set to ENABLE to enable multi-boot mode.
 - MULTI_BOOT_SEL option selects the booting address for the multiple boot event when MULTI_BOOT_MODE is set to ENABLE.
 - STATIC (default): Use the value stored in the MultiBoot Offset register as the booting address. The MultiBoot Offset register value is set according to the bitstream pattern boot sequence in the Lattice Radiant Deployment Tool. Refer to the MultiBoot Offset section in the [Lattice Nexus 2 sysCONFIG User Guide \(FPGA-TN-02370\)](#) for more information about the MultiBoot Offset register.
 - DYNAMIC: Use the multi-boot address register programmed through the CONFIG_LMMI interface.
- MCCLK_FREQ, MULTI_BOOT_MODE, and MULTI_BOOT_SEL can be configured using the Global tab of the Device Constraint Editor in the Lattice Radiant software ([Figure 7.13](#)).

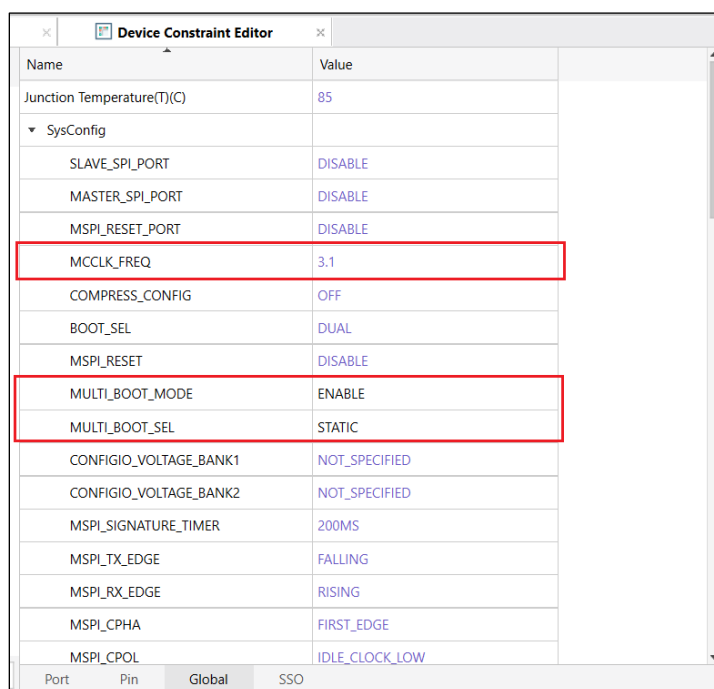


Figure 7.13. Configuring MCCLK_FREQ, MULTI_BOOT_MODE, and MULTI_BOOT_SEL in Device Constraint Editor

- Invoke the Lattice Radiant Deployment Tool by choosing **Tools > Deployment Tool** from the Lattice Radiant Programmer.
- In the Radiant Deployment Tool window, select **External Memory** as the **Function Type** and select **Advanced SPI Flash** as the **Output File Type** (Figure 7.14).
- Click **OK**.

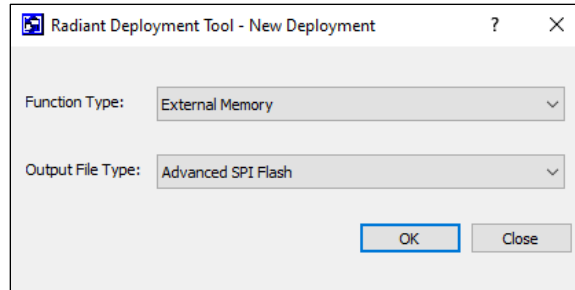


Figure 7.14. Creating New Deployment for Multi-Boot File

Step 1 of 4: Select Input File(s) window (Figure 7.15)

- Click on the ... button in the **File Name** fields to browse and select the primary bitstream file to be used to create the SPI storage device hex file.
- The **Device Family** and **Device** fields automatically populate based on the bitstream file selected.
- Click **Next**.

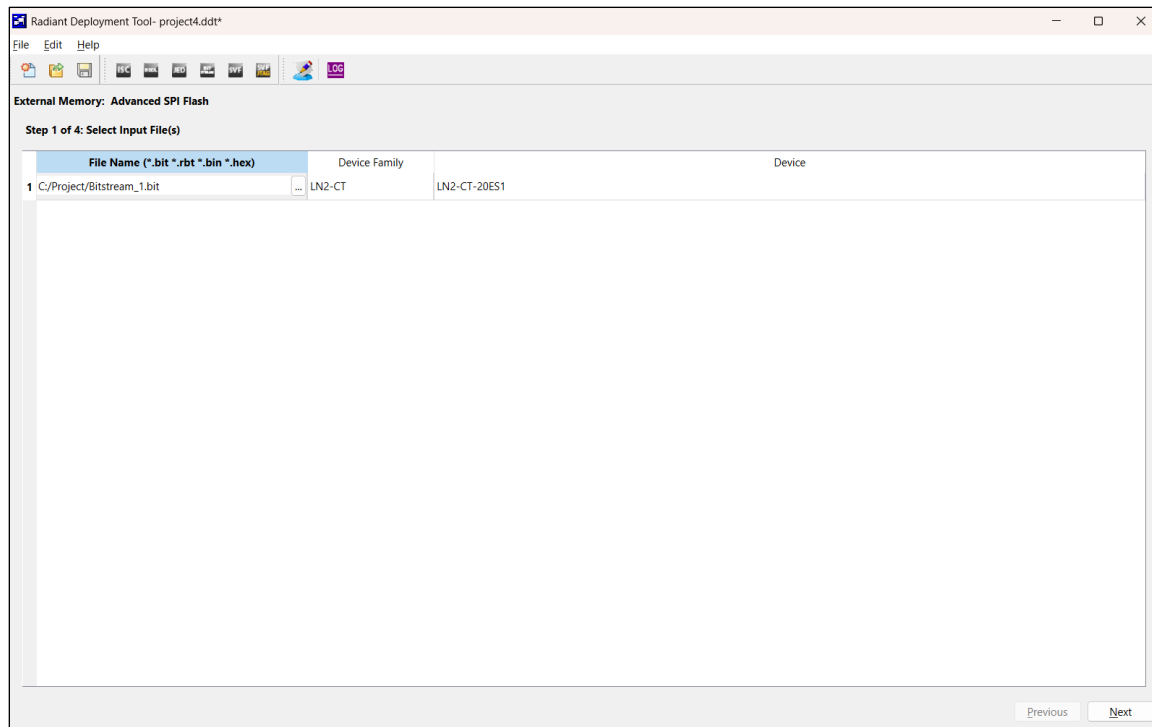


Figure 7.15. Select Input File(s) Window

Step 2 of 4: Advanced SPI Flash Options window

- Edit the following options in the **Options** tab (Figure 7.16):
 - Select the **Output Format** (Intel Hex, Motorola Hex, or Extended Tektronix Hex).
 - Select the **SPI Flash Size (Mb)** (4, 8, 16, 32, 64, 128, 256, 512, or 1024 Mb).
 - Select **SPI Flash Read Mode** (Standard Read, Fast Read, Dual I/O SPI Flash Read, Quad I/O SPI Flash Read, xSPI Profile 1 Read, or xSPI Profile 2 Read).
 - Select the **Clock Mode**.
 - When **SPI Flash Read Mode** is set to Standard Read, Fast Read, Dual I/O SPI Flash Read, or Quad I/O SPI Flash Read, only the *Single-ended Clock* option is available.
 - When **SPI Flash Read Mode** is set to xSPI Profile 1 Read or xSPI Profile 2 Read, the *Single-ended Clock* and *Differential Clock* options are available.
- Select the following options as required:
 - **Byte Wide Bit Mirror** – Flips each byte in Intel, Extended Tektronix, or Motorola hexadecimal data files. For example, 0xCD (b1100 1101) becomes 0xB3 (b1011 0011) when this is selected.
 - **Retain Bitstream Header** – By default, Radiant Deployment Tool replaces the bitstream header information (name, version number, and date of the file) with 0xFF values. Selecting this option retains the header information that is generated as the header.
 - **Optimize Memory Space** – When checked, it uses the bitstream file size instead of the worst-case bitstream size.
 - **SFDP Enabled** – Enables the checking of SFDP signature during controller SPI booting. This option must be enabled for flash devices that support SFDP.
 - **Use Read Mode Settings for Golden Pattern** – Appears when **SPI Flash Read Mode** is set to Dual I/O SPI Flash Read or Quad I/O SPI Flash Read. When checked, the JUMP command in the backup jump table operates using the SPI flash read mode specified by the **SPI Flash Read Mode** setting instead of the default serial (x1) read mode. For example, if **SPI Flash Read Mode** is set to *Quad I/O SPI Flash Read*, the JUMP command loads the golden pattern using the quad I/O read mode. This allows the golden image to be loaded using the faster dual I/O or quad I/O read mode.
 - **Encryption** – Enables encryption of an unencrypted bitstream. After enabling, click on the **Edit Key** button and select the encryption key to be used.
 - **ECDSA Authentication** – Enables bitstream ECDSA authentication. After enabling, select **Authentication Mode** from the drop-down list, then load the **Public Key** and the **Private Key**.
 - **RSA Authentication** – Enables bitstream RSA Authentication. After enabling, select **Authentication Mode** from the drop-down list, then load the **Public Key** and the **Private Key**.

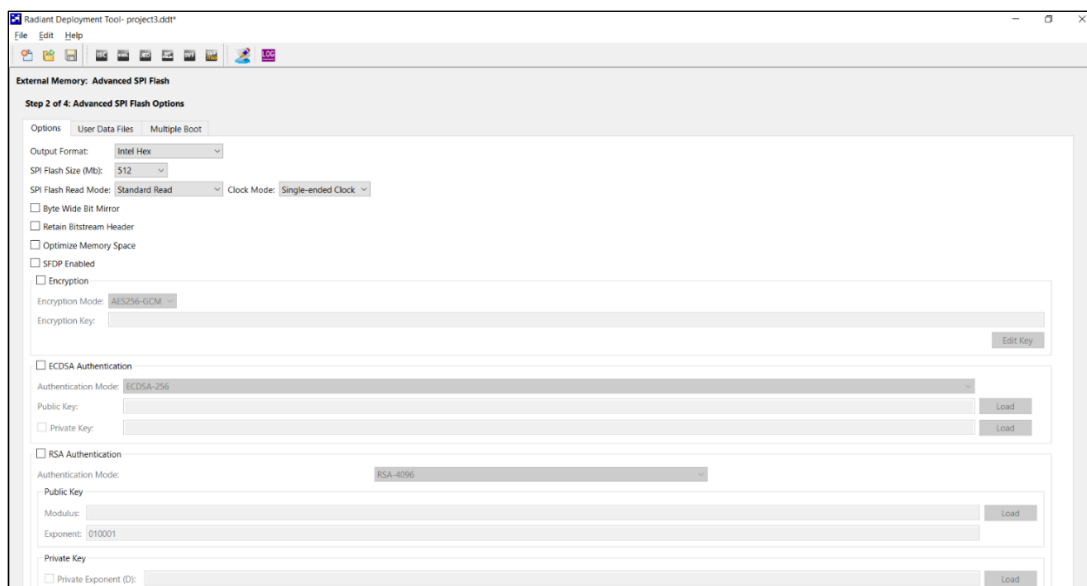


Figure 7.16. Advanced SPI Flash Options Window – Options Tab

- Edit the following options in the **Multiple Boot** tab (Figure 7.17):
 - Click on the **Multiple Boot** check box to enable multi-boot mode.
 - Click on the **Golden Pattern ...** button and select the bitstream file.
 - The starting address of the golden pattern is automatically populated.
 - Change the **Starting Address** of the patterns by clicking on the drop-down menu.
 - **Protect Golden Sector** – By default, the golden sector, where the golden pattern is stored, is located immediately after the primary sector to save SPI flash space. When this option is selected, the golden pattern location is moved to the first sector in the upper half of the SPI flash. The new location is reflected in the **Golden Pattern Starting Address** field. This protects the golden pattern from accidental erase/reprogram by protecting the upper half of the SPI flash when it is programmed.
 - In the **Multi_Boot_Sel Options** field, select either **Static** or **Dynamic** from the drop-down list.
 - In the **Number of Alternate Patterns** field, select the number of alternate patterns to include through the drop-down menu, selecting between one and four alternate patterns.
 - In the **Alternate Pattern 1** field, click on the ... button to select the first alternate pattern.
 - The **Starting Address of Alternate Pattern 1** is automatically populated.
 - You can change the **Starting Address of Alternate Pattern 1** by clicking on the drop-down menu.
 - The **Next Alternate Pattern to Configure** field is automatically populated. This is the pattern that is loaded during the next PROGRAMN/REFRESH event. You can change the pattern by clicking on the drop-down menu.
 - Configure the remaining alternate patterns as necessary.
- (Optional) If programming the flash memory with user data is required, click the **User Data Files** tab (Figure 7.18).
 - Check **User Data Files** to enable the insertion of the user data file.
 - Select the **Number of User Data Files** from the drop-down list.
 - Click the ... (browse) button associated with a user data file field, navigate to, and then select the user data file (*.bit, *.rbt, *.bin, *.mcs, *.exo, *.xtek) to be programmed into the SPI storage device.
 - Select the starting address to store the user data file from the **Starting Address** drop-down list.
- Click **Next**.

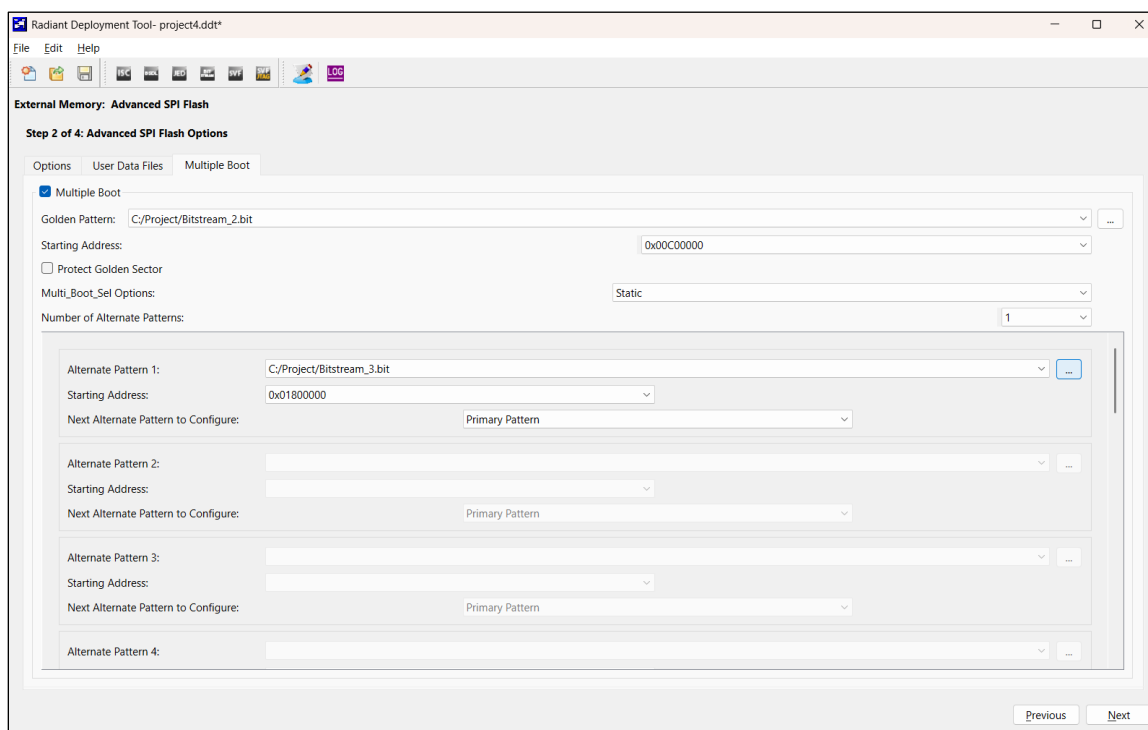


Figure 7.17. Advanced SPI Flash Options Window – Multiple Boot Tab

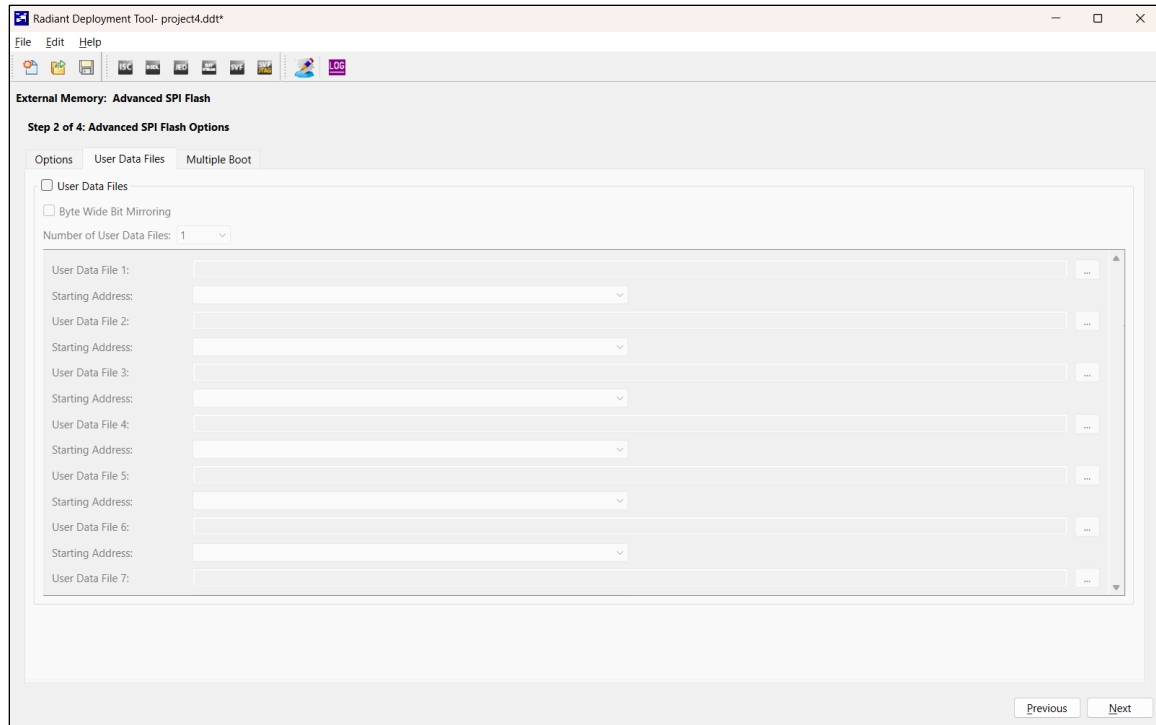


Figure 7.18. Advanced SPI Flash Options Window – User Data Files Tab

Step 3 of 4: Select Output File(s) window (Figure 7.19)

- Specify the name and location of the output SPI storage device hex file in the **Output File 1** field.
- Click **Next**.

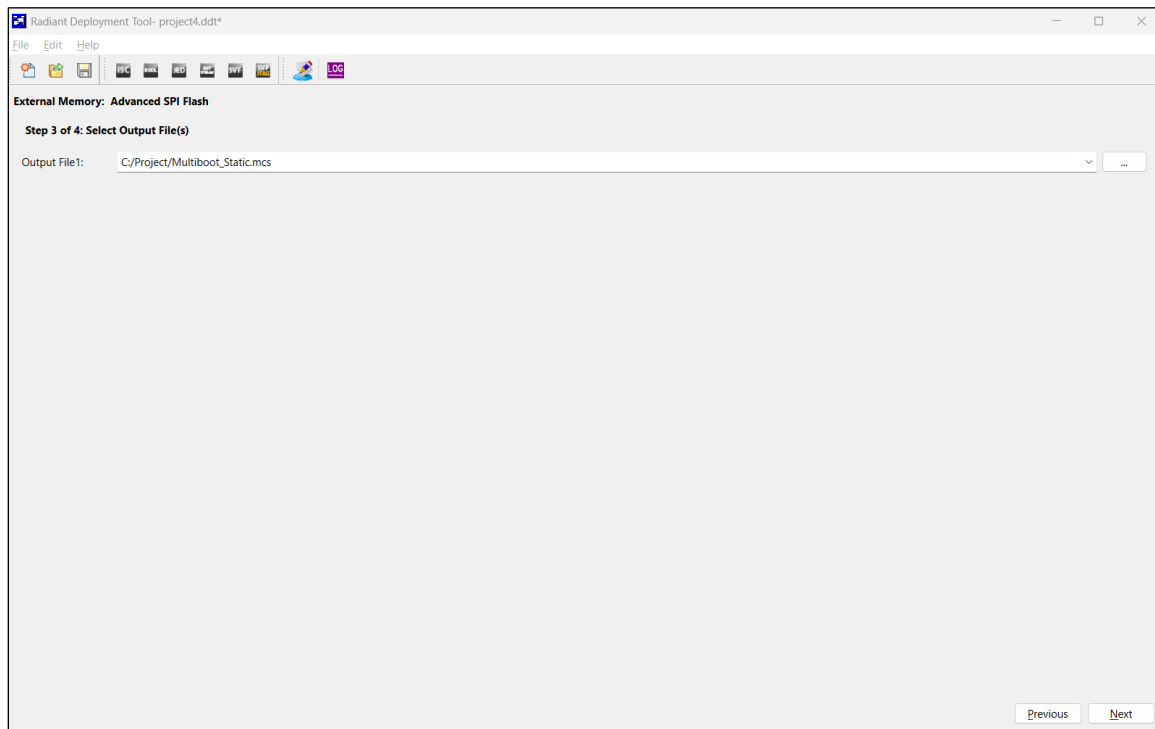


Figure 7.19. Select Output File(s) Window

Step 4 of 4: Generate Deployment window (Figure 7.20)

- Review the summary information.
- If everything is correct, click the **Generate** button.
- The **Generate Deployment** pane should indicate that the SPI storage device file is generated successfully.
- The generated .mcs file is located in the directory as specified in step 3 of 4.
- Save the deployment settings by selecting **File > Save** or **File > Save As....**
- To exit, select **File > Exit**.

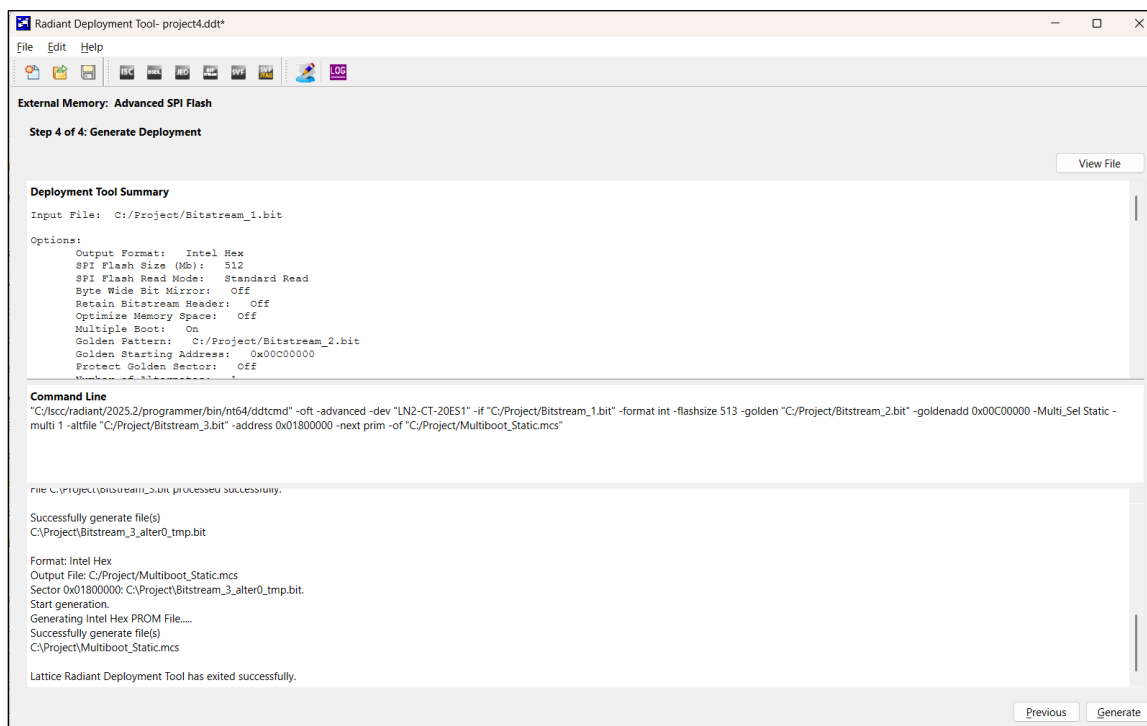


Figure 7.20. Generate Deployment Window

8. Programming Dual-Boot, Multi-Boot, or Ping-Pong Boot Pattern into an External SPI Storage Device – For Certus-N2 Only

The following procedure is for programming a dual-boot, multi-boot, or ping-pong boot pattern into the SPI storage device using the Radiant Programmer:

1. Connect power to the board and connect a download cable from the board to the PC.
2. Invoke Radiant Programmer 2025.2 or later using one of the following methods:
 - In the Radiant Software window, select **Tools > Programmer**.
 - In the Radiant Software window, select the **Programmer** icon (🔌) in the Radiant toolbar.
 - In the Windows Start menu, select **Start > Lattice Radiant Software 2025.2 > Radiant Programmer**.
3. The Radiant Programmer – Getting Started window opens (Figure 8.1).
 - Select **Create a new project from a scan**, **Create a new blank project**, or **Open an existing programmer project**.
 - If **Create a new project from a scan** is selected, click **Detect Cable** to scan the PC to determine what cable is connected or manually select the type of **Cable** and **Port**.
 - Click **OK**.

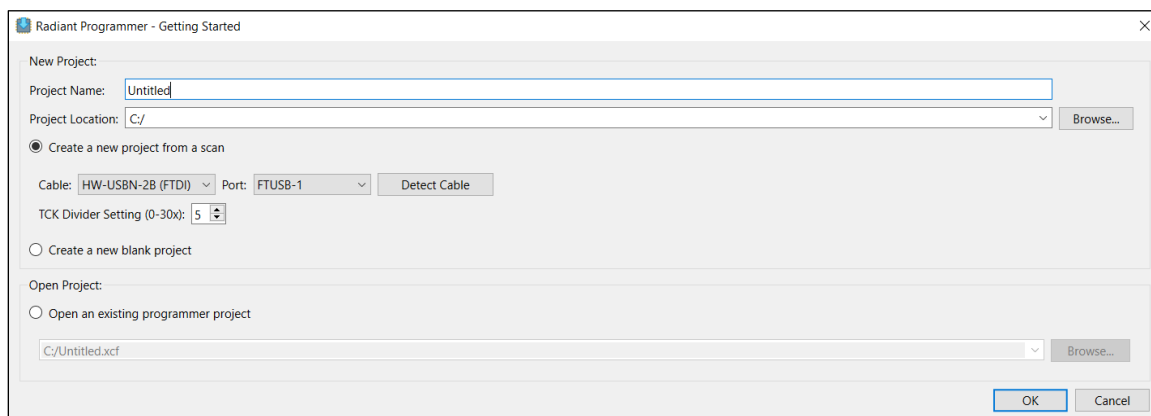


Figure 8.1. Radiant Programmer – Getting Started Window

4. The Radiant Programmer main window opens listing the device as found in a scan if applicable (Figure 8.2).

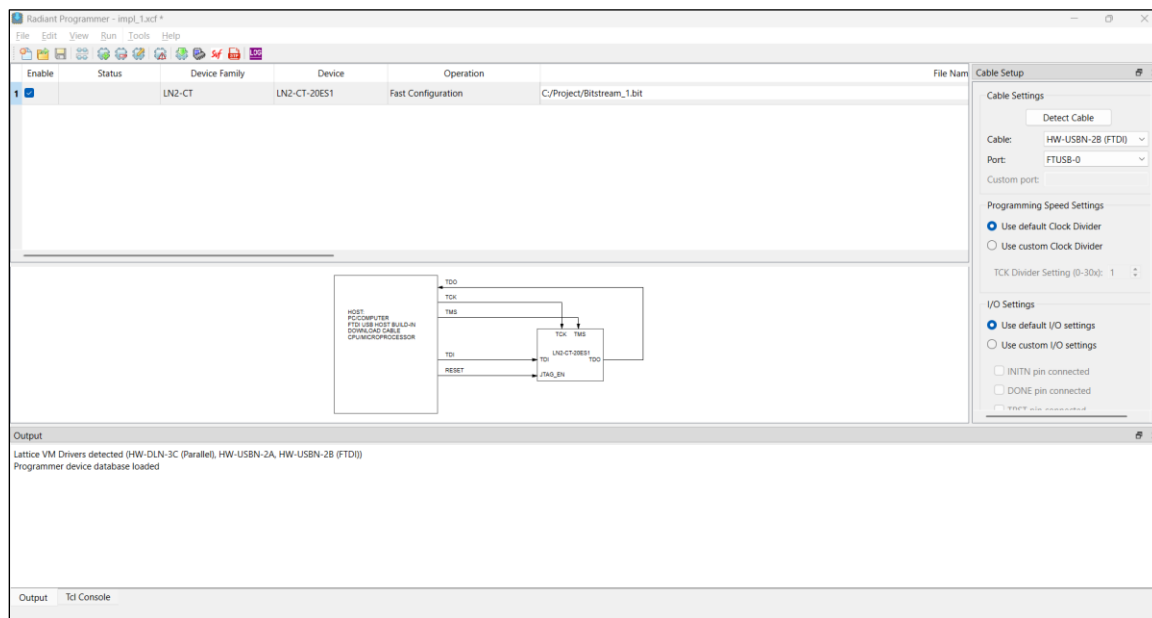


Figure 8.2. Radiant Programmer – Main Window (with Certus-N2 Device)

5. Double-click on the **Operation** field to open the **Device Properties** window.
6. Set the **Device Properties** fields to the following (Figure 8.3):
- For **Target Memory**, select **External SPI Flash Memory (SPI Flash)**.
 - For **Port Interface**, select **JTAG2SPI**.
 - For **Access Mode**, select **Direct Programming**.
 - For **Operation**, select **Erase, Program, Verify**.
 - For **Programming file**, browse to select the .mcs file to be programmed.
 - In the **SPI Flash Options** field, specify the **Family**, **Vendor**, **Device**, and **Package** of the flash device used on the board.
 - **Family** – SPI Serial Flash
 - **Vendor** – Winbond
 - **Device** – W35T51NW
 - **Package** – 24-ball BGA
 - For **Data file size (Bytes)**, click on the **Load from File** button.
 - Click **OK**.

Note: This example shows the configuration for programming of the LN2-CT-20 Evaluation Board.

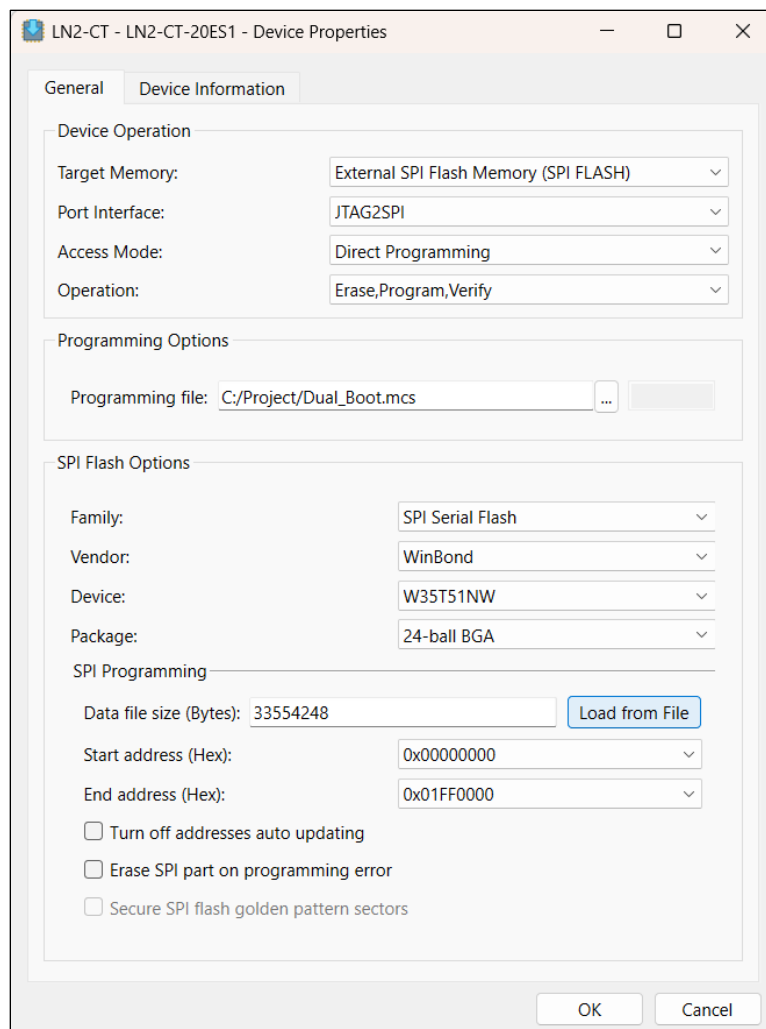


Figure 8.3. Radiant Programmer – Device Properties

7. Program the external SPI storage device by selecting **Run > Program Device** from the menu bar or clicking the **Program Device** icon () on the toolbar in the main Radiant Programmer window.

9. Programming Dual-Boot or Ping-Pong Boot Pattern into an Internal Memory Storage Device – For Mach-N2 Only

Note: The Mach-N2 device supports the multi-boot mode. The procedure for programming a multi-boot pattern into the internal memory storage device will be included in a future revision of this document.

The following procedure is for programming a dual-boot or ping-pong boot pattern into the non-volatile memory storage device using the Radiant Programmer:

1. Generate two bitstream files in the Lattice Radiant software.
 - Set the MCCLK_FREQ (MCLKP frequency) setting of the bitstream files to a value not exceeding the non-volatile memory storage device normal or standard read speed. MCCLK_FREQ can be configured using the Global tab of the Device Constraint Editor in the Lattice Radiant software.
 - Set BOOT_SEL = DUAL in the Device Constraint Editor.
 - Set PRIMARY_BOOT = IMAGE_0 in the Device Constraint Editor. The start address for booting the primary pattern corresponds to the start address of CFG0. This primary boot pattern start address is reflected in the *header.mcs file generated by the Radiant software upon successful compilation.
 - Set SECONDARY_BOOT = IMAGE_1 in the Device Constraint Editor. The start address for booting the golden pattern corresponds to the start address of CFG1. This golden boot pattern start address is reflected in the *header.mcs and *tail.mcs files generated by the Radiant software upon successful compilation.
2. Connect power to the board and connect a download cable from the board to the PC.
3. Invoke Radiant Programmer 2025.2 or later using one of the following methods:
 - In the Radiant Software window, select **Tools > Programmer**.
 - In the Radiant Software window, select the **Programmer** icon () in the Radiant toolbar.
 - In the Windows Start menu, select **Start > Lattice Radiant Software 2025.2 > Radiant Programmer**.
4. The Radiant Programmer – Getting Started window opens ([Figure 8.1](#)).
 - Select **Create a new project from a scan**, **Create a new blank project**, or **Open an existing programmer project**.
 - If **Create a new project from a scan** is selected, click **Detect Cable** to scan the PC to determine what cable is connected or manually select the type of **Cable** and **Port**.
 - Click **OK**.
5. The Radiant Programmer main window opens listing the device as found in a scan if applicable ([Figure 9.1](#)).

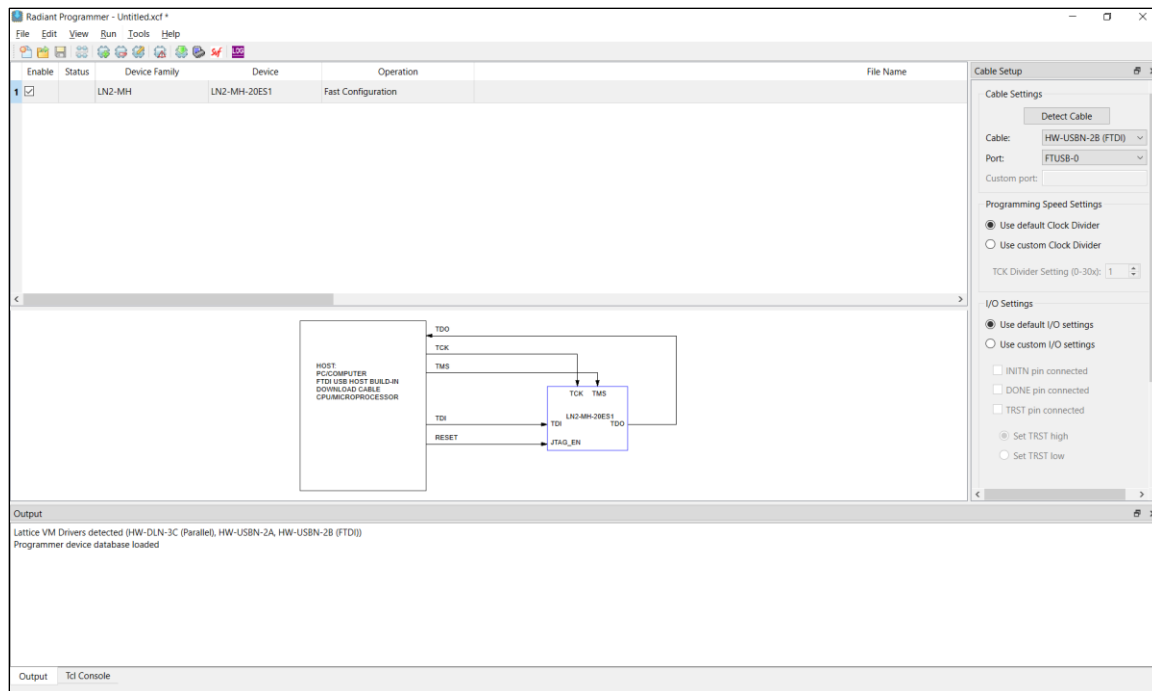


Figure 9.1. Radiant Programmer – Main Window (with Mach-N2 Device)

6. Double-click on the **Operation** field to open the **Device Properties** window.
7. Set the **Device Properties** fields to the following (Figure 9.2):
 - For **Target Memory**, select **FLASH Configuration Memory**.
 - For **Port Interface**, select **JTAG**.
 - For **Access Mode**, select **Direct FLASH Programming**.
 - For **Operation**, select **Erase, Program, Verify**.
 - Check the **Flash Header/Dual Image Programming Options** check box and insert the flash header programming file. The supported file format is *.header.mcs. This option is only required for the ping-pong boot mode and is not required for the dual-boot mode. The header MCS file is generated by the Radiant software upon successful compilation.
 - Check the **CFG0/UFM0 Programming Options** or **CFG1/UFM1 Programming Options** check box and insert the CFG0 or CFG1 programming file. The programming file format is .jed. The JED file is generated by the Radiant software upon successful compilation.
 - For single boot mode, you can insert the configuration pattern in CFG0.
 - For dual or ping-pong boot mode, you can insert the primary pattern in CFG0 and the secondary pattern in CFG1.
 - With the **CFG0/UFM0 Programming Options** or **CFG1/UFM1 Programming Options** check box checked, you can also check the corresponding **UFM0 Programming Options** or **UFM1 Programming Options** check box and insert the UFM0 or UFM1 programming file. The programming file format is .mcs. This option is optional if you do not plan to store user data in the UFM sector.
 - Check the **UserDataX Programming** check box, where X denotes 0 to 7, and insert the corresponding user data programming file if you plan to store user data in the user flash memory sector. The programming file format is .mcs. After inserting the programming file, you can check the start and end addresses from the **Start address (Hex)** and **End address (Hex)** drop-down lists, respectively. This option is optional if you do not plan to store user data in the UserData sector.
 - Check the **Jump Command Programming** check box and insert the jump command programming file, which contains jump table data. The programming file format is *.tail.mcs. This option is required if you are using dual or ping-pong boot mode. The jump command MCS file is generated by the Radiant software upon successful compilation.

- Click **OK** to close the **Device Properties** window.

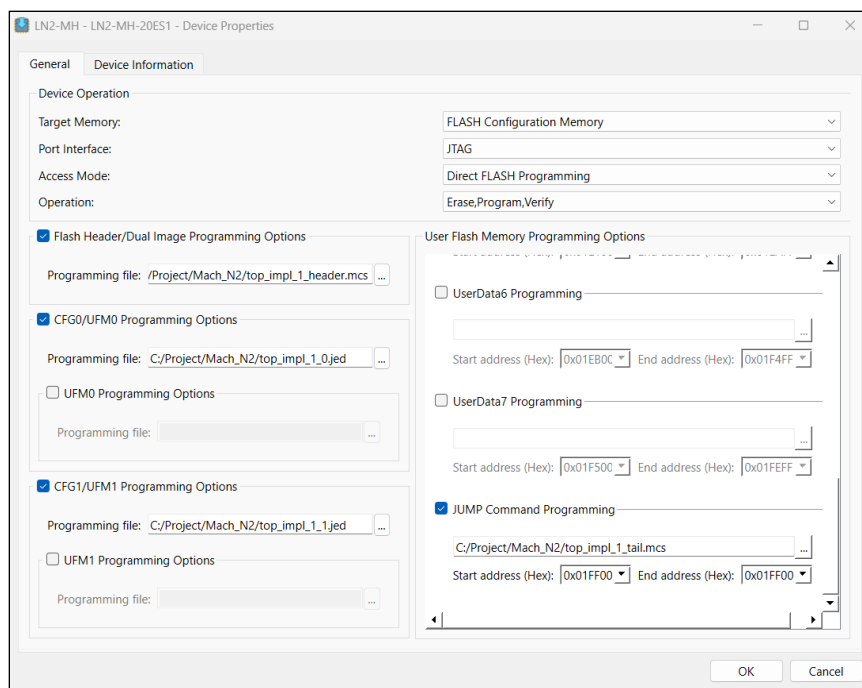


Figure 9.2. Radiant Programmer – Device Properties for FLASH Configuration Memory

Program the internal memory storage device by selecting **Run > Program Device** from the menu bar or clicking the **Program Device** icon () on the toolbar in the main Radiant Programmer window.

10. Pattern Update Guideline

Note: This guideline is applicable to the dual-boot and ping-pong boot modes.

The primary jump table contains a JUMP command that directs the device to the primary bitstream pattern and a program command that identifies the location of the golden bitstream pattern, as shown in the following table.

Table 10.1. Primary Jump Table Structure

Item	Value	Address Offset	Size (Bytes)
LSCC Signature	0x4C534343	0x00	4
Preamble	0xFFFFBDB3	0x04	4
Segment Length	0x00000020	0x08	4
Reset CRC command	0x02000000	0x0C	4
Program Secondary/Golden Boot Command and Address	0x80062000	0x10	4
	Address[47:0]	0x16	6
JUMP command and address	0x80060200	0x1A	4
	Address[47:0]	0x20	6
Check CRC command and expected CRC for Jump Table	0x03000000	0x24	4
	CRC[31:0]	0x28	4
Padding Bytes	0xFFFFFFFF	0x2C	4

The following is the recommended pattern update process:

1. Change the primary jump table JUMP command address to point to the golden pattern.
2. Write the new primary pattern to the non-volatile memory device.
3. Verify the new primary pattern in the non-volatile memory device.
4. If verification is successful:
 - Change the primary jump table JUMP command address to point to the primary pattern.
 Otherwise:
 - Retry the write process until the new primary pattern in the non-volatile memory device is verified.

This update process is recommended because if the primary pattern write is interrupted and the flash data becomes corrupted, the device can still boot from the golden pattern, as the JUMP command address points to the golden pattern. You can then recover the device as appropriate.

References

- [Lattice Nexus 2 Platform – Overview Data Sheet \(FPGA-DS-02122\)](#)
- [Lattice Nexus 2 Platform – Specifications Data Sheet \(FPGA-DS-02121\)](#)
- [Certus-N2 web page](#)
- [Mach-N2 web page](#)

A variety of technical notes for the Lattice Nexus 2 family are available.

- [Lattice Nexus 2 sysCONFIG User Guide \(FPGA-TN-02370\)](#)
- [High-Speed PCB Design Considerations \(FPGA-TN-02178\)](#)
- [Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide \(FPGA-UG-02039\)](#)
- [Lattice Nexus 2 Configuration Security User Guide \(FPGA-TN-02369\)](#)
- [Lattice Nexus 2 Embedded Memory User Guide \(FPGA-TN-02366\)](#)
- [Lattice Nexus 2 High-Speed I/O and External Memory Interface \(FPGA-TN-02372\)](#)
- [Lattice Nexus 2 Multi-Boot User Guide \(FPGA-TN-02385\)](#)
- [Lattice Nexus 2 Power User Guide \(FPGA-TN-02381\)](#)
- [Lattice Nexus 2 SED/SEC User Guide \(FPGA-TN-02380\)](#)
- [Lattice Nexus 2 sysCLOCK PLL Design and User Guide \(FPGA-TN-02364\)](#)
- [Lattice Nexus 2 sysDSP User Guide \(FPGA-TN-02362\)](#)
- [Lattice Nexus 2 sysI/O User Guide \(FPGA-TN-02365\)](#)
- [Sub-LVDS Signaling Using Lattice Devices \(FPGA-TN-02028\)](#)
- [Thermal Management \(FPGA-TN-02044\)](#)
- [Using TraceID \(FPGA-TN-02084\)](#)

For more information on Lattice Nexus 2-related IP and reference designs, refer to the following pages:

- [IP and Reference Designs for Nexus 2](#)

For further information on interface standards refer to the following websites:

- JEDEC Standards (LVTTTL, LVCMOS, SSTL) – www.jedec.org
- PCI – www.pcisig.com

Other references:

- [Lattice Radiant](#) FPGA design software
- [Lattice Insights](#) for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

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

Revision History

Revision 0.81, September 2026

Section	Change Summary
All	- Added descriptions to reference internal non-volatile memory or internal flash memory where appropriate. - Changed references from <i>SPI storage device</i> or <i>external SPI storage device</i> to <i>non-volatile memory storage device</i> where appropriate. - Changed references from <i>non-volatile SPI memory</i> or <i>external SPI flash</i> to <i>non-volatile memory</i> where appropriate. - Made minor editorial changes.
Abbreviations in This Document	Added <i>BGA</i> , <i>CFG</i> , <i>ECDSA</i> , <i>FPGA</i> , <i>FSM</i> , <i>GUI</i> , <i>ID</i> , <i>JTAG</i> , <i>LMMI</i> , <i>OSC</i> , <i>PC</i> , <i>RAM</i> , <i>RSA</i> , <i>SFDP</i> , and <i>UFM</i> .
Definition of Terms	- Updated definition of <i>Alternative Boot Pattern</i> to include reference to <i>alternate pattern</i>. - Updated definition of <i>Programming</i> to reference both <i>external SPI storage device</i> and <i>internal flash memory</i>.
Resources	- In Table 3.1. Bitstream Size versus Recommended SPI Storage Size (Single Bitstream Boot Mode): - Updated bitstream size for LN2-CT-20 (maximum EBR only). - Updated recommended SPI storage device size for LN2-CT-20 (maximum EBR only). - Added LN2-MH-20 device and note on internal flash memory size. - In Table 3.2. Bitstream Size versus Recommended SPI Storage Size (Dual-Boot Mode, Ping-Pong Boot Mode): - Updated bitstream sizes for LN2-CT-20 (maximum EBR only). - Updated recommended SPI storage device sizes for LN2-CT-20 (maximum EBR only). - Added LN2-MH-20 device and note on internal flash memory size. - In Table 3.3. Bitstream Size versus Recommended SPI Storage Size (Multi-Boot Mode): - Updated bitstream sizes for LN2-CT-20 (maximum EBR only). - Updated recommended SPI storage device size for LN2-CT-20 with the following: - Maximum EBR; 4 bitstream patterns - Added LN2-MH-20 device and note on internal flash memory size.
Dual-Boot Mode	- Updated description of dual-boot mode to include primary jump table. - In Figure 4.1. Dual-Boot Flow Diagram: - Updated figure to include primary jump table preamble and primary pattern preamble checks. - Added primary jump table. - Changed jump table to backup jump table. - Updated label to <i>Non-volatile Memory Storage Content</i>". - Added notes on primary jump table, backup jump table, primary jump table contents, primary pattern loading, golden pattern loading, and primary pattern fixed address. - In the Dual-Boot Flow section: - Updated introducing description to list of reasons a pattern might fail to load and made minor editorial changes to the list. - Updated description on golden pattern to include primary jump table. - Updated jump table note to become backup jump table note. Also, updated references to 24-bit, 32-bit, and 45-bit addressing. Additionally, updated references from <i>SPI flash</i> to <i>non-volatile memory</i> or <i>flash</i> where appropriate, as well as related description. - Added notes regarding 32-bit MSPI Address and 32-bit MSPI Commands items.
Ping-Pong Boot Mode	- In Figure 5.1. Ping-Pong Boot Flow Diagram: - Updated figure to include primary jump table preamble and primary pattern preamble checks. - Updated label to <i>Non-volatile Memory Storage Content</i>. - Added notes on primary jump table contents, primary pattern loading, and golden pattern loading.

Section	Change Summary
	- In the Ping-Pong Boot Flow section: - Made minor editorial changes to the list of reasons a pattern might fail to load. - Updated descriptions on golden pattern and condition for loading the backup jump table. - Streamlined description on when the device drives the INITN pin low. - In a note, updated references to 24-bit, 32-bit, and 45-bit addressing. Additionally, updated references from <i>SPI flash</i> to <i>non-volatile memory or flash</i> where appropriate, as well as related description. - Added notes regarding 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row.
Multi-Boot Mode	- In Figure 6.1. Multi-Boot Flow Diagram: - Updated label to <i>Non-volatile Memory Storage Content</i>. - In the Multi-Boot Flow section: - Added Figure 6.2. Static Multi-Boot Mode Flow and Figure 6.3. Dynamic Multi-Boot Mode Flow. - Made minor editorial changes to description. - Updated description to reference both <i>external SPI storage device</i> and <i>internal flash memory</i> where appropriate. - Added statement on condition when golden pattern configuration fails. - In a note, updated references to 24-bit, 32-bit, and 45-bit addressing. Additionally, updated references from <i>SPI flash</i> to <i>non-volatile memory or flash</i> where appropriate, as well as related description. - Added notes regarding 32-bit MSPI Address and 32-bit MSPI Commands items in the feature row. - Updated description of STATIC setting for the MULTI_BOOT_SEL option. - Added description for the Multiboot Offset option. - Added the Implementation of Static or Dynamic Multi-Boot Mode section.
Creating an SPI Storage Device Hex File - For Certus-N2 Only	Added new section.
Programming Dual-Boot, Multi-Boot, or Ping-Pong Boot Pattern into an External SPI Storage Device - For Certus-N2 Only	Added new section.
Programming Dual-Boot or Ping-Pong Boot Pattern into an Internal Memory Storage Device – For Mach-N2 Only	Added new section.
Pattern Update Guideline	Added new section.
References	- Added Mach-N2 web page. - Added link to FAQ page for Lattice Nexus 2 Configuration Security User Guide. - Added IP and Reference Designs for Nexus 2. - Reorganized items.

Revision 0.80, December 2024

Section	Change Summary
All	Initial preliminary release.



www.latticesemi.com