Common Programming and Configuration FAQs with Supplementary Concepts for CrossLink-NX, Certus-NX, and CertusPro-NX

Application Note

FPGA-AN-02048-1.2

July 2023

Disclaimers

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AES	Advanced Encryption Standard
FPGA	Field-Programmable Gate Array
GOE	Global Output Enable
GSRN	Global Set/Reset
GWE	Global Write Enable
I^2C	Inter-Integrated Circuit
JTAG	Joint Test Action Group
LUT	Look-Up Table
MCU	Microcontroller Unit
MIB	Memory Interface Block
MSB	Most Significant Bit
MSPI	Master Serial Peripheral Interface
OTP	One-Time Programmable
RAM	Random Access Memory
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SSPI	Slave Serial Peripheral Interface
SVF	Serial Vector Format
TCK	Test Clock

1. Introduction

The purpose of this document is to provide an additional reference for frequently asked questions and recurring inquiries relating mainly to configuration and programming issues regarding the Lattice Nexus™ devices.

2. PROGRAMN/INITN/DONE as GPIO

This section covers the following:

- Using the PROGRAMN/INITN/DONE as GPIO
- Difference between *.bit* file and *.fea* file generated by the Lattice Radiant™ software
- Read and Program the feature row

2.1. Terminologies

2.1.1. DONE

The DONE pin is a bi-directional open drain with a weak pull-up that signals the FPGA is in user mode. DONE is first able to indicate entry into user mode only after an internal DONE bit is asserted. The internal DONE bit defines the beginning of the FPGA Wake-Up state. The FPGA can be held from entering User mode indefinitely by having an external agent keep the DONE pin asserted low. A common reason for keeping DONE driven low is to allow multiple FPGAs to be completely configured. As each FPGA reaches the DONE state, it is ready to begin operation. The last FPGA to configure can cause all FPGAs to start in unison. The DONE pin drives low in tandem with the INITN pin when the FPGA enters Initialization mode. As described earlier, this condition happens when power is applied, PROGRAMN is asserted, or an IEEE 1532 Refresh command is received through an active configuration port. Sampling the DONE pin is a way for an external device to tell if the FPGA has finished configuration.

2.1.2. INITN

The INITN pin is a bidirectional open-drain control pin. It has the following functions:

- After power is applied, after a PROGRAMN assertion, or a REFRESH command it goes low to indicate the SRAM configuration memory is being erased. The low time assertion is specified with the t_{INIT_LOW} parameter.
- After the t_{INITL} time period has elapsed the INITN pin is deasserted (which is active high) to indicate the Nexus device is ready for its configuration bits. The Nexus device begins loading configuration data from an external SPI Flash.
- INITN can be asserted low by an external agent before the t_{INITL} time period has elapsed in order to prevent the FPGA from reading configuration bits. This is useful when there are multiple programmable devices chained together. The programmable device with the longest t_{INITL} time can hold all other devices in the chain from starting to get data until it is ready itself.

The last function provided by INITN is to signal an error during the time configuration data is being read. Once t_{INITL} has elapsed and the INITN pin has gone high, any subsequent INITN assertion signals the Nexus device has detected an error during configuration.

The following conditions cause INITN to become active, indicating the Initialization state is active:

- Power has just been applied
- PROGRAMN falling edge occurred
- The IEEE 1532 REFRESH command is sent using a slave configuration port (JTAG or SSPI). If the INITN pin is asserted due to an error condition, the error can be cleared by correcting the configuration bitstream and forcing the FPGA into the Initialization state.

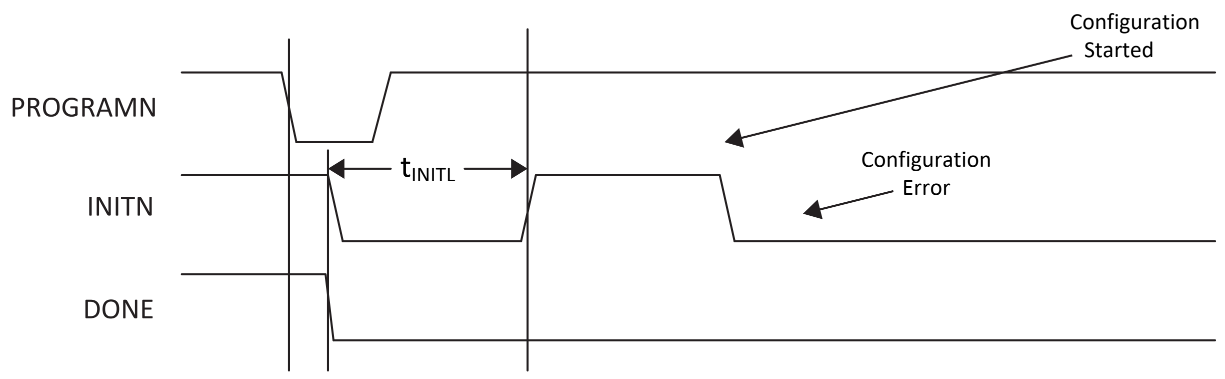


Figure 2.1. Configuration Error Notification

If an error is detected when reading the bitstream, INITN goes low, the internal DONE bit is not set, the DONE pin stays low, and the device does not wake up. The device configuration fails when the following happens:

- The bitstream CRC error is detected.
- The invalid command error is detected.
- A time out error is encountered when loading from the external Flash. This can occur when the device is in MSPI configuration mode and the SPI Flash device is not programmed.
- The program done command is not received when the end of on-chip SRAM configuration or external Flash memory is reached.

2.1.3. PROGRAMN

PROGRAMN is an input used to configure the FPGA. The PROGRAMN pin is low level sensitive and has an internal weak pull-up. When PROGRAMN is asserted low, the FPGA exits user mode and starts a device configuration sequence at the Initialization phase, as described earlier. Holding the PROGRAMN pin low prevents the Nexus device from leaving the Initialization phase. The PROGRAMN has a minimum pulse width assertion period ($t_{PROGRAMN}$) in order for it to be recognized by the FPGA. User can find this minimum time in the sysCONFIG Port Timing Specifications of CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), and CertusPro-NX Family Data Sheet (FPGA-DS-02086).

Be aware of the following special cases when the PROGRAMN pin is active:

- If the device is currently being programmed through JTAG, PROGRAMN is ignored until the JTAG mode programming sequence is complete.
- Toggling the PROGRAMN pin during device configuration interrupts the process and restart the configuration cycle.
- PROGRAMN must not be asserted low until after all power rails have reached stable operation. This can be achieved by either placing an external pull-up resistor and tying it to the VCCIO0 voltage or permitting the FPGA's internal pull-up resistor to pull the input high.
- PROGRAMN must not make a falling edge transition during the time the FPGA is in the Initialization state to avoid interrupting, restarting or failing configuration. PROGRAMN must be asserted for a minimum low period of $t_{PROGRAMN}$ in order for it to be recognized by the FPGA. Failure to meet this requirement can cause the device to become non-operational, requiring power to be cycled.
- For slave port configuration or programming, if the PROGRAMN pin is persisted in user function mode, it has to be driven high before the ISC_DISABLE command is issued.

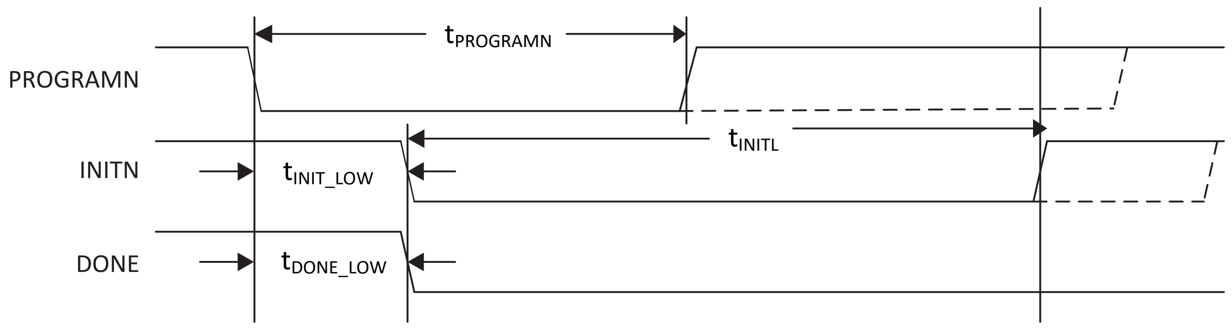


Figure 2.2. Configuration from PROGRAMN Timing

2.1.4. sysCONFIG Pins

The Nexus device provides a set of sysCONFIG I/O pins that the user use to program and configure the FPGA. The sysCONFIG pins are grouped together to create ports (such as JTAG, SSPI, MSPI) that are used to interact with the FPGA for programming, configuration, and access of resources inside the FPGA. The sysCONFIG pins in a configuration port group may be active, and used for programming the FPGA, or they can be reconfigured to act as general purpose I/O. Recovering the configuration port pins for use as general purpose I/O requires user to adhere to the following guidelines:

- User must DISABLE the unused port by using the Lattice Radiant Device Constraint Editor under the Global tab.
- User must prevent external logic from interfering with the device programming.

Table 2.1 lists the shared sysCONFIG pins of the device, and the default state of these pins in user mode. After programming the Nexus device, the default state of the SSPI sysCONFIG pins become general purpose I/O. This means user loses the ability to program the Nexus device using SSPI or Slave I^{2}C/I3C when using the default sysCONFIG port settings. To retain the SSPI or Slave I^{2}C/I3C sysCONFIG pins in user mode, be sure to ENABLE them using the Lattice Radiant Device Constraint editor.

Unless specified otherwise, the sysCONFIG pins are powered by the VCCIO0 and VCCIO1 voltage. It is crucial for this to be taken into consideration when provisioning other logic attached to Bank 0 and Bank 1. The function of each sysCONFIG pin is described in detail.

Table 2.1. Default State of PROGRAMN/INITN/DONE Pins

sysCONFIG Pins		Pull During Configuration	Configuration Modes			
Name	Type		JTAG	MSPI	SSPI	I^{2}C/I3C
PROGRAMN	Shared	UP	1'b0	1'b1	1'b0	1'b0
INITN	Shared	UP	INITN			
DONE	Shared	UP	DONE			

Notes:

1. SCSN should have 4.7 kΩ pull-up resistor on-board for SSPI.
2. MCSN should have 4.7 kΩ pull-up on-board resistor for MSPI.
3. Suggest 1 kΩ pull down resistor on-board for MCLK.
4. Inter weak pull-up or pull down is determined by the CR1 bit 22 (CPOL) setting. UP if CPOL = 1; DOWN if CPOL = 0.

2.1.5. GPIO

If the configuration pins (PROGRAMN/INITN/DONE) are disabled by programming the corresponding feature row bits, the configuration functionality are also disabled and user can use them as GPIOs. These bits are OTPs.

2.1.6. OTP

One-Time Programmable permits data to be written to memory only once.

2.1.7. Global Settings

The global settings are used to add some constraints on the design such as setting voltage on banks, enabling/disabling of the SSPI port, or enabling/disabling of the PROGRAMN port. Some of these settings (specifically the non-feature row-based settings) take effect together with the programming of the bitstream (*.bit*) on the FPGA. However, there are global settings which are feature row-based and this requires an additional step of programming the feature row. The Lattice Radiant design flow generates a *.fea* file format which specifies the contents of the feature row based on the feature row-based global settings. The additional step that is needed to be performed is to use this *.fea* file to perform a feature row programming.

2.2. Frequently Asked Questions

Q: How can I use the DONE, INITN, and PROGRAMN pins of Nexus devices as GPIO?

A: User needs to program the feature row with the *.fea* file generated during the design flow to be able to use them as GPIO.

Note: If user changes the Global settings, user needs to rerun the design flow to re-generate a new *.fea* file that reflects the changes on the Global settings with DONE, INITN, and PROGRAMN pins set to DISABLED).

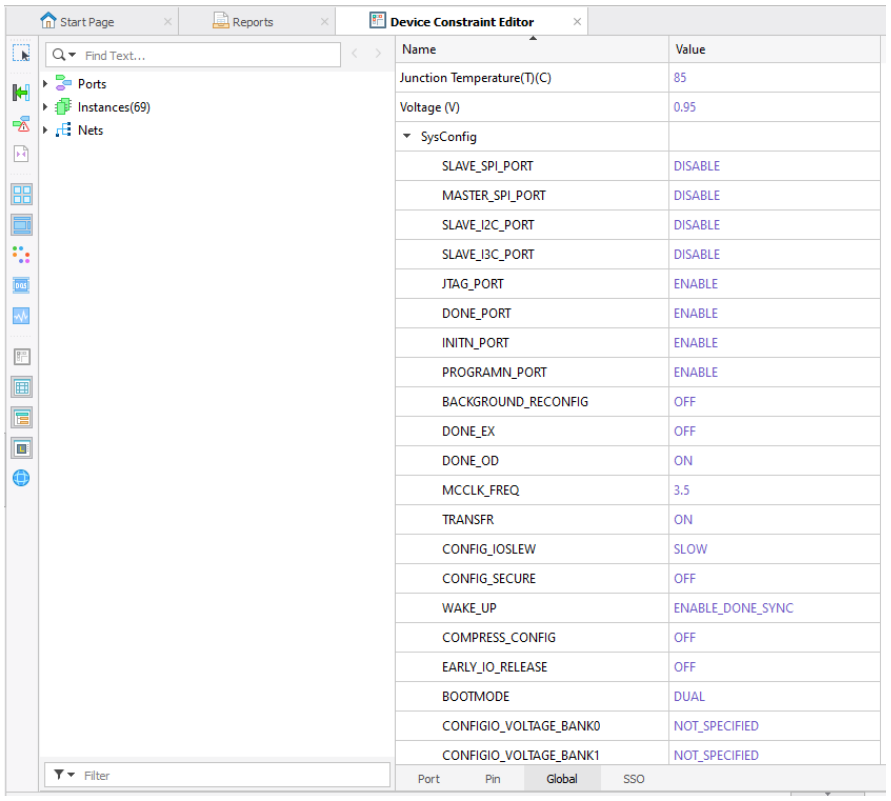
WARNING: These feature row bits are OTPs (One-time programmable). Disabling all these config pins could limit what user can do to the device as user is not able to re-configure using different interfaces SSPI or I^{2}C (instead of JTAG) since these pins are required during enabling of these interfaces.

Q: How does the user program the feature row for Nexus devices?

A: There are two ways the user can program the feature row: using *.fea* file and *Update feature row* operation.

To program the feature row using the *.fea* file generated during the design flow:

1. Open the Lattice Radiant software and edit the **Global Settings** under **Device Constraint Editor** as shown in Figure 2.3.



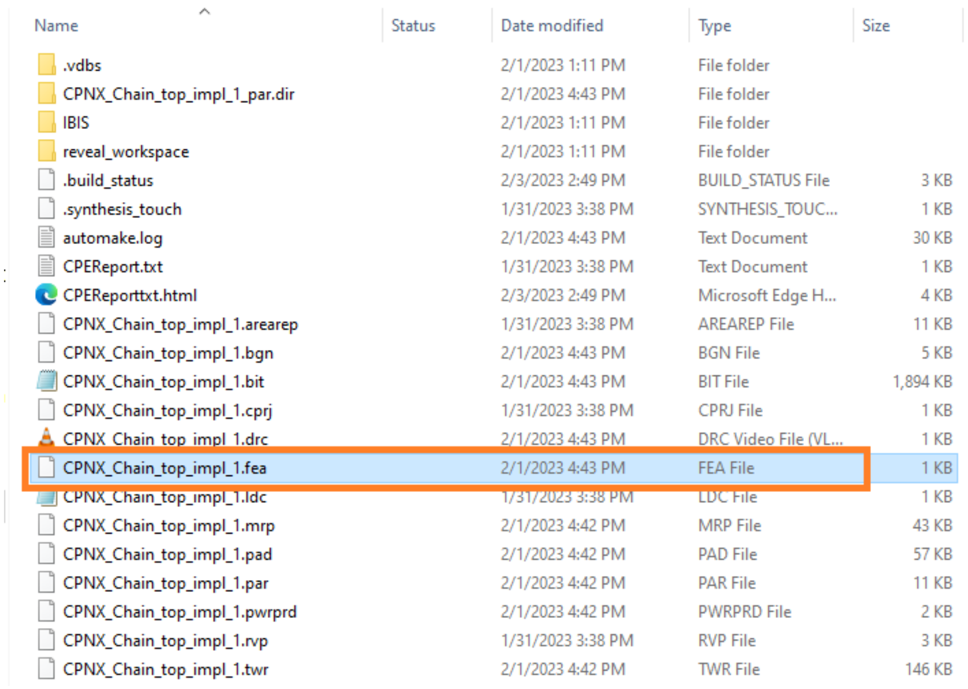
Name	Value
Junction Temperature(T)(C)	85
Voltage (V)	0.95
SysConfig	
SLAVE_SPI_PORT	DISABLE
MASTER_SPI_PORT	DISABLE
SLAVE_I2C_PORT	DISABLE
SLAVE_I3C_PORT	DISABLE
JTAG_PORT	ENABLE
DONE_PORT	ENABLE
INITN_PORT	ENABLE
PROGRAMN_PORT	ENABLE
BACKGROUND_RECONFIG	OFF
DONE_EX	OFF
DONE_OD	ON
MCCLK_FREQ	3.5
TRANSFR	ON
CONFIG_IOSLEW	SLOW
CONFIG_SECURE	OFF
WAKE_UP	ENABLE_DONE_SYNC
COMPRESS_CONFIG	OFF
EARLY_IO_RELEASE	OFF
BOOTMODE	DUAL
CONFIGIO_VOLTAGE_BANK0	NOT_SPECIFIED
CONFIGIO_VOLTAGE_BANK1	NOT_SPECIFIED

Figure 2.3. Lattice Radiant Software Global Settings

2. After editing the preferred settings, click **Export Files** to generate the *.fea* file.

   WARNING: DONE_PORT/INITN_PORT/PROGRAMN_PORT are all feature-based, which means these are OTP (One-time programmable).

   Figure 2.4 shows the generated file during the design flow, which is used to program the feature-row based settings.

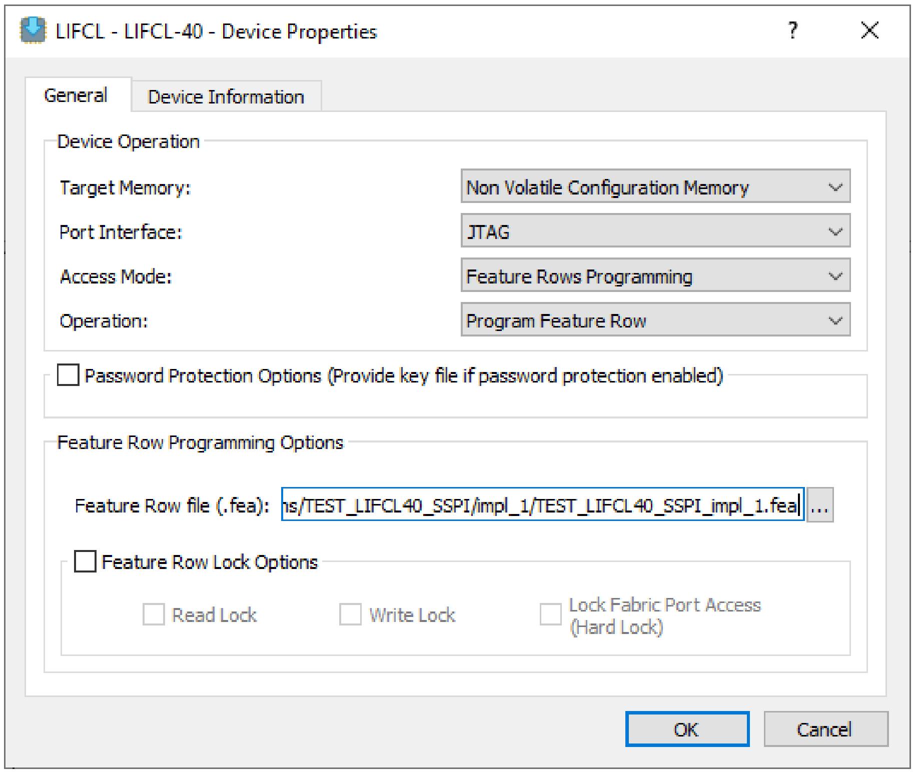


Name	Status	Date modified	Type	Size
.vdbs		2/1/2023 1:11 PM	File folder	
CPNX_Chain_top_impl_1_par.dir		2/1/2023 4:43 PM	File folder	
IBIS		2/1/2023 1:11 PM	File folder	
reveal_workspace		2/1/2023 1:11 PM	File folder	
.build_status		2/3/2023 2:49 PM	BUILD_STATUS File	3 KB
.synthesis_touch		1/31/2023 3:38 PM	SYNTHESIS_TOUC...	1 KB
automake.log		2/1/2023 4:43 PM	Text Document	30 KB
CPEReport.txt		1/31/2023 3:38 PM	Text Document	1 KB
CPEReporttxt.html		2/3/2023 2:49 PM	Microsoft Edge H...	4 KB
CPNX_Chain_top_impl_1.arearep		1/31/2023 3:38 PM	AREAREP File	11 KB
CPNX_Chain_top_impl_1.bgn		2/1/2023 4:43 PM	BGN File	5 KB
CPNX_Chain_top_impl_1.bit		2/1/2023 4:43 PM	BIT File	1,894 KB
CPNX_Chain_top_impl_1.cprj		1/31/2023 3:38 PM	CPRJ File	1 KB
CPNX_Chain_top_impl_1.drc		2/1/2023 4:43 PM	DRC Video File (VL...	1 KB
CPNX_Chain_top_impl_1.fea		2/1/2023 4:43 PM	FEA File	1 KB
CPNX_Chain_top_impl_1.ldc		1/31/2023 3:38 PM	LDC File	1 KB
CPNX_Chain_top_impl_1.mrp		2/1/2023 4:42 PM	MRP File	43 KB
CPNX_Chain_top_impl_1.pad		2/1/2023 4:42 PM	PAD File	57 KB
CPNX_Chain_top_impl_1.par		2/1/2023 4:42 PM	PAR File	11 KB
CPNX_Chain_top_impl_1.pwrprd		2/1/2023 4:42 PM	PWRPRD File	2 KB
CPNX_Chain_top_impl_1.rvp		1/31/2023 3:38 PM	RVP File	3 KB
CPNX_Chain_top_impl_1.twr		2/1/2023 4:42 PM	TWR File	146 KB

Figure 2.4. Generated .fea File

3. Using the Lattice Radiant programmer, user can now program the feature row as shown in Figure 2.5. The updated *.fea* file should now reflect within the feature row once programming operation is successful. User can double-check if the feature row has been updated appropriately using the *Read Feature Row* operation. Click **OK**.

   WARNING: These feature row bits are OTPs (One-time programmable). Disabling all these config pins could limit what user can do to the device as are not able to re-configure using different interfaces SSPI or I^{2}C (instead of JTAG) since these pins are required during enabling of these interfaces.


LIFCL - LIFCL-40 - Device Properties

General | Device Information

Device Operation

Target Memory: Non Volatile Configuration Memory

Port Interface: JTAG

Access Mode: Feature Rows Programming

Operation: Program Feature Row

☐ Password Protection Options (Provide key file if password protection enabled)

Feature Row Programming Options

Feature Row file (.fea): ns/TEST_LIFCL40_SSPI/impl_1/TEST_LIFCL40_SSPI_impl_1.fea

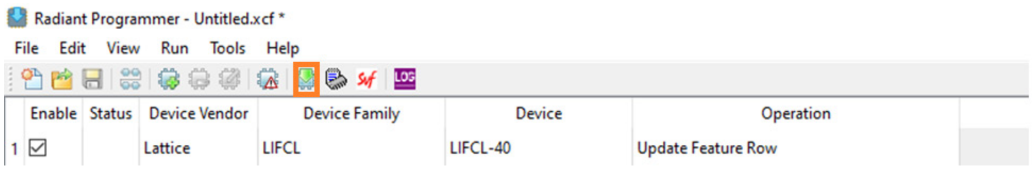
☐ Feature Row Lock Options

☐ Read Lock ☐ Write Lock ☐ Lock Fabric Port Access (Hard Lock)

OK Cancel

Figure 2.5. Device Properties

4. Click the **Program Device** button to program the feature row.


Radiant Programmer - Untitled.xcf *

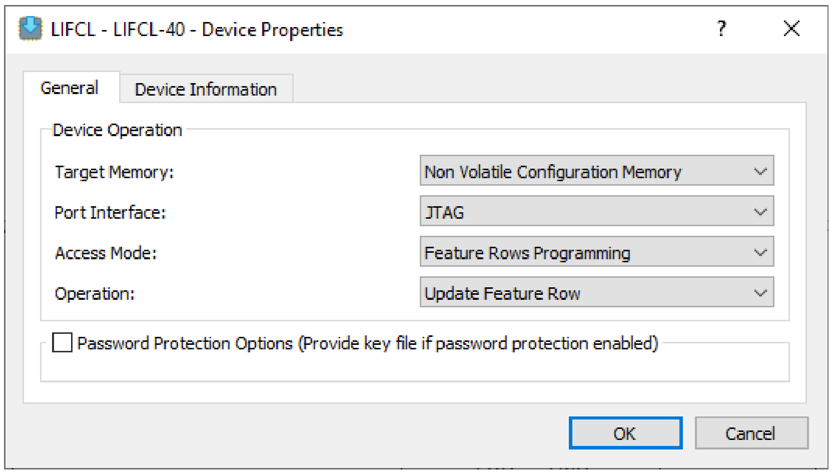
File Edit View Run Tools Help

	Enable	Status	Device Vendor	Device Family	Device	Operation
1	☑		Lattice	LIFCL	LIFCL-40	Update Feature Row

Figure 2.6. Program Device

To program the feature row using the *Update feature row* operation:

1. Open the Radiant Programmer and set the appropriate settings as shown in Figure 2.7. Click **OK**.


LIFCL - LIFCL-40 - Device Properties

General | Device Information

Device Operation

Target Memory: Non Volatile Configuration Memory

Port Interface: JTAG

Access Mode: Feature Rows Programming

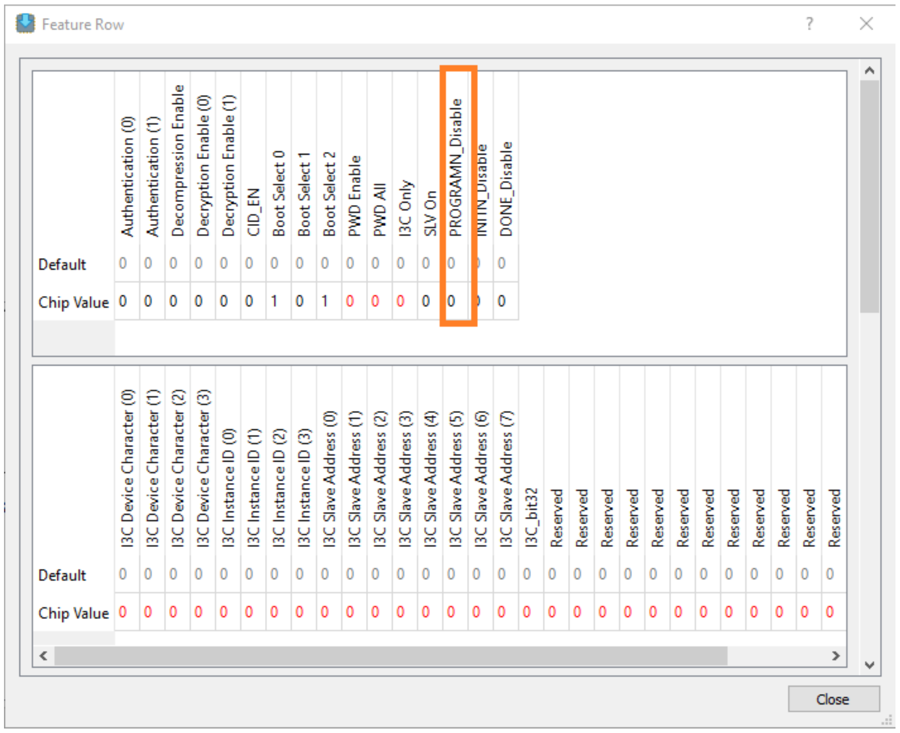
Operation: Update Feature Row

☐ Password Protection Options (Provide key file if password protection enabled)

OK Cancel

Figure 2.7. Device Properties

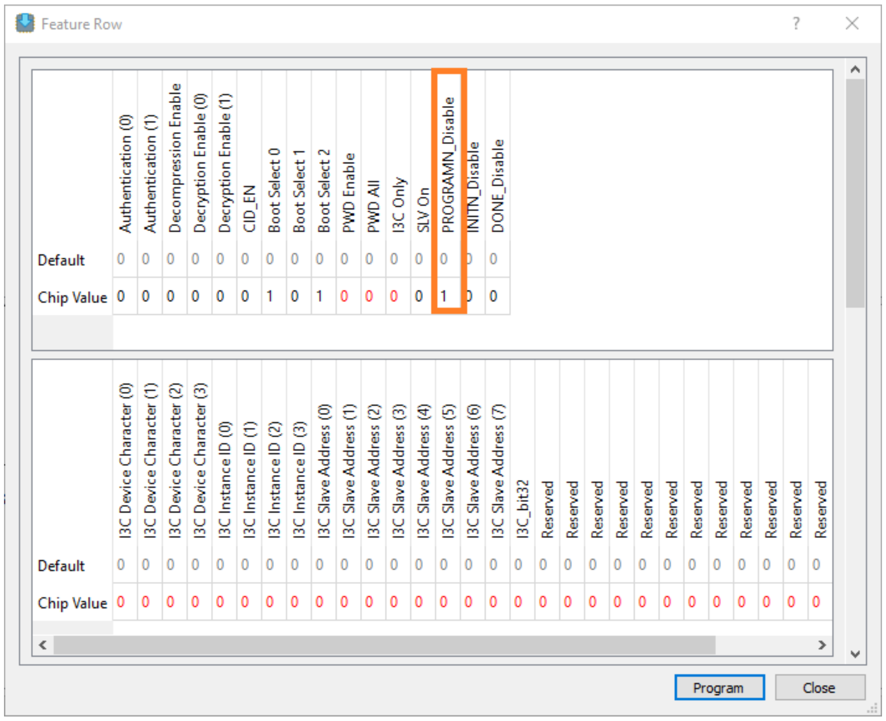
2. Click the **Program Device** button to update the feature row.
3. Select the feature row bit to toggle the value as shown in Figure 2.8. User can double-check if the feature row has been updated using the *Read Feature Row* operation.


Feature Row

	Authentication (0)	Authentication (1)	Decompression Enable	Decryption Enable (0)	Decryption Enable (1)	CID_EN	Boot Select 0	Boot Select 1	Boot Select 2	PWD Enable	PWD All	I3C Only	SLV On	PROGRAMN_Disable	INITN_Disable	DONE_Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0

	I3C Device Character (0)	I3C Device Character (1)	I3C Device Character (2)	I3C Device Character (3)	I3C Instance ID (0)	I3C Instance ID (1)	I3C Instance ID (2)	I3C Instance ID (3)	I3C Slave Address (0)	I3C Slave Address (1)	I3C Slave Address (2)	I3C Slave Address (3)	I3C Slave Address (4)	I3C Slave Address (5)	I3C Slave Address (6)	I3C Slave Address (7)	I3C_bit32	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2.8. Current Bit Value


Feature Row

	Authentication (0)	Authentication (1)	Decompression Enable	Decryption Enable (0)	Decryption Enable (1)	CID_EN	Boot Select 0	Boot Select 1	Boot Select 2	PWD Enable	PWD All	I3C Only	SLV On	PROGRAMN_Disable	INITN_Disable	DONE_Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	1	0	1	0	0	0	0	1	0	0

	I3C Device Character (0)	I3C Device Character (1)	I3C Device Character (2)	I3C Device Character (3)	I3C Instance ID (0)	I3C Instance ID (1)	I3C Instance ID (2)	I3C Instance ID (3)	I3C Slave Address (0)	I3C Slave Address (1)	I3C Slave Address (2)	I3C Slave Address (3)	I3C Slave Address (4)	I3C Slave Address (5)	I3C Slave Address (6)	I3C Slave Address (7)	I3C_bit32	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2.9. Toggled Bit Value

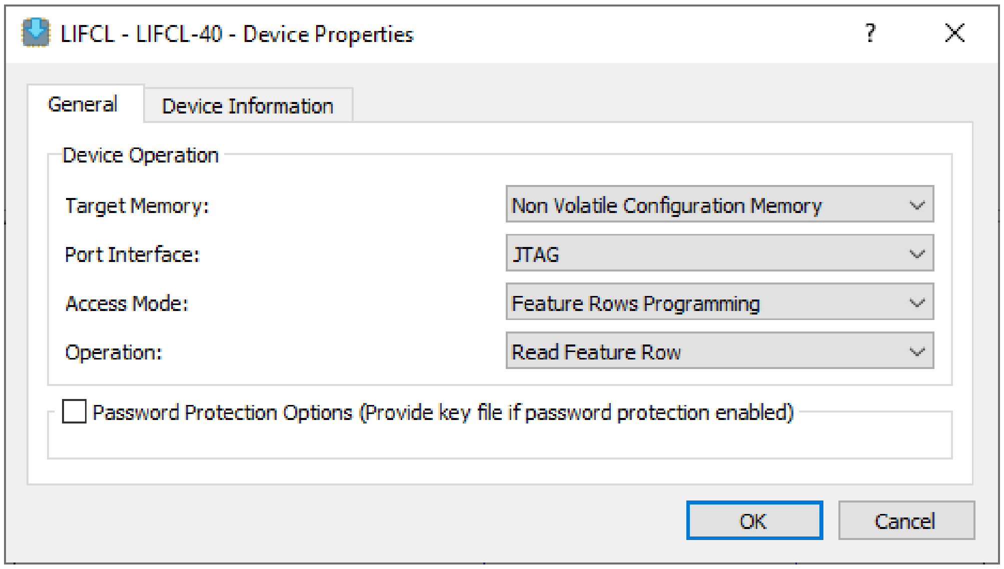
4. Click **Program** to update the settings.

   WARNING: These feature row bits are OTPs (One-time programmable). Disabling all these config pins could limit what user can do to the device as user is not able to re-configure using different interfaces SSPI or I^{2}C (instead of JTAG) as these pins are required during enabling of these interfaces.

Q: How does the user read the feature row for Nexus devices?

A: The procedure below shows the user how to read the feature row.

1. Open the Radiant Programmer and select **Read Feature Row** as **Operation** in the Device Properties. Click **OK**.


LIFCL - LIFCL-40 - Device Properties

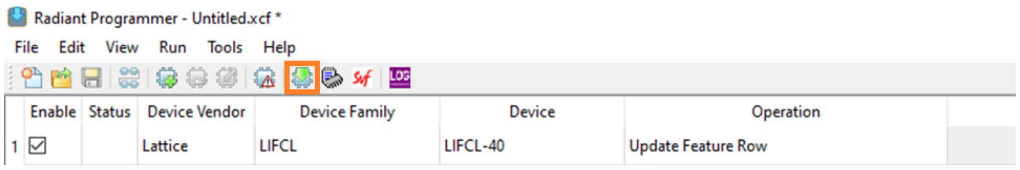
Device Operation

Target Memory:	Non Volatile Configuration Memory
Port Interface:	JTAG
Access Mode:	Feature Rows Programming
Operation:	Read Feature Row

☐ Password Protection Options (Provide key file if password protection enabled)

Figure 2.10. Device Properties

2. Click the **Program Device** button to read the feature row.

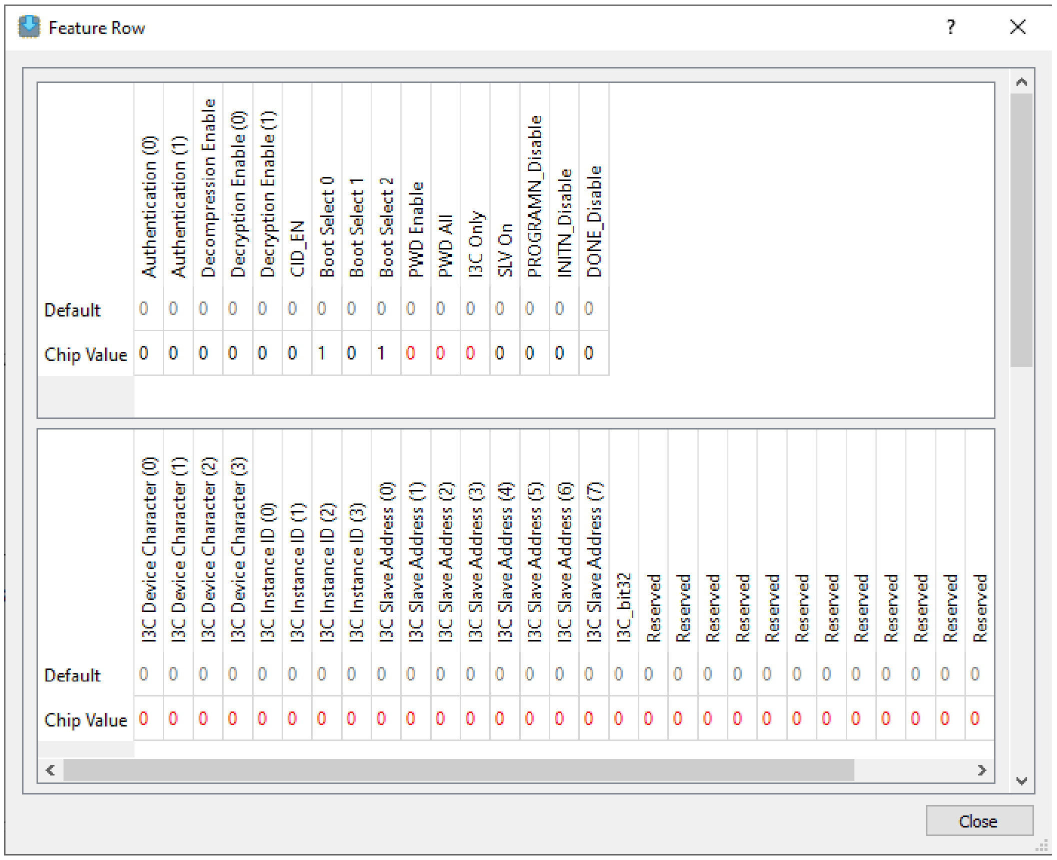

Radiant Programmer - Untitled.xcf *

	Enable	Status	Device Vendor	Device Family	Device	Operation
1	✓		Lattice	LIFCL	LIFCL-40	Update Feature Row

Figure 2.11. Program Device

The current settings of the feature row is displayed as shown in Figure 2.12.

Note: It is important for the user to be informed of the current settings of the feature row.



	Authentication (0)	Authentication (1)	Decompression Enable	Decryption Enable (0)	Decryption Enable (1)	CID_EN	Boot Select 0	Boot Select 1	Boot Select 2	PWD Enable	PWD All	I3C Only	SLV On	PROGRAMN_Disable	INITN_Disable	DONE_Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0

	I3C Device Character (0)	I3C Device Character (1)	I3C Device Character (2)	I3C Device Character (3)	I3C Instance ID (0)	I3C Instance ID (1)	I3C Instance ID (2)	I3C Instance ID (3)	I3C Slave Address (0)	I3C Slave Address (1)	I3C Slave Address (2)	I3C Slave Address (3)	I3C Slave Address (4)	I3C Slave Address (5)	I3C Slave Address (6)	I3C Slave Address (7)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	I3C_bit32	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Default	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0

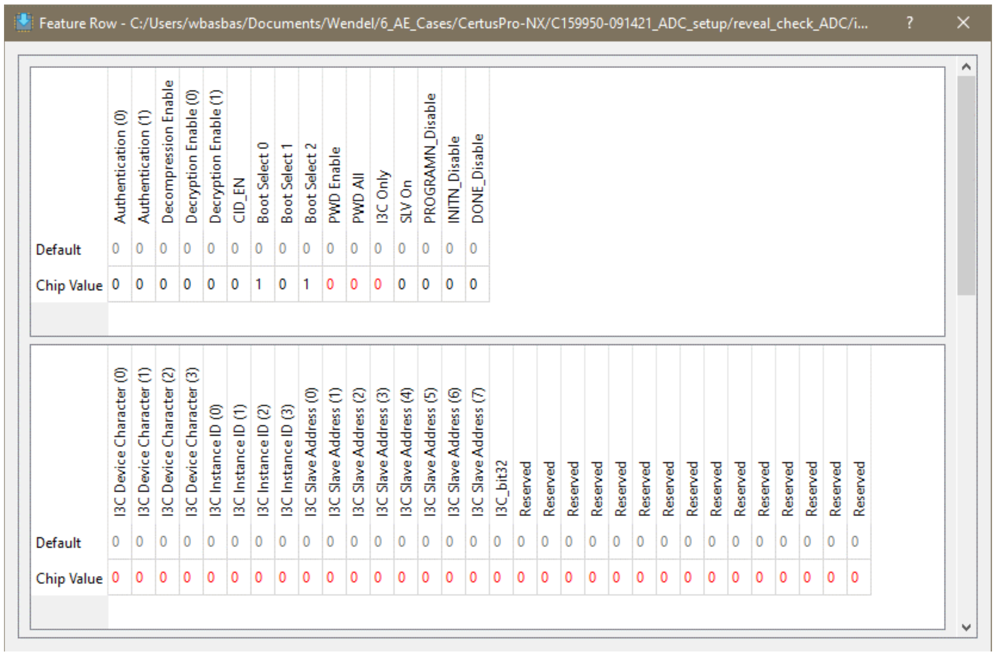
Figure 2.12. Feature Row Current Values

Q: What are the Feature-Row bits and settings?

A: The following are the tabulated list of the feature-row based settings. Some of these bits are also found under **Device Constraint Editor > Global Settings**.

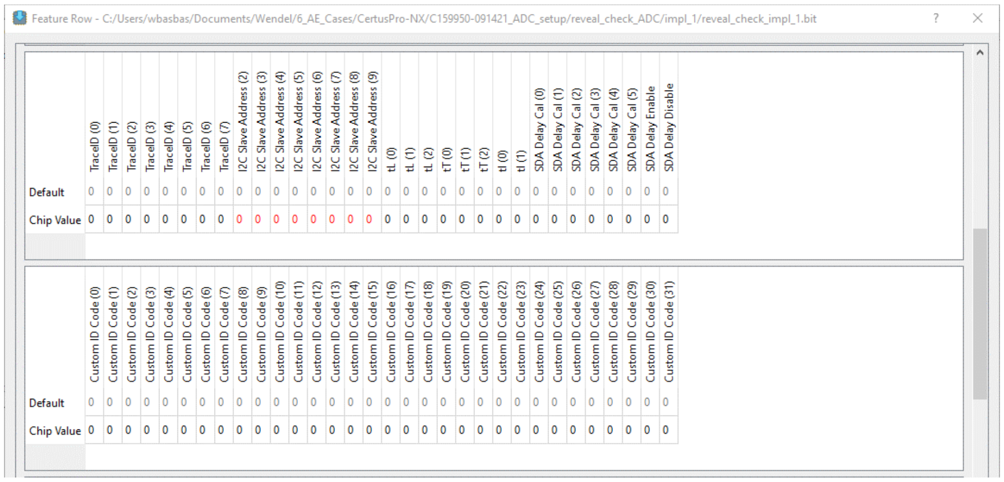
Note: If user can see any of these bits under the Global Settings, this means that they are OTP (One-Time Programmable, can only be modified once). All feature row bits which can be accessed through Radiant Programmer, as shown in Figure 2.12, are OTP.

User can also check these bits manually using the **Read Feature Row** operation using the Radiant Programmer. Refer to How does the user read the feature row for Nexus devices section for the detailed procedure. User may also look up the Global Settings Definition to understand the difference between feature-row based and non-feature row-based settings.


Feature Row - C:/Users/wbasbas/Documents/Wendel/6_AE_Cases/CertusPro-NX/C159950-091421_ADC_setup/reveal_check_ADC/i...

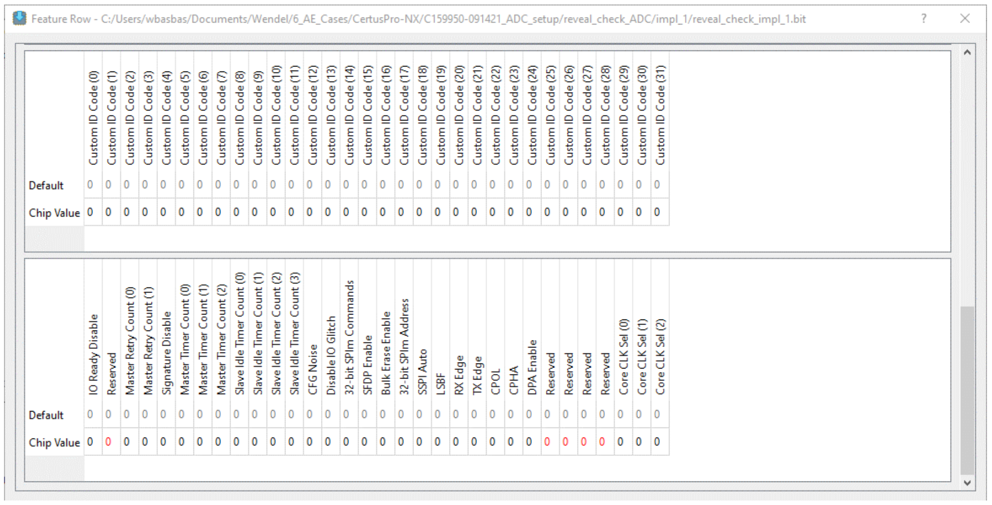
	Authentication (0)	Authentication (1)	Decompression Enable	Decryption Enable (0)	Decryption Enable (1)	CID_EN	Boot Select 0	Boot Select 1	Boot Select 2	PWD Enable	PWD All	I3C Only	SLV On	PROGRAMN_Disable	INITN_Disable	DONE_Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0

	I3C Device Character (0)	I3C Device Character (1)	I3C Device Character (2)	I3C Device Character (3)	I3C Instance ID (0)	I3C Instance ID (1)	I3C Instance ID (2)	I3C Instance ID (3)	I3C Slave Address (0)	I3C Slave Address (1)	I3C Slave Address (2)	I3C Slave Address (3)	I3C Slave Address (4)	I3C Slave Address (5)	I3C Slave Address (6)	I3C Slave Address (7)	I3C_bit32	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0


Feature Row - C:/Users/wbasbas/Documents/Wendel/6_AE_Cases/CertusPro-NX/C159950-091421_ADC_setup/reveal_check_ADC/impl_1/reveal_check_impl_1.bit

	TraceID (0)	TraceID (1)	TraceID (2)	TraceID (3)	TraceID (4)	TraceID (5)	TraceID (6)	TraceID (7)	I2C Slave Address (2)	I2C Slave Address (3)	I2C Slave Address (4)	I2C Slave Address (5)	I2C Slave Address (6)	I2C Slave Address (7)	I2C Slave Address (8)	I2C Slave Address (9)	tL (0)	tL (1)	tL (2)	tT (0)	tT (1)	tT (2)	tI (0)	tI (1)	SDA Delay Cal (0)	SDA Delay Cal (1)	SDA Delay Cal (2)	SDA Delay Cal (3)	SDA Delay Cal (4)	SDA Delay Cal (5)	SDA Delay Enable	SDA Delay Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

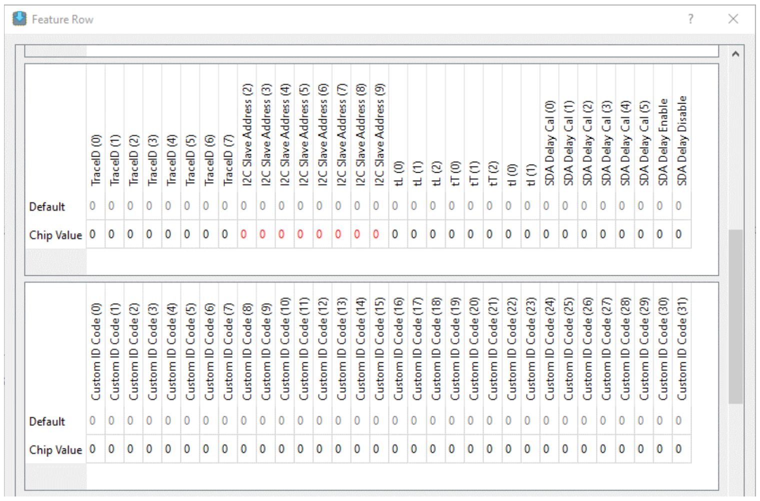
	Custom ID Code (0)	Custom ID Code (1)	Custom ID Code (2)	Custom ID Code (3)	Custom ID Code (4)	Custom ID Code (5)	Custom ID Code (6)	Custom ID Code (7)	Custom ID Code (8)	Custom ID Code (9)	Custom ID Code (10)	Custom ID Code (11)	Custom ID Code (12)	Custom ID Code (13)	Custom ID Code (14)	Custom ID Code (15)	Custom ID Code (16)	Custom ID Code (17)	Custom ID Code (18)	Custom ID Code (19)	Custom ID Code (20)	Custom ID Code (21)	Custom ID Code (22)	Custom ID Code (23)	Custom ID Code (24)	Custom ID Code (25)	Custom ID Code (26)	Custom ID Code (27)	Custom ID Code (28)	Custom ID Code (29)	Custom ID Code (30)	Custom ID Code (31)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0


Feature Row - C:/Users/wbasbas/Documents/Wendel/6_AE_Cases/CertusPro-NX/C159950-091421_ADC_setup/reveal_check_ADC/impl_1/reveal_check_impl_1.bit

	Custom ID Code (0)	Custom ID Code (1)	Custom ID Code (2)	Custom ID Code (3)	Custom ID Code (4)	Custom ID Code (5)	Custom ID Code (6)	Custom ID Code (7)	Custom ID Code (8)	Custom ID Code (9)	Custom ID Code (10)	Custom ID Code (11)	Custom ID Code (12)	Custom ID Code (13)	Custom ID Code (14)	Custom ID Code (15)	Custom ID Code (16)	Custom ID Code (17)	Custom ID Code (18)	Custom ID Code (19)	Custom ID Code (20)	Custom ID Code (21)	Custom ID Code (22)	Custom ID Code (23)	Custom ID Code (24)	Custom ID Code (25)	Custom ID Code (26)	Custom ID Code (27)	Custom ID Code (28)	Custom ID Code (29)	Custom ID Code (30)	Custom ID Code (31)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

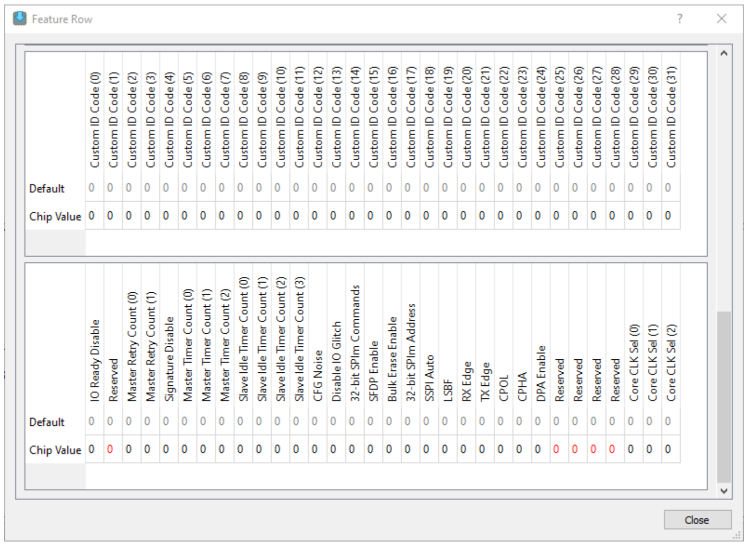
	IO Ready Disable	Reserved	Master Retry Count (0)	Master Retry Count (1)	Signature Disable	Master Timer Count (0)	Master Timer Count (1)	Master Timer Count (2)	Slave Idle Timer Count (0)	Slave Idle Timer Count (1)	Slave Idle Timer Count (2)	Slave Idle Timer Count (3)	CFG Noise	Disable IO Glitch	32-bit SPIm Commands	SFDP Enable	Bulk Erase Enable	32-bit SPIm Address	SSPI Auto	LSBF	RX Edge	TX Edge	CPOL	CPHA	DPA Enable	Reserved	Reserved	Reserved	Reserved	Core CLK Sel (0)	Core CLK Sel (1)	Core CLK Sel (2)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2.13. CertusPro-NX Feature-Row Bits



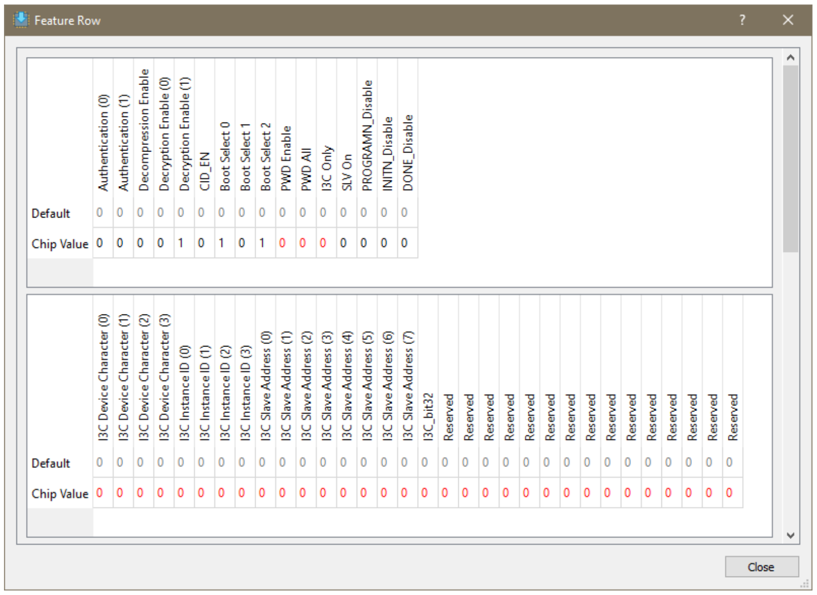
	TraceID (0)	TraceID (1)	TraceID (2)	TraceID (3)	TraceID (4)	TraceID (5)	TraceID (6)	TraceID (7)	I2C Slave Address (2)	I2C Slave Address (3)	I2C Slave Address (4)	I2C Slave Address (5)	I2C Slave Address (6)	I2C Slave Address (7)	I2C Slave Address (8)	I2C Slave Address (9)	tL (0)	tL (1)	tL (2)	tT (0)	tT (1)	tT (2)	tI (0)	tI (1)	SDA Delay Cal (0)	SDA Delay Cal (1)	SDA Delay Cal (2)	SDA Delay Cal (3)	SDA Delay Cal (4)	SDA Delay Cal (5)	SDA Delay Enable	SDA Delay Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	Custom ID Code (0)	Custom ID Code (1)	Custom ID Code (2)	Custom ID Code (3)	Custom ID Code (4)	Custom ID Code (5)	Custom ID Code (6)	Custom ID Code (7)	Custom ID Code (8)	Custom ID Code (9)	Custom ID Code (10)	Custom ID Code (11)	Custom ID Code (12)	Custom ID Code (13)	Custom ID Code (14)	Custom ID Code (15)	Custom ID Code (16)	Custom ID Code (17)	Custom ID Code (18)	Custom ID Code (19)	Custom ID Code (20)	Custom ID Code (21)	Custom ID Code (22)	Custom ID Code (23)	Custom ID Code (24)	Custom ID Code (25)	Custom ID Code (26)	Custom ID Code (27)	Custom ID Code (28)	Custom ID Code (29)	Custom ID Code (30)	Custom ID Code (31)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



	Custom ID Code (0)	Custom ID Code (1)	Custom ID Code (2)	Custom ID Code (3)	Custom ID Code (4)	Custom ID Code (5)	Custom ID Code (6)	Custom ID Code (7)	Custom ID Code (8)	Custom ID Code (9)	Custom ID Code (10)	Custom ID Code (11)	Custom ID Code (12)	Custom ID Code (13)	Custom ID Code (14)	Custom ID Code (15)	Custom ID Code (16)	Custom ID Code (17)	Custom ID Code (18)	Custom ID Code (19)	Custom ID Code (20)	Custom ID Code (21)	Custom ID Code (22)	Custom ID Code (23)	Custom ID Code (24)	Custom ID Code (25)	Custom ID Code (26)	Custom ID Code (27)	Custom ID Code (28)	Custom ID Code (29)	Custom ID Code (30)	Custom ID Code (31)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

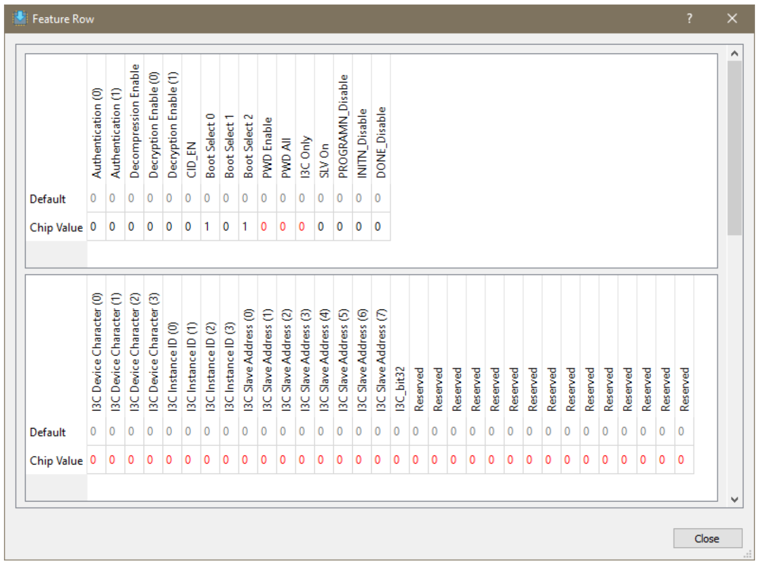
	IO Ready Disable	Reserved	Master Retry Count (0)	Master Retry Count (1)	Signature Disable	Master Timer Count (0)	Master Timer Count (1)	Master Timer Count (2)	Slave Idle Timer Count (0)	Slave Idle Timer Count (1)	Slave Idle Timer Count (2)	Slave Idle Timer Count (3)	CFG Noise	Disable IO Glitch	32-bit SPIm Commands	SFDP Enable	Bulk Erase Enable	32-bit SPIm Address	SSPI Auto	LSBF	RX Edge	TX Edge	CPOL	CPHA	DPA Enable	Reserved	Reserved	Reserved	Reserved	Core CLK Sel (0)	Core CLK Sel (1)	Core CLK Sel (2)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



	Authentication (0)	Authentication (1)	Decompression Enable	Decryption Enable (0)	Decryption Enable (1)	CID_EN	Boot Select 0	Boot Select 1	Boot Select 2	PWD Enable	PWD All	I3C Only	SLV On	PROGRAMN_Disable	INITN_Disable	DONE_Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0

	I3C Device Character (0)	I3C Device Character (1)	I3C Device Character (2)	I3C Device Character (3)	I3C Instance ID (0)	I3C Instance ID (1)	I3C Instance ID (2)	I3C Instance ID (3)	I3C Slave Address (0)	I3C Slave Address (1)	I3C Slave Address (2)	I3C Slave Address (3)	I3C Slave Address (4)	I3C Slave Address (5)	I3C Slave Address (6)	I3C Slave Address (7)	I3C_bit32	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2.14. Certus-NX Feature-Row Bits

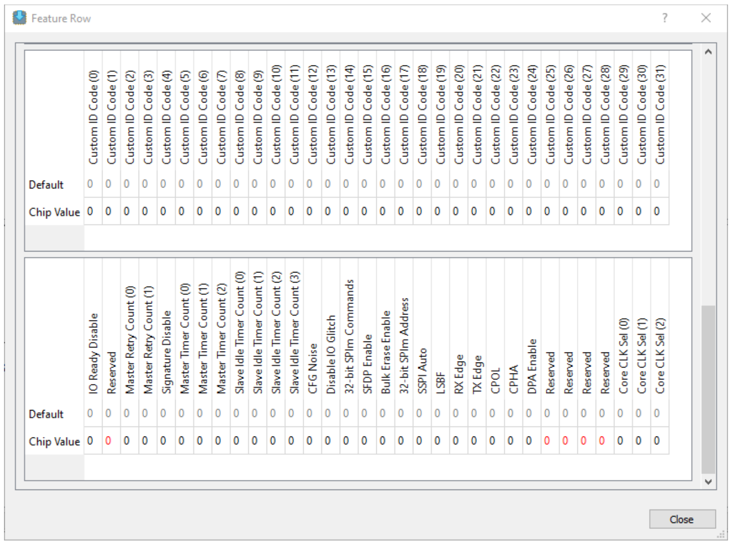


Feature Row

	Authentication (0)	Authentication (1)	Decompression Enable	Decryption Enable (0)	Decryption Enable (1)	CID_EN	Boot Select 0	Boot Select 1	Boot Select 2	PWD Enable	PWD All	I3C Only	SLV On	PROGRAMN_Disable	INITN_Disable	DONE_Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0

	I3C Device Character (0)	I3C Device Character (1)	I3C Device Character (2)	I3C Device Character (3)	I3C Instance ID (0)	I3C Instance ID (1)	I3C Instance ID (2)	I3C Instance ID (3)	I3C Slave Address (0)	I3C Slave Address (1)	I3C Slave Address (2)	I3C Slave Address (3)	I3C Slave Address (4)	I3C Slave Address (5)	I3C Slave Address (6)	I3C Slave Address (7)	I3C_bit32	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Close

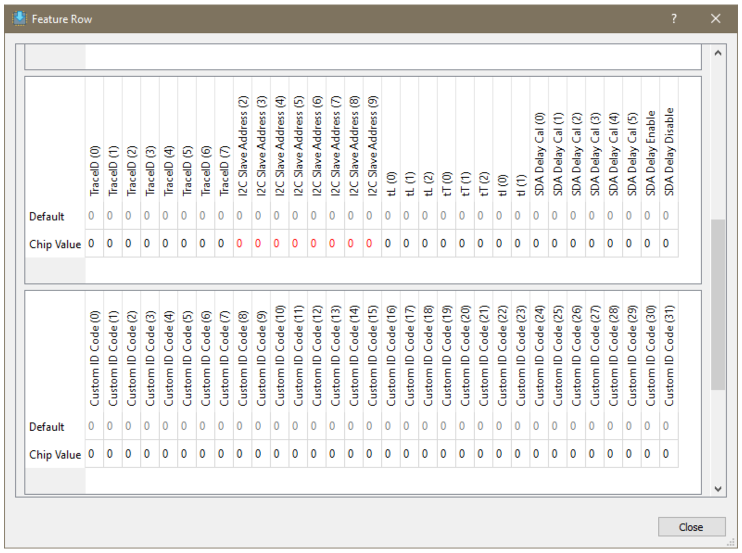


Feature Row

	Custom ID Code (0)	Custom ID Code (1)	Custom ID Code (2)	Custom ID Code (3)	Custom ID Code (4)	Custom ID Code (5)	Custom ID Code (6)	Custom ID Code (7)	Custom ID Code (8)	Custom ID Code (9)	Custom ID Code (10)	Custom ID Code (11)	Custom ID Code (12)	Custom ID Code (13)	Custom ID Code (14)	Custom ID Code (15)	Custom ID Code (16)	Custom ID Code (17)	Custom ID Code (18)	Custom ID Code (19)	Custom ID Code (20)	Custom ID Code (21)	Custom ID Code (22)	Custom ID Code (23)	Custom ID Code (24)	Custom ID Code (25)	Custom ID Code (26)	Custom ID Code (27)	Custom ID Code (28)	Custom ID Code (29)	Custom ID Code (30)	Custom ID Code (31)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	IO Ready Disable	Reserved	Master Retry Count (0)	Master Retry Count (1)	Signature Disable	Master Timer Count (0)	Master Timer Count (1)	Master Timer Count (2)	Slave Idle Timer Count (0)	Slave Idle Timer Count (1)	Slave Idle Timer Count (2)	Slave Idle Timer Count (3)	CFG Noise	Disable IO Glitch	32-bit SPIm Commands	SFDP Enable	Bulk Erase Enable	32-bit SPIm Address	SSPI Auto	LSBF	RX Edge	TX Edge	CPOL	CPHA	DPA Enable	Reserved	Reserved	Reserved	Reserved	Core CLK Sel (0)	Core CLK Sel (1)	Core CLK Sel (2)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Close



Feature Row

	TraceID (0)	TraceID (1)	TraceID (2)	TraceID (3)	TraceID (4)	TraceID (5)	TraceID (6)	TraceID (7)	I2C Slave Address (2)	I2C Slave Address (3)	I2C Slave Address (4)	I2C Slave Address (5)	I2C Slave Address (6)	I2C Slave Address (7)	I2C Slave Address (8)	I2C Slave Address (9)	tL (0)	tL (1)	tL (2)	tT (0)	tT (1)	tT (2)	tI (0)	tI (1)	SDA Delay Cal (0)	SDA Delay Cal (1)	SDA Delay Cal (2)	SDA Delay Cal (3)	SDA Delay Cal (4)	SDA Delay Cal (5)	SDA Delay Enable	SDA Delay Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	Custom ID Code (0)	Custom ID Code (1)	Custom ID Code (2)	Custom ID Code (3)	Custom ID Code (4)	Custom ID Code (5)	Custom ID Code (6)	Custom ID Code (7)	Custom ID Code (8)	Custom ID Code (9)	Custom ID Code (10)	Custom ID Code (11)	Custom ID Code (12)	Custom ID Code (13)	Custom ID Code (14)	Custom ID Code (15)	Custom ID Code (16)	Custom ID Code (17)	Custom ID Code (18)	Custom ID Code (19)	Custom ID Code (20)	Custom ID Code (21)	Custom ID Code (22)	Custom ID Code (23)	Custom ID Code (24)	Custom ID Code (25)	Custom ID Code (26)	Custom ID Code (27)	Custom ID Code (28)	Custom ID Code (29)	Custom ID Code (30)	Custom ID Code (31)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Close

Figure 2.15. CrossLink-NX Feature-Row Bits

Q: Is OTP the only way to force the PROGRAMN, INITN, and DONE ports to be disabled so that they can be used as standard I/O?

A: Yes, the only way to use the PROGRAMN, INITN, and DONE ports as GPIO is to program the OTP feature row.

Q: The PROGRAMN pin is not working correctly as GPIO (applicable to INITN and DONE) when user sets the PROGRAMN_PORT to DISABLE in the Global Settings of the Device Constraint Editor and load the design to the Crosslink-NX board.

A: The SW setting on the PROGRAMN_PORT=Disable is only to allows user to assign the GPIO to the PROGRAMN pin in the Radiant SW flow, it does not automatically set the Feature Row PROGRAMN_DISABLE (Radiant Programmer) bit accordingly. It is needed to modify the feature row value separately.

Q: User wants to program the device through SSPI and it is not possible if the PROGRAMN is already disabled in the feature row.

A: For the slave configuration port programing after the PRGRAMN pin is disabled, it should be good but there some things the user needs to be aware of:

- If only PROGRAMN pin is disabled, but the INITN pin is not disabled, user could drive the INITN pin (Open Drain output) low to hold off the bitstream booting, then activate the Slave configuration port to perform the programming.
- If both PROGRAMN pin and INITN pin are disabled, there are two possible scenarios:
  - There are no valid bitstream to boot (for example a brand new board). The user have to wait a little longer ((t_{INIT} + preamble count time) for single boot setup, or (t_{INIT} + 2 × preamble count time) for dual boot setup), then activate the slave configuration port to perform the programming. After the booting sequence finished and the booting failed, the device goes to unprogrammed state. Then the slave configuration port could be activated the same way as holding PORGRAMN pin.
  - If the board been successfully programmed before, the device boots from the bitstream since both PROGRAMN pin and INITN pin are disabled. Then the user have to persist the target slave configuration port in the design (bitstream), so user could access the slave configuration port in user function mode to perform the device programming.

Q: DONE/INITN/PROGRAMN ports are not disabled even it was disabled on constraint.

A: Update the registers for these ports go to the Feature rows programming (using the Radiant Programmer) then update and read feature row. Change the register values for DONE/INITN/PROGRAMN ports disable from *0* to *1*.

3. Boot Process

This section provides an overview on a series of events from device power on up to entering user mode.

3.1. Terminologies

3.1.1. Master Port Configuration Process and Flow

Prior to becoming operational, the FPGA goes through a sequence of states including initialization, configuration, and wake-up.

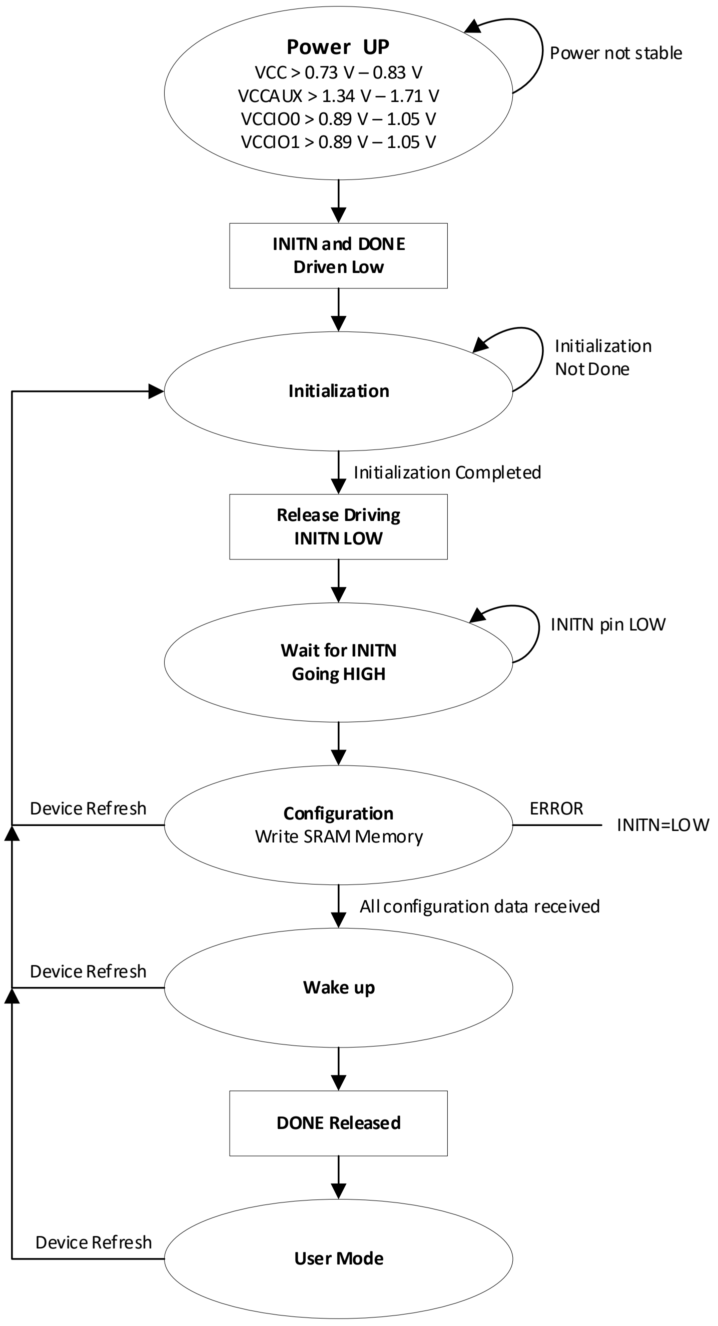


Figure 3.1. Master Port Configuration Flow

3.1.2. Power-Up Sequence

For the Nexus device to operate, power must be applied to the device. During a short period of time, as the voltages applied to the system rise, the FPGA stays in an indeterminate state.
As power continues to ramp, a Power-On-Reset (POR) circuit inside the FPGA becomes active. The POR circuit, once active, makes sure the external I/O pins are in a high-impedance state. It also monitors the V_{CC}, V_{CCAUX}, V_{CCIO0}, and V_{CCIO1} input rails. The POR circuit waits for the following conditions:

- V_{CC} > 0.73 V – 0.83 V
- V_{CCAUX} > 1.34 V – 1.71 V
- V_{CCIO0} > 0.89 V – 1.05 V
- V_{CCIO1} > 0.89 V – 1.05 V

Power-On-Reset (POR) puts the Nexus device into a reset state. There is no power up sequence required for the Nexus device.

Table 3.1. Power-On Reset

Symbol	Parameter		Min	Typ	Max	Unit
V_{PORUP}	Power-On-Reset ramp-up trip point (Monitoring V_{CC}, V_{CCAUX}, V_{CCIO0}, and V_{CCIO1})	V_{CC}	0.73	—	0.83	V
		V_{CCAUX}	1.34	—	1.71	V
		V_{CCIO0},V_{CCIO1}	0.89	—	1.05	V
V_{PORDN}	Power-On-Reset ramp-up trip point (Monitoring V_{CC} and V_{CCAUX})	V_{CC}	0.51	—	0.81	V
		V_{CCAUX}	1.38	—	1.54	V

When these conditions are met the POR circuit releases an internal reset strobe, allowing the device to begin its initialization process. The Nexus device asserts INITN active low, and drives DONE low. When INITN and DONE are asserted low, the device moves to the initialization state, as shown in Figure 3.2.

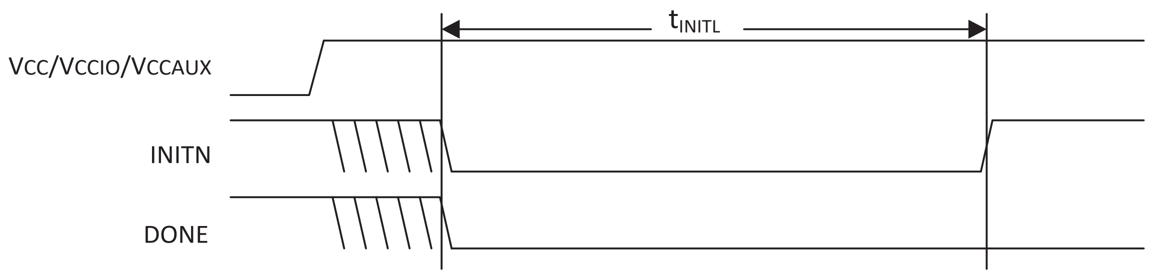


Figure 3.2. Configuration from Power-On-Reset Timing

3.1.3. Initialization

The Nexus device enters the memory initialization phase immediately after the Power On Reset circuit drives the INITN and DONE status pins low. The purpose of the initialization state is to clear all the SRAM memory inside the FPGA.

The FPGA remains in the initialization state until all the following conditions are met:

- The t_{INITL} time period has elapsed
- The PROGRAMN pin is deasserted
- The INITN pin is no longer asserted low by an external master

The dedicated INITN pin provides two functions during the initialization phase. The first is to indicate the FPGA is currently clearing its configuration SRAM. The second is to act as an input preventing the transition from the initialization state to the configuration state.

During the t_{INITL} time period the FPGA is clearing the configuration SRAM. When the Nexus device is part of a chain of devices each device has different t_{INITL} initialization time. The FPGA with the slowest t_{INITL} parameter can prevent other devices in the chain from starting to configure. Premature release of the INITN in a multi-device chain may cause configuration of one or more chained devices to fail to configure intermittently.

The active-low, open-drain initialization signal INITN must be pulled high by an external resistor when initialization is complete. To synchronize the configuration of multiple FPGAs, one or more INITN pins should be wire-ANDed. If one or more FPGAs or an external device holds INITN low, the FPGA remains in the initialization state.

The GPIO of the device at power-up defaults to tri-stated outputs with active weak input pull-downs. After configuration, all GPIO included in the user design wake up in the user-defined condition. GPIO not defined in the user design remain output tri-stated and the input with a weak pull-down enabled. This default avoids inadvertent effects of the inputs rising while powering up. In some cases, this can cause a problem if other connected devices on the board reset or trigger from an active high signal.

Before/during configuration, the dedicated JTAG_EN pin has pull-down, and dual-purpose sysCONFIG I/O with pull-up, which are excluded from the GPIO default setting.

3.1.4. Configuration

The releasing HIGH on the INITN pin causes the FPGA to enter the configuration state. The FPGA can accept the configuration bitstream created by the Lattice Radiant Software.

Following power-up, the Nexus device begins the external SPI flash boot process with the Signature Verification phase. The Nexus device attempts a signature read-back in a finite loop until the expected result is received or the loop count is exceeded. A correct signature read allows the Nexus device to proceed to the Preamble Verification phase immediately. Opposed to waiting for a fixed power-ramp timer to expire, this process allows for the fastest possible boot times.

The Signature Verification process verifies either the Lattice Specified LSCC signature or JEDEC Standard SFDP, depending on the value of Control Register 1, Bit 15 (CR1 [15]). If CR1 [15] is *0* (default), the Nexus device reads the boot bitstream image from the base boot address and check for the LSCC signature (0x4C534343) using SPI flash command code 8’H03. If CR1 [15] is *1*, the Nexus device performs a SFDP read (SPI flash command code 8’H5A) and checking for SFDP code (0x50444653) contained in SFPD compliant SPI flash devices. The Nexus device retries the SFDP/LSCC signature read until three consecutive matches are found. When successful, the Nexus device sets the internal Signature Successful Flag and proceeds to the Preamble Verification step. If the loop timer expires (600,000 SPI clock cycles, or about 150 ms) the device proceeds to the Preamble Verifications step without setting the Signature Successful Flag. When the internal Signature Successful Flag is set, the Signature Verification phase is bypassed for subsequent warm-boot events (for example, the PROGRAMN pin toggle or REFRESH command).

For proper bitstream data alignment, the bitstream Preamble must be detected once. If the preamble does not come before the preamble timer (counting for 600,000 SPI clock cycles) expires, a boot failure is declared, and the boot process aborts.

Once the preamble is detected, the Nexus device continues fetching data from non-volatile memory to configure the FPGA SRAM memory. The Nexus device does not leave the Configuration state if there is no valid configuration data.

INITN is used to indicate an error exists in the configuration data. When INITN is high, configuration is proceeding without issue. If INITN is asserted low, an error has occurred and the FPGA does not operate.

3.1.5. Wake Up

Wake-up is the transition from configuration mode to user mode. The Nexus device's fixed four-phase wake-up sequence starts when the device has correctly received all of its configuration data. User can arrange the order of these four phases to meet specific implementation requirements. When all configuration data is received, the FPGA asserts an internal DONE status bit. The assertion of the internal DONE causes a Wake Up state machine to run that sequences four controls. The four control strobes are:

- Global Set/Reset (GSRN)
- Global Write Enable (GWE)
- Global Output Enable (GOE)
- External DONE

One phase of the Wake-Up process is for the FPGA to release the Global Output Enable. When it is asserted, permits the FPGA's I/O to exit a high-impedance state and take on the programmed output function. The FPGA inputs are always active. The input signals are prevented from performing any action on the FPGA flip-flops by the assertion of the Global Set/Reset (GSRN).

Other phases of the Wake-Up process release the Global Set/Reset and the Global Write Enable controls.

The Global Set/Reset is an internal strobe that, when asserted, causes all I/O flip-flops, Look Up Table (LUT) flip-flops, distributed RAM output flip-flops, and Embedded Block RAM output flip-flops that have the GSR enabled attribute to be set/cleared per the hardware description language definition.

The Global Write Enable is a control that overrides the write enable strobe for all RAM logic inside the FPGA. As mentioned previously, the inputs on the FPGA are always active. Keeping GWE deasserted prevents accidental corruption of the instantiated RAM resources inside the FPGA.

Another phase of the Wake-Up process asserts the external DONE pin. The external DONE is a bi-directional, open-drain I/O only when it is enabled. An external agent that holds the external DONE pin low prevents the wake-up process of the FPGA from proceeding. Only after the external DONE, if enabled, is active high does the final wake-up phase complete. Wake-Up completes uninterrupted when the external DONE pin is not enabled.

Once the final wake-up phase is complete, the FPGA enters user mode.

3.1.6. Early I/O Release

The Nexus device supports an Early I/O Release feature, which allows the I/O that reside in the I/O Banks on the left and right of the device (I/O Bank 1, I/O Bank 2, I/O Bank 6, and I/O Bank 7), to assume user-defined drive states at the beginning of bistream processing. The Early I/O Release feature releases the I/O after processing the I/O configuration for the left and right banks, which is located near the head of the bitstream data. Once data is programmed in the left/right Memory Interface Block (MIB) the I/O is released to a predefined state. This feature is enabled by setting the EARLY_IO_RELEASE preferences to ON in the Lattice Radiant Device Constraint Editor.

In addition, Early I/O Release requires user to instantiate an output buffer register with an asynchronous set or reset function, to indicate the desired drive 1 or drive 0 behavior, respectively, during the Early Release period. Unregistered outputs in Early-Release banks drive High-Z until full device configuration is complete. Be aware that some of the I/O in Bank 1, including the dual-purpose sysCONFIG I/O, cannot be utilized as Early Released I/O. Also, if the ECDSA bitstream authentication is enable for the Nexus device, the Early I/O Release feature is not supported.

3.1.7. User Mode

The Nexus device enters user mode immediately following the Wake-Up sequence has completed. User Mode is the point in time when the Nexus device begins performing the logic operations the user designed. The device remains in this state until one of three events occurs:

- The PROGRAMN input pin is asserted
- A REFRESH command is received through one of the configuration ports
- Power is cycled or power supply levels drop below the specified trigger levels

3.1.8. Clearing the Configuration Memory and Re-Initialization

The current user mode configuration of the Nexus device remains in operation until they are actively cleared, or power is lost. Several methods are available to clear the internal configuration memory of the Nexus device. The first is to remove power and reapply power. Another method is to toggle the PROGRAMN pin. Lastly, user can reinitialize the memory through a Refresh command. Any active configuration port can be used to send a Refresh command.

Invoking one of these methods causes the Nexus device to drive INITN and DONE low. The Nexus device enters the initialization state as described earlier.

3.1.9. Reconfiguration Priority

There are many sources that can initiate a reconfiguration while a configuration is already in process. When initiating a reconfiguration, the sources are prioritized depending on which of them initiated the original configuration. Note that if an interruption occurs, the reconfiguration occurs without informing the original configuration source that the configuration did not complete. JTAG always has the highest priority and any JTAG initiated configuration event causes a reconfiguration to occur. Toggling the PROGRAMN pin has the next highest priority. It interrupts any current configuration other than a JTAG configuration.

3.1.10. Dual Boot

Switches to load from the second known good (Golden) pattern when the first pattern becomes corrupted.

3.1.11. Ping-Pong Boot

Switches between two bitstream patterns based on the user's choice. If the system fails to boot from one of the bitstreams, it automatically boots from the second bitstream.

3.1.12. Multi-Boot

Allows the system to dynamically switch between two to five bitstream patterns while still being protected with a Golden (sixth) pattern.

3.1.13. MASTER_SPI_PORT

The MASTER_SPI_PORT allows user to preserve the Master SPI configuration port after the Nexus device enters user mode. There are four states to which the MASTER_SPI_PORT preference can be set:

- DISABLE – This setting disconnects the Master SPI port pins from the configuration logic. The Master SPI pins (MCLK, MCSN, MOSI/MD0, MISO/MD1, MD2 and MD3) could be general purpose user I/O.
- SERIAL – This setting preserves the SPI port I/O (MCLK, MCSN, MOSI/MD0, MISO/MD1) in serial mode when the Nexus device is in user mode. The preference also prevents user from over-assigning I/O to those pins.
- DUAL – This setting preserves the SPI port I/O (MCLK, MCSN, MOSI/MD0, MISO/MD1) in dual mode when the Nexus device is in user mode. The preference also prevents user from over-assigning I/O to those pins.
- QUAD – This setting preserves the SPI port I/O (MCLK, MCSN, MOSI/MD0, MISO/MD1, MD2 and MD3) in quad mode when the Nexus device is in user mode. The preference also prevents user from over-assigning I/O to those pins.

The default setting for this preference is DISABLE.

3.1.14. BOOTMODE

The BOOTMODE preference allows user to select the booting mode at power up (POR), PROGRAMN pin toggle or REFRESH command execution.

- DUAL – Perform the dual boot for booting event. In case the primary booting image (bitstream) is failed, the secondary (Golden) booting image is automatically invoked to boot up the device.
- SINGLE – Perform the single boot only. If failed, device go to unprogrammed mode directly.
- NONE – No Master SPI booting at POR, PROGRAMN pin toggle or REFRESH command execution, wait for Slave Configuration Port to configure the device.

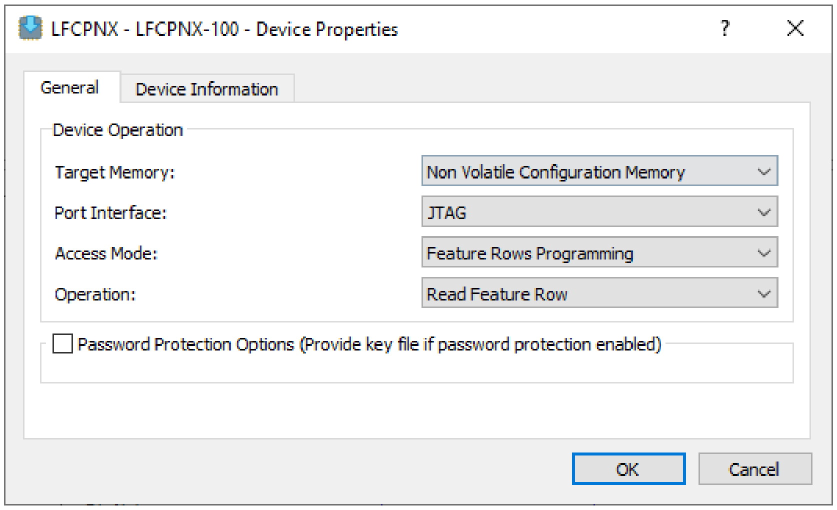
The default BOOTMODE selection is DUAL mode.

3.2. Frequently Asked Questions

Q: What settings are needed to run dual boot on FPGA?

A: The user should make sure that the Boot Select[2:0] bits are set to 101. It is not required to ENABLE the MASTER_SPI_PORT in the Radiant Software Global Settings. The default value for the Boot Select bits is 101. The user can access (R/W) the Boot Select bits using Radiant Programmer via Feature Rows Programming Operation.

Note: All the feature row bits are OTP. The user should not modify any bit without knowing the consequences.



LFCPNX - LFCPNX-100 - Device Properties

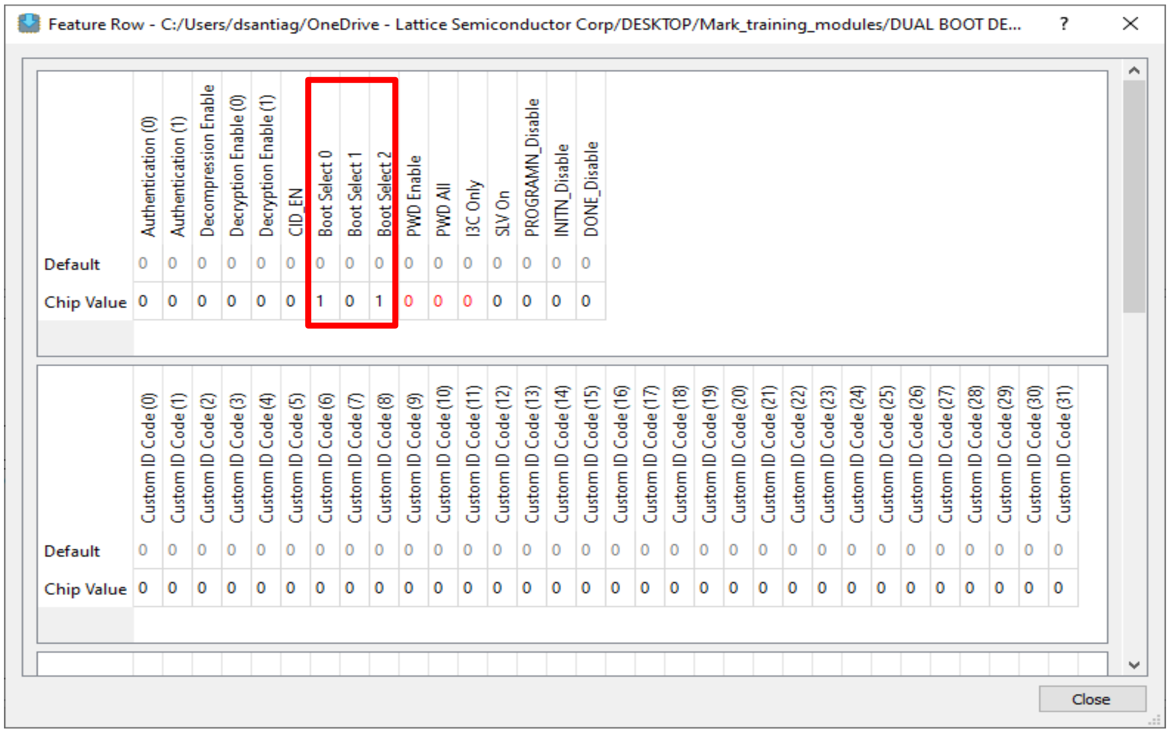
General | Device Information

Device Operation

Target Memory:	Non Volatile Configuration Memory
Port Interface:	JTAG
Access Mode:	Feature Rows Programming
Operation:	Read Feature Row

☐ Password Protection Options (Provide key file if password protection enabled)

Figure 3.3. Feature Rows Programming Operation



Feature Row - C:/Users/dsantiag/OneDrive - Lattice Semiconductor Corp/DESKTOP/Mark_training_modules/DUAL BOOT DE...

	Authentication (0)	Authentication (1)	Decompression Enable	Decryption Enable (0)	Decryption Enable (1)	CID_EN	Boot Select 0	Boot Select 1	Boot Select 2	PWD Enable	PWD All	I3C Only	SLV On	PROGRAMN_Disable	INITN_Disable	DONE_Disable
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0

	Custom ID Code (0)	Custom ID Code (1)	Custom ID Code (2)	Custom ID Code (3)	Custom ID Code (4)	Custom ID Code (5)	Custom ID Code (6)	Custom ID Code (7)	Custom ID Code (8)	Custom ID Code (9)	Custom ID Code (10)	Custom ID Code (11)	Custom ID Code (12)	Custom ID Code (13)	Custom ID Code (14)	Custom ID Code (15)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	Custom ID Code (16)	Custom ID Code (17)	Custom ID Code (18)	Custom ID Code (19)	Custom ID Code (20)	Custom ID Code (21)	Custom ID Code (22)	Custom ID Code (23)	Custom ID Code (24)	Custom ID Code (25)	Custom ID Code (26)	Custom ID Code (27)	Custom ID Code (28)	Custom ID Code (29)	Custom ID Code (30)	Custom ID Code (31)
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chip Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.4. Boot Select Bits

Q: Is it possible to program a dual boot image, then a user-data image?

A: Yes, it is possible. Assuming the user have already programmed the dual boot image, user can still program a user data image using the programmer. The EPV operation (erase, program, and verify) allows user to select a start address for the operation. The end address would be automatically calculated depending on the user data size.

Only the addresses from the start address to the end address would be affected by the EPV operation. So in this case, user must choose the address limits that does not affect the primary bitstream, golden bitstreams, and the jump instruction at the end of the Flash.

4. Programming the Device

This section provides workaround on the commonly encountered issues when programming the device.

4.1. Terminologies

4.1.1. SRAM

Static Random Access Memory (static RAM or SRAM) is a type of random-access memory, which is a volatile memory; data is lost when power is removed.

4.1.2. SPI Flash

SPI Flash is a type of flash memory interfaced with SPI (Serial Peripheral Interface). Flash memory is non-volatile (NVM), it means that the device can retain data without requiring a constant power supply, allowing devices to store information even when powered off.

4.1.3. TCK (JTAG)

Test Clock is a type of signal that synchronizes the internal state machine operations.

4.1.4. FTDI (Chip)

FTDI is a type of chip made by Future Technologies Devices International. They specialize in chips that connect directly to USB, then offer a different interface that is more convenient for simple hardware to connect to.

4.1.5. Bitstream

The FPGA bitstream is a file that contains the programming information for an FPGA such as configuration, usercode, and others.

4.1.6. JTAG_PORT

The JTAG_PORT allows user to preserve the JTAG configuration port after the Nexus device enters user mode. There are two states to which JTAG_PORT preference can be set:

- ENABLE – This setting preserves the JTAG port pins (TCK, TMS, TDI, and TDO) when the Nexus device is in user mode. The preference also prevents user from over-assigning I/O to those pins.
- DISABLE – This setting disconnects the JTAG port pins (TCK, TMS, TDI, and TDO) from the configuration logic. It allows those ports pins to be general purpose I/O, if they are not persisted for Slave SPI port as well.

The default setting for this preference is DISABLE. (Radiant 3.0 =DISABLED, Radiant 3.1 = ENABLED).

4.1.7. MASTER_SPI_PORT

The MASTER_SPI_PORT allows user to preserve the Master SPI configuration port after the Nexus device enters user mode. There are four states to which the MASTER_SPI_PORT preference can be set:

- DISABLE – This setting disconnects the Master SPI port pins from the configuration logic. The Master SPI pins (MCLK, MCSN, MOSI/MD0, MISO/MD1, MD2 and MD3) could be general purpose user I/O.
- SERIAL – This setting preserves the SPI port I/O (MCLK, MCSN, MOSI/MD0, MISO/MD1) in serial mode when the Nexus device is in user mode. The preference also prevents user from over-assigning I/O to those pins.
- DUAL – This setting preserves the SPI port I/O (MCLK, MCSN, MOSI/MD0, MISO/MD1) in dual mode when the Nexus device is in user mode. The preference also prevents user from over-assigning I/O to those pins.
- QUAD – This setting preserves the SPI port I/O (MCLK, MCSN, MOSI/MD0, MISO/MD1, MD2 and MD3) in quad mode when the Nexus device is in user mode. The preference also prevents user from over-assigning I/O to those pins.

The default setting for this preference is DISABLE.

4.2. Frequently Asked Questions

Q: How to solve SRAM programming issue (for example: Erase, Program and Verify) when user is using Radiant 3.0?

A: Radiant 3.0 sysCONFIG has default setting for the JTAG_PORT as DISABLED. This can lead to JTAG access issue on the device after loading the generated bitstream. For example, SRAM verify and even erase afterwards (if the previous bitstream using JTAG_PORT= disabled).

Possible affected boards are CrossLink™-NX VIP Sensor Input board, CrossLink-NX Evaluation board, CrossLink-NX PCIe Bridge Board, and others.

There are two ways user can perform to solve this issue:

- Remove the bitstream (with JTAG_PORT = Disable) into the SPI Flash memory.
  To perform this procedure:
  a. Connect to the external SPI flash (for example, MX25L12833F) through the external programmer, and then erase all Flash content.

  Note: If HW-USBN-2B is not available, HW-USBN-2A can be also used.

Parallel Config Header (J6)	Lattice Programming Cable (HW-USBN-2B)
MOSI	TDI/SI
MISO	TDO/SO
VCCIO0	VCC
GND	GND
SPI_MCLK	TCK
FLASH_CS	ISPEN/PROG

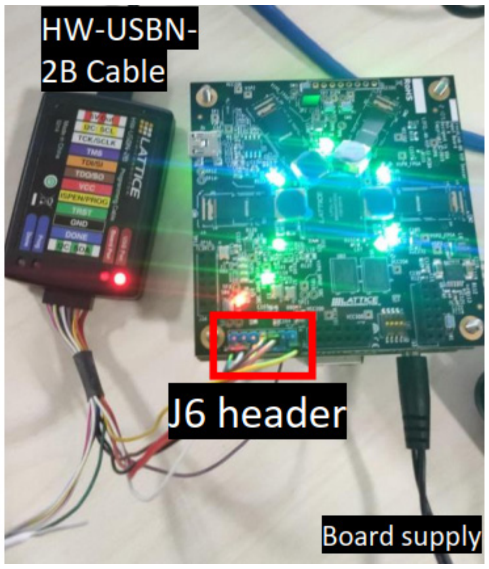
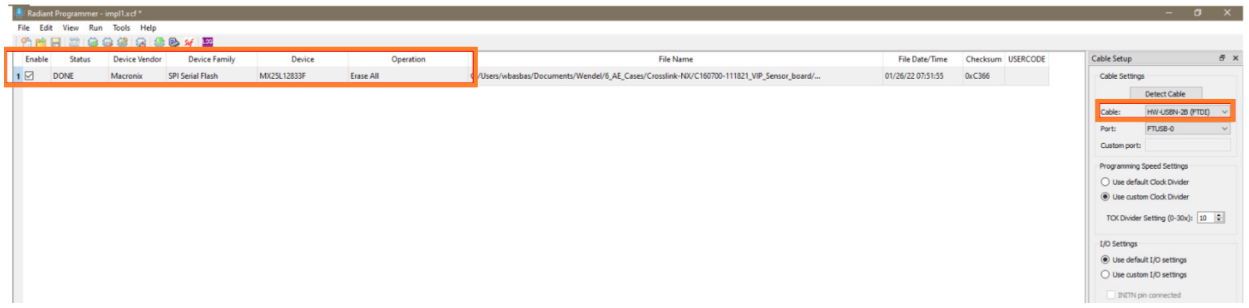


Figure 4.1. USBN-2B Hardware Connection to SPI Flash

  b. Set-up the programmer based on the SPI Serial Flash vendor, device name, and operation which is **Erase All**.


Radiant Programmer - impl1.xcf *
File Edit View Run Tools Help

	Enable	Status	Device Vendor	Device Family	Device	Operation	File Name	File Date/Time	Checksum	USERCODE
1	✓	DONE	Macronix	SPI Serial Flash	MX25L12833F	Erase All	C:/Users/wbasbas/Documents/Wendel/6_AE_Cases/Crosslink-NX/C160700-111821_VIP_Sensor_board/...	01/26/22 07:51:55	0xC366	

Cable Setup

Cable Settings

Detect Cable

Cable: HW-USBN-2B (FTDI)

Port: FTUSB-0

Custom port:

Programming Speed Settings

( ) Use default Clock Divider

(●) Use custom Clock Divider

TCK Divider Setting (0-30x): 10

I/O Settings

(●) Use default I/O settings

( ) Use custom I/O settings

☐ INITN pin connected

Figure 4.2. SPI Flash Erase All Operation

c. Perform *Erase All* on the SPI Flash Memory.

- Accessing the SRAM
  After removing bitstream from the SPI Flash with disabled JTAG_PORT, user can now perform SRAM Erase, Program and Verify.
  To perform this procedure:
  a. Remove the external programmer into the SPI Flash memory then connect the FPGA JTAG to the built-in FTDI or by using an external FTDI programmer.

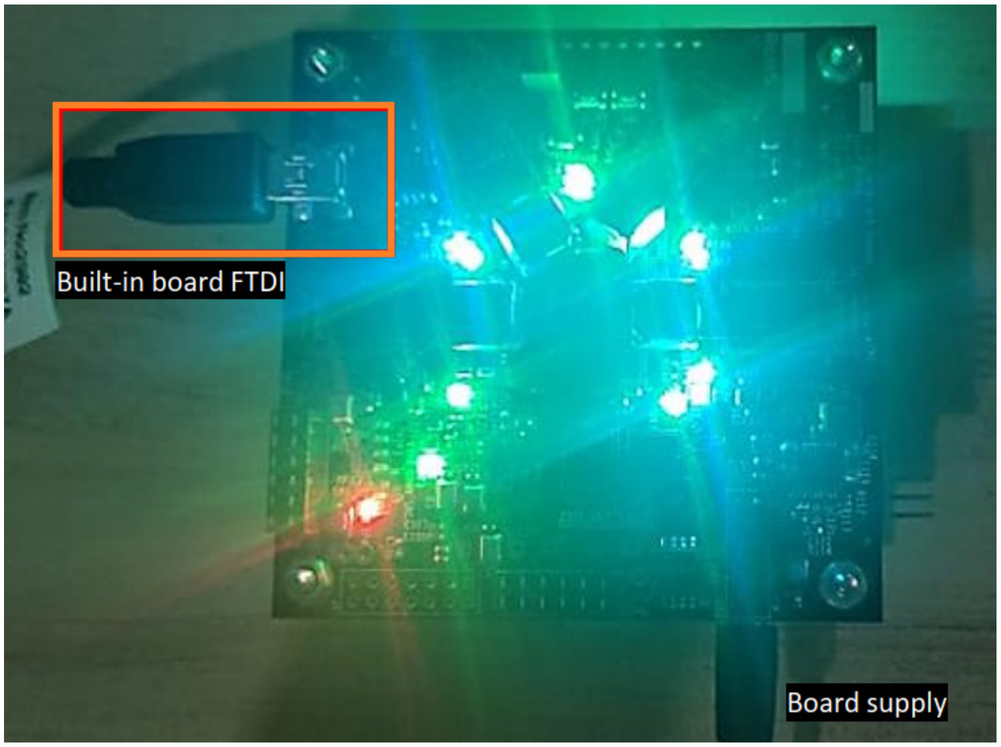
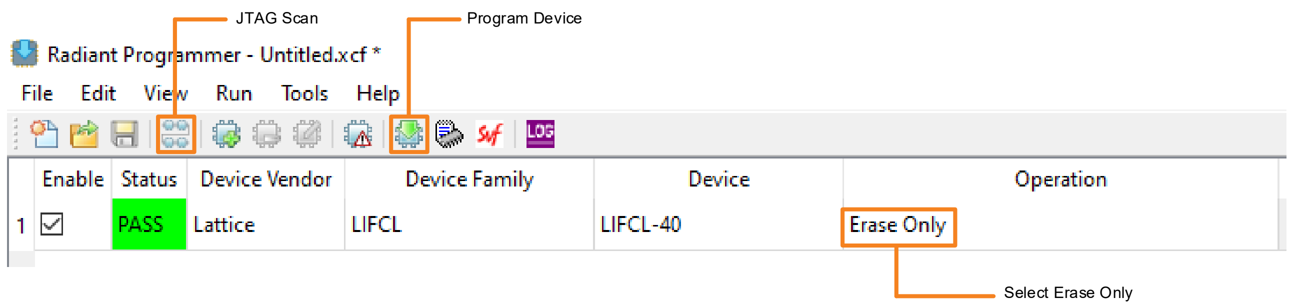


Figure 4.3. On-Board FTDI USB Port

b. Perform JTAG scan to read the device and the board.

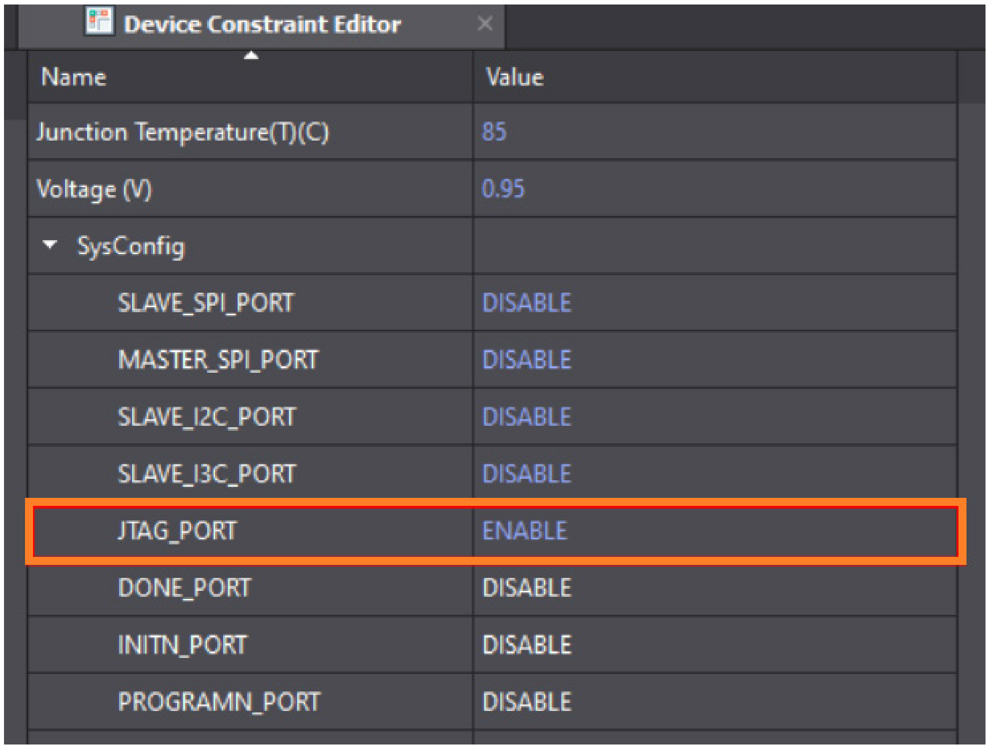
c. Set the desired SRAM operation and click **Program Device**.



	Enable	Status	Device Vendor	Device Family	Device	Operation
1	✓	PASS	Lattice	LIFCL	LIFCL-40	Erase Only

Figure 4.4. SRAM Erase Only Operation

d. If the new bitstream is loaded to the SPI flash, ensure that the JTAG_PORT is enabled this time. This can be set under **Device Constraint Editor**.



Name	Value
Junction Temperature(T)(C)	85
Voltage (V)	0.95
SysConfig	
SLAVE_SPI_PORT	DISABLE
MASTER_SPI_PORT	DISABLE
SLAVE_I2C_PORT	DISABLE
SLAVE_I3C_PORT	DISABLE
JTAG_PORT	ENABLE
DONE_PORT	DISABLE
INITN_PORT	DISABLE
PROGRAMN_PORT	DISABLE

Figure 4.5. JTAG_PORT Persistence

Q: How to solve Crosslink-NX evaluation board's FTDI chip not detected in Radiant Programmer and Device Manager?

A: Check the JP1 jumper setting of the Crosslink-NX evaluation board if it is 3.3 V when open since it is connected to 3.3 V through a 2.2 kΩ resistor. An active FTDI should have open connection. When it is shorted to ground, the FTDI is at reset state. JP1 jumper should be left open for it to remain operational as shown in Figure 4.6.

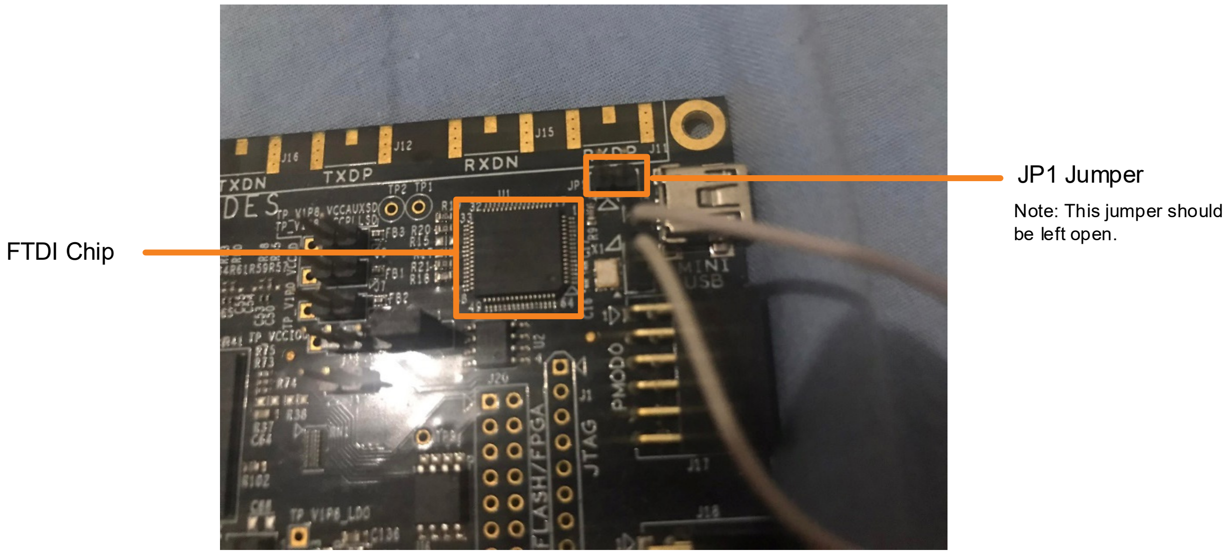
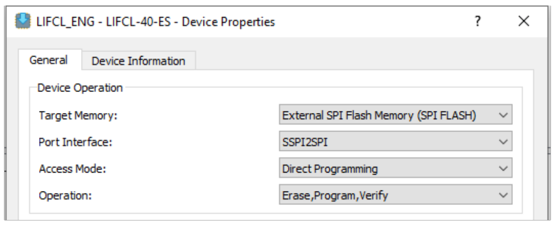


Figure 4.6. JP1 Header

Q: How to program external SPI flash through the SSPI2SPI port?

A: In the SSPI2SPI schematic as shown below, this indicates that the CrossLink-NX device is acting as a bridge to programming SPI Flash. To program the SPI directly, user can switch the Device Family to *SPI Serial Flash* and choose the SPI device.


LIFCL_ENG - LIFCL-40-ES - Device Properties

General | Device Information

Device Operation

Target Memory:	External SPI Flash Memory (SPI FLASH)
Port Interface:	SSPI2SPI
Access Mode:	Direct Programming
Operation:	Erase,Program,Verify

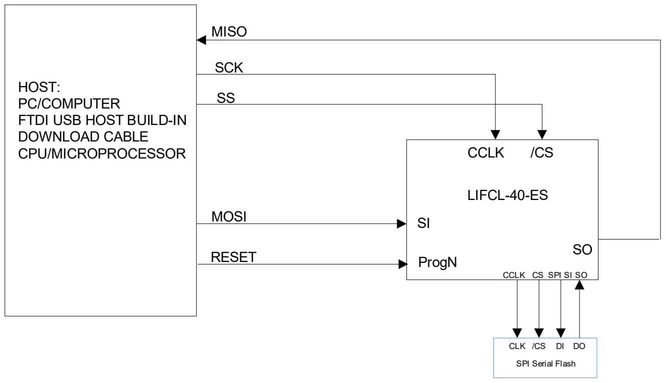


Figure 4.7. SPI Flash Programming through FPGA SSPI Port

Q: If the FPGA is not programmed, user can program the FPGA or the SPI flash without any problem. Once the FPGA is programmed, user can no longer download another image or program the flash.

A: This happens when user configures the SRAM with a bitstream with the MASTER_SPI_PORT set to DISABLED on the Device Constraints Editor. User needs to set MASTER_SPI_PORT and JTAG_PORT to ENABLED to make sure the JTAG-to-SSPI bridge remains functional. Thus, allowing user to download another image into the SPI Flash.

Q: CPOL of "Program Control NV Register 1" from "0" to "1". After that, even if the user changes the CPOL from "1" to "0", it remains "1" and does not change.

A: OTP bits cannot be changed once they are programmed.

Q: Programmed the SPI Flash with a bitstream with JTAG_PORT = DISABLED and MASTER_SPI_PORT = DISABLED causing the FPGA to be no longer configurable.

A: Provided that user did not set the PROGRAMN_DISABLE (Radiant Programmer) bit on the feature row, user can still perform hardware mode erase by performing the following steps:

1. If user has a PROGRAMN pin connected to a pushbutton (pb) switch, hold the PROGRAMN button to pull it ground.
2. Using JTAG, run an **Erase Only** operation on the SRAM using the programmer while pressing on the PROGRAMN pb switch.
3. Release the PROGRAMN once user can see the DONE pin asserting or if user has an LED connected to DONE release PROGRAMN once user can see the DONE pin LED turning on.

Q: User wants to program the configuration data from Host IC to SPI Flash though FPGA. Should the user write bitstream data from the address 0x00000000 in order from the first byte?

A: Yes, because the bitstream is transferred starting with the first byte of the data file then, starting with the MSB of the byte.

Q: User needs to use the JTAG interface pins for dedicated debugging purposes and use SPI programming after power-up but the JTAG and SSPI are sharing the same pins.

A: As stated in the Certus™-NX SPI Connections table in the Certus-NX Versa Evaluation Board (FPGA-EB-02032), user can use those pins for SSPI Configuration since the Certus-NX Versa Evaluation Board can support both Master SPI (MSPI) and Slave SPI (SSPI) modes for Certus-NX configuration. With that, user can use the JTAG interface dedicated only for JTAG.

Q: Does Lattice programming software and driver support FT4232HQ?

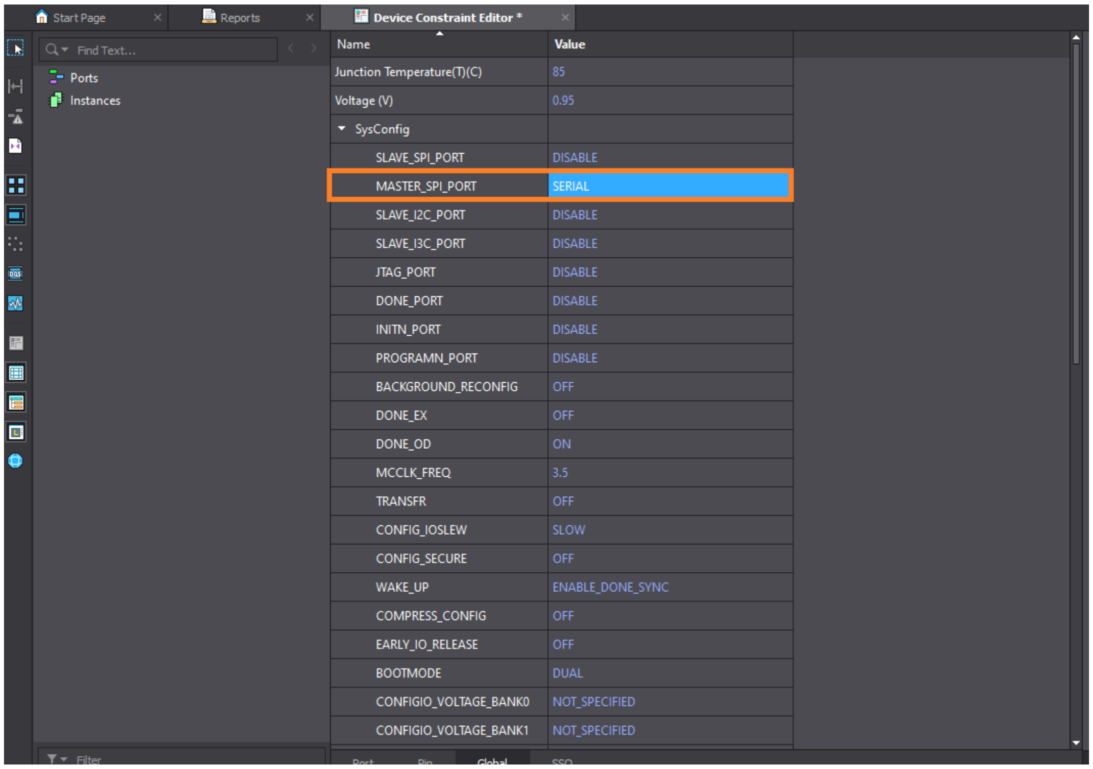
A: No current support for FT4232 and none planned for the foreseeable future.

Q: User programmed the SPI flash chip on the Evaluation board once and cannot do it again.

A: The probable reason user cannot reprogram the SPI Flash after one programming is that the configuration bitstream the user used on the design does not use the persistence of the MSPI pins, turning those pins into GPIO upon entering user mode.

To resolve this, user may erase the SRAM first (*SRAM Programming > Erase Only*) and then reprogram the SPI Flash. Do not power cycle in between.

To prevent this from happening again, user may set the persistence of the Master SPI port on the design using the *Device Constraint Editor*.



Name	Value
Junction Temperature(T)(C)	85
Voltage (V)	0.95
SysConfig	
SLAVE_SPI_PORT	DISABLE
MASTER_SPI_PORT	SERIAL
SLAVE_I2C_PORT	DISABLE
SLAVE_I3C_PORT	DISABLE
JTAG_PORT	DISABLE
DONE_PORT	DISABLE
INITN_PORT	DISABLE
PROGRAMN_PORT	DISABLE
BACKGROUND_RECONFIG	OFF
DONE_EX	OFF
DONE_OD	ON
MCCLK_FREQ	3.5
TRANSFR	OFF
CONFIG_IOSLEW	SLOW
CONFIG_SECURE	OFF
WAKE_UP	ENABLE_DONE_SYNC
COMPRESS_CONFIG	OFF
EARLY_IO_RELEASE	OFF
BOOTMODE	DUAL
CONFIGIO_VOLTAGE_BANK0	NOT_SPECIFIED
CONFIGIO_VOLTAGE_BANK1	NOT_SPECIFIED

Figure 4.8. MSPI_PORT Persistence

Q: Is it possible to program the device without having the MISO signal connected?

A: This is not recommended but it is possible. User needs to modify the programming flow such that only Class B, Class C, and Class D commands are used.

For SPI configuration flow, refer to Figure 4.9.

- Skip step 2 and step 3 (READ ID), as well as step 7 (check Status Register).
- Since the DONE and BUSY bit are not checked using step 7, the user needs to allocate ample delay before going to the next step. User can check the status through the DONE bit if it is connected to the Master.

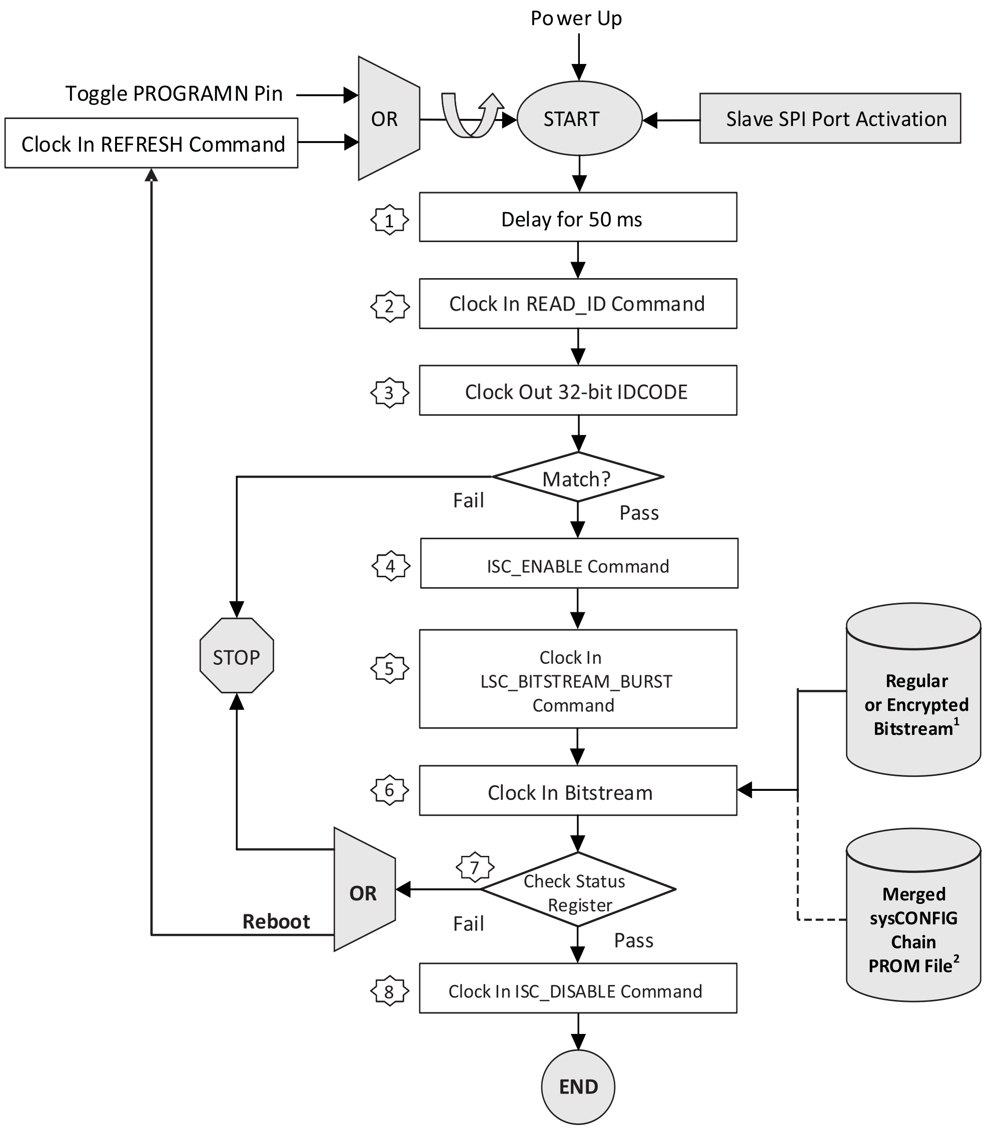


Figure 4.9. Slave SPI Configuration Flow

Q: Downloading the Programming Cable User Guide (FPGA-UG-02042) does not include any Nexus devices.

A: The following connections/signals are required to establish a successful JTAG connection to CertusPro™-NX:

- TCK/SCLK (White)
- TMS (Purple)
- TDI/SI (Orange)
- TDO/SO (Brown)
- VCC (Red) – This should be connected to VCCIO1 (Bank I/O of the JTAG pins of CertusPro-NX)
- GND (Black)

In addition, the JTAG_EN pin of the Certus-NX device should be high.

5. ID Check Read Error

5.1. Terminologies

5.1.1. JTAG_EN

Once the dedicated JTAG_EN pin is driven HIGH, it can enable the JTAG port at any time, no matter whether the device is in configuration mode or user functional mode. Pulling JTAG_EN to low at any time disables the JTAG interface.

5.1.2. CPHA and CPOL

Table 5.1. CPHA and CPOL Control Bits

Bit	Definition	Default	Note
[31:24]	Reserved	0	—
23	CPHA	0	This control bit is used to select SPI clock format. 0 – Sampling of data occurs at the rising edge of the clock. 1 – Sampling of data occurs at the falling edge of the clock.
22	CPOL	0	This control bit selects an inverted or non-inverted clock. 0 – Active high clocks selected. In idle state, clock is low. 1 – Active low clocks selected. In idle state, clock is high.

5.2. Frequently Asked Questions

Q: Flash ID Check fails and cannot detect flash in the CrossLink-NX ES device.

A: Perform the following steps to resolve this issue:

1. Select LIFCL_ENG as the **Device Family**.
2. Under Programming Speed Settings, select Use customer Clock Divider. Use at least 2 as the TCK Divider Setting.
3. Re-do **Detect Cable** and perform Flash Verify ID again.

   Note: The maximum supported Radiant Software version of the CrossLink-NX ES device is Radiant 2.0 SP1.

Q: There is a problem in reading the device ID through Slave SPI.

A: The main reason for the problem can be the high level of JTAG_EN. The JTAG_EN should be pulled low to enable the Slave SPI Access. When JTAG_EN is high, the JTAG interface is enabled, and this blocks the Slave SPI. This is because the JTAG and SSPI share the same port. Also, if the device is in user mode, the current running configuration should contain an ENABLED SLAVE_SPI_PORT in the Device Constraint Editor.

To pull the JTAG_EN down, user can add a switch to ground. The proper SPI mode (Mode 0, CPHA=0, CPOL=0).

6. Embedded-related

6.1. Terminologies

6.1.1. Serial Vector Format (SVF)

Serial Vector Format is a file format that includes a programming and configuration sequence or series of commands for a specific operation that can be used as a reference. This file can be generated by using either the Radiant Programmer or Deployment Tool.

6.2. Frequently Asked Questions

Q: Is there an example design or c source code for the FPGA configuration through MCU?

A: The C source code for embedded programming is provided in the software installation directory:
C:\lscc\radiant\3.0\programmer\embedded_source.

Q: How to flash FPGA binary file (.rbt) file and Propel SDK (RISC-V) binary file?

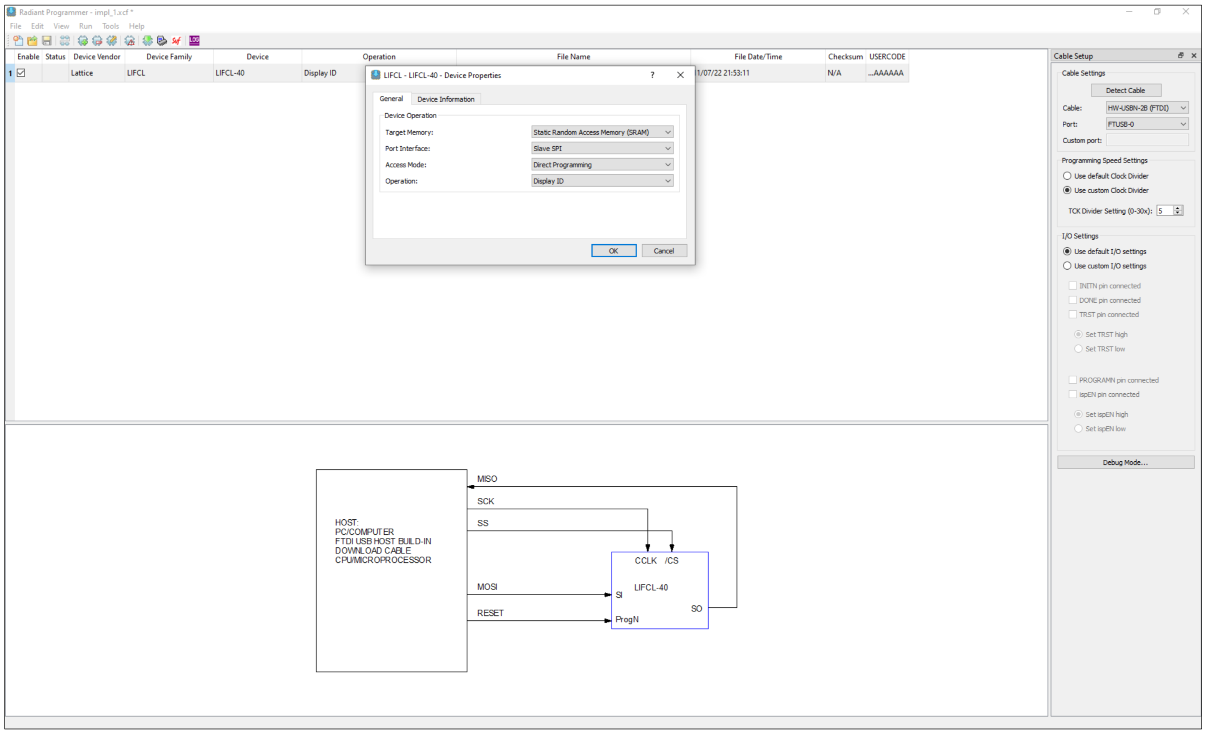
A: For *.rbt*, user can use the provided C source code in the installation directory for embedded programming.

As for the Propel binary file, it is not possible as of now to load it separately from the bitstream.

Q: How to create a SVF file in Radiant Programmer?

A: Perform the following steps to create the SVF file:

1. In Radiant Programmer, set up the operation as shown in Figure 6.1, which is an example of the Display ID operation using SSPI.



	Enable	Status	Device Vendor	Device Family	Device	Operation	File Name	File Date/Time	Checksum	USERCODE
1	✓		Lattice	LIFCL	LIFCL-40	Display ID		1/07/22 21:53:11	N/A	...AAAAAA

Device Operation
Target Memory: Static Random Access Memory (SRAM)
Port Interface: Slave SPI
Access Mode: Direct Programming
Operation: Display ID

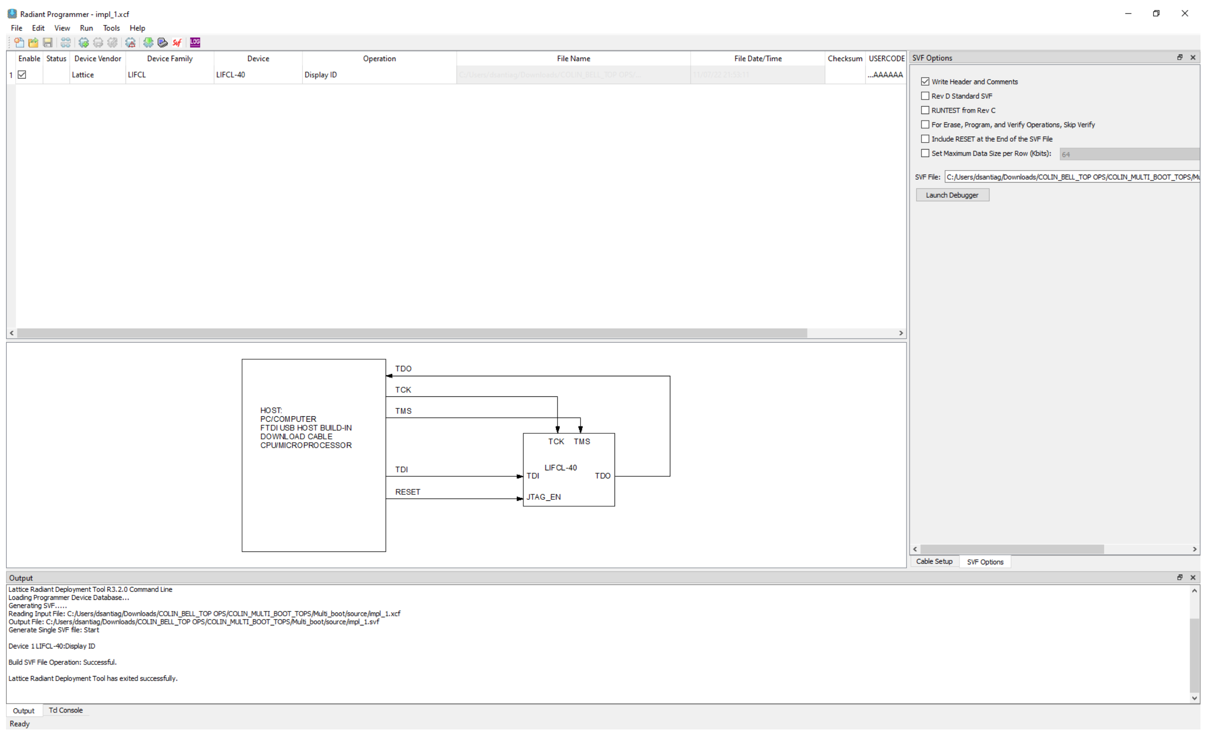
Cable Settings
Cable: HW-USBN-2B (FTDI)
Port: FTUSB-0
Custom port:

Programming Speed Settings
Use default Clock Divider [✓ Use custom Clock Divider]
TCK Divider Setting (0-30x): 5

I/O Settings
[✓ Use default I/O settings] Use custom I/O settings
INITN pin connected
DONE pin connected
TRST pin connected
Set TRST high
Set TRST low
PROGRAMN pin connected
ispEN pin connected
Set ispEN high
Set ispEN low

Figure 6.1. Radiant Programmer – Display ID Operation

2. Once the .xcf file is saved, click the **Generate SVF** icon. The Radiant Programmer should return a message that the SVF file is successfully built. After the SVF file is generated, click **Launch Debugger**.



	Enable	Status	Device Vendor	Device Family	Device	Operation	File Name	File Date/Time	Checksum	USERCODE
1	☑		Lattice	LIFCL	LIFCL-40	Display ID	C:/Users/dsantiag/Downloads/COLIN_BELL_TOP OPS/...	11/07/22 21:53:11		...AAAAAA

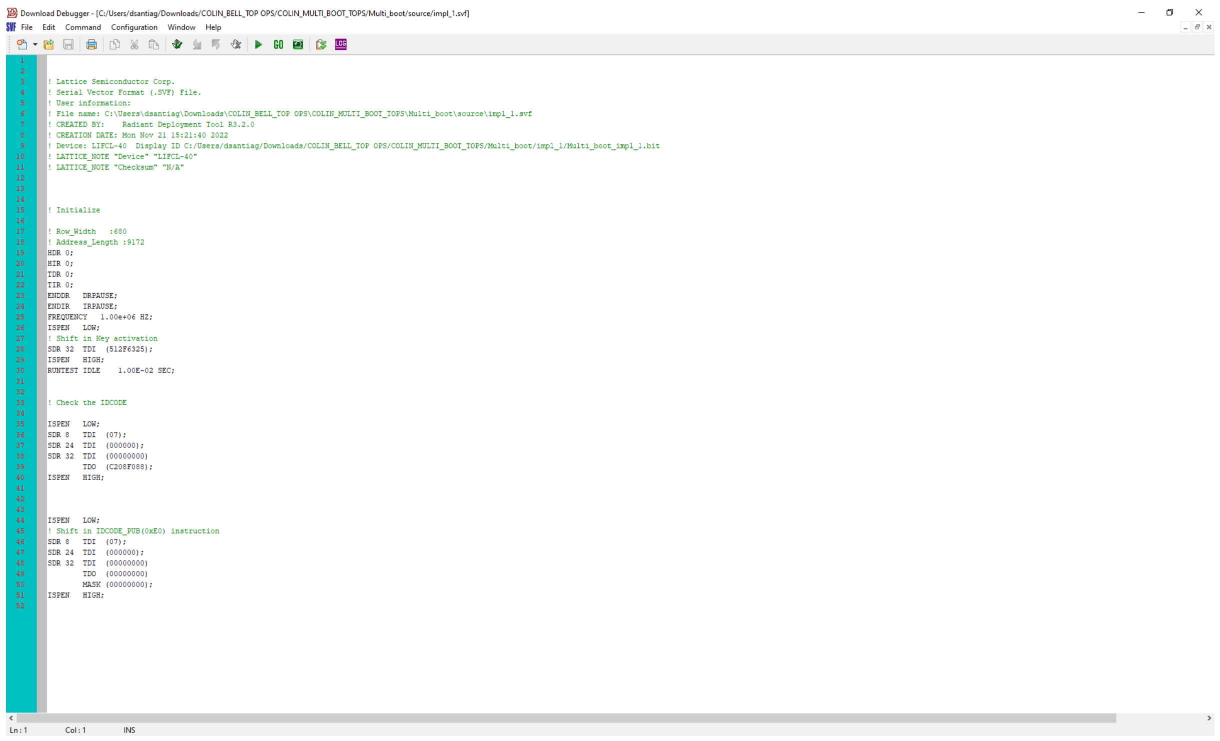
☑ Write Header and Comments
☐ Rev D Standard SVF
☐ RUNTEST from Rev C
☐ For Erase, Program, and Verify Operations, Skip Verify
☐ Include RESET at the End of the SVF File
☐ Set Maximum Data Size per Row (Kbits): 64

SVF File: C:/Users/dsantiag/Downloads/COLIN_BELL_TOP OPS/COLIN_MULTI_BOOT_TOPS/Mu

```
Lattice Radiant Deployment Tool R3.2.0 Command Line
Loading Programmer Device Database...
Generating SVF.....
Reading Input File: C:/Users/dsantiag/Downloads/COLIN_BELL_TOP OPS/COLIN_MULTI_BOOT_TOPS/Multi_boot/source/impl_1.xcf
Output File: C:/Users/dsantiag/Downloads/COLIN_BELL_TOP OPS/COLIN_MULTI_BOOT_TOPS/Multi_boot/source/impl_1.svf
Generate Single SVF file: Start

Device 1 LIFCL-40:Display ID

Build SVF File Operation: Successful.

Lattice Radiant Deployment Tool has exited successfully.
```

Figure 6.2. SVF File Generation

3. The SVF file generated provides the sequence needed to perform the saved operation in the .xcf file.


Download Debugger - [C:/Users/dsantiag/Downloads/COLIN_BELL_TOP OPS/COLIN_MULTI_BOOT_TOPS/Multi_boot/source/impl_1.svf]
File Edit Command Configuration Window Help
Ln : 1 Col : 1 INS

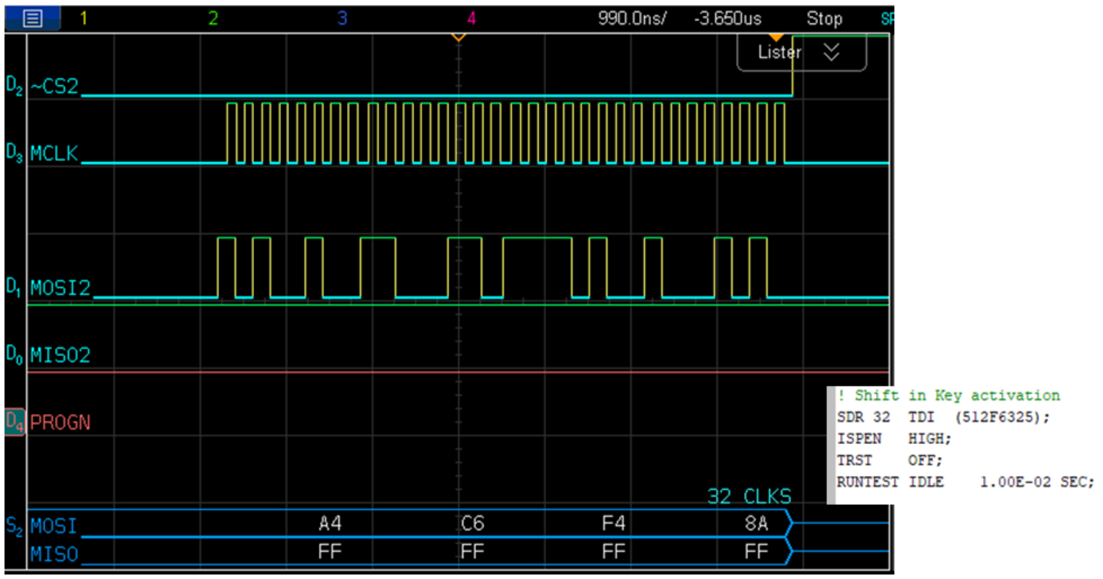
```


! Lattice Semiconductor Corp.
! Serial Vector Format (.SVF) File.
! User information:
! File name: C:\Users\dsantiag\Downloads\COLIN_BELL_TOP OPS\COLIN_MULTI_BOOT_TOPS\Multi_boot\source\impl_1.svf
! CREATED BY:    Radiant Deployment Tool R3.2.0
! CREATION DATE: Mon Nov 21 15:21:40 2022
! Device: LIFCL-40  Display ID C:/Users/dsantiag/Downloads/COLIN_BELL_TOP OPS/COLIN_MULTI_BOOT_TOPS/Multi_boot/impl_1/Multi_boot_impl_1.bit
! LATTICE_NOTE "Device" "LIFCL-40"
! LATTICE_NOTE "Checksum" "N/A"



! Initialize

! Row_Width    :680
! Address_Length :9172
HDR 0;
HIR 0;
TDR 0;
TIR 0;
ENDDR   DRPAUSE;
ENDIR   IRPAUSE;
FREQUENCY   1.00e+06 HZ;
ISPEN   LOW;
! Shift in Key activation
SDR 32  TDI  (512F6325);
ISPEN   HIGH;
RUNTEST IDLE    1.00E-02 SEC;


! Check the IDCODE

ISPEN   LOW;
SDR 8   TDI  (07);
SDR 24  TDI  (000000);
SDR 32  TDI  (00000000)
        TDO  (C208F088);
ISPEN   HIGH;



ISPEN   LOW;
! Shift in IDCODE_PUB(0xE0) instruction
SDR 8   TDI  (07);
SDR 24  TDI  (000000);
SDR 32  TDI  (00000000)
        TDO  (00000000)
        MASK (00000000);
ISPEN   HIGH;

```

Figure 6.3. SVF File for Display ID

4. The sequence provided in the SVF file is what the Radiant Programmer does in the operation the user saves in the .xcf file. Figure 6.4 and Figure 6.5 show the waveforms that correlates with the sequence.

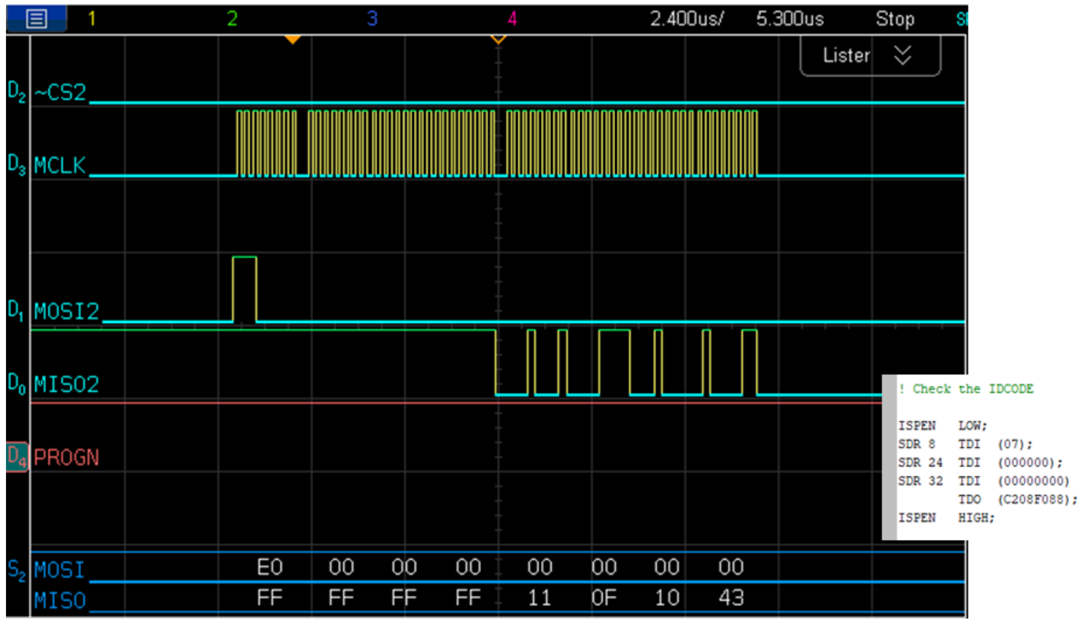
Notes:

- The SVF file generation supports all Nexus Devices.
- The SVF file “Step” and “Go” processes in Download Debugger are supported in JTAG Mode only.
- The SVF file “Step” and “Go” processes in Download Debugger Slave operations SSPI and I^{2}C are not supported, these are only meant to generate a sequence that can serve as a guide needed for configuration and programming in case user wants to create its own.
- User can generate an SVF file using any operation that can be found in the Radiant Programmer such as Fast Configuration or Erase, Program, Verify.
- User can also generate an SVF file for external SPI Flash Programming if it’s supported by the Radiant Programmer.



```
! Shift in Key activation
SDR 32  TDI  (512F6325);
ISPEN   HIGH;
TRST    OFF;
RUNTEST IDLE    1.00E-02 SEC;
```

Figure 6.4. Shift in Activation Key



```
! Check the IDCODE

ISPEN   LOW;
SDR 8   TDI  (07);
SDR 24  TDI  (000000);
SDR 32  TDI  (00000000)
        TDO  (C208F088);
ISPEN   HIGH;
```

Figure 6.5. READ_ID Command and Device IDCODE Readout

7. Encryption Keys

7.1. Frequently Asked Questions

Q: Can encryption key be reprogrammed?

A: AES keys are OTP.

Q: How to improve successful programming of key through JTAG?

A: Low noise JTAG interface and higher TCK.

Appendix A. Other Common Issues Affecting Configuration and Programming

Q: Bitstream support when using Radiant Software across Crosslink-NX devices.

A: For ES2 devices, it is recommended to use Radiant 2.2 and later. For ES devices, the Radiant compatibility is limited to Radiant 2.0 SP1.

To install Radiant 2.0 SP1, perform the following steps:

- Download and install Radiant 2.0 using this link:
  https://www.latticesemi.com/FileExplorer?media={A7767FCC-BA13-42EF-BA64-7809197E680E}&document_id=52797
- Download and install the SP1 using this link:
  https://www.latticesemi.com/FileExplorer?media={4516DF15-7F55-4BB4-A841-4E39FE639C7F}&document_id=52877

Table A.1 lists the Radiant software version support across CrossLink-NX devices and highlights the flow support for bitstream generation. For example, the CrossLink-NX 40 ES bitstreams can only be generated using Radiant 2.0 SP1 release. The Radiant software flow support for the silicon revisions are summarized in Table A.1.

Table A.1. Lattice Radiant Software Flow Support for Nexus Family Silicon Revisions

	Radiant Support		
	Radiant 2.0 SP1	**Radiant 2.2**	**Radiant 2.2 SP1**
CrossLink-NX 40 ES	Yes	No	No
CrossLink-NX 40 ES2	No	Yes	Yes
CrossLink-NX 17 ES	No	Yes	Yes
CrossLink-NX 40 PSR	No	Yes[1]	Yes
CrossLink-NX 17 PSR	No	Yes[1]	Yes
Certus-NX 40 PSR	No	Yes[1]	Yes
Certus-NX 17 PSR	No	Yes[1]	Yes

Note:

1. The timing data is not final. Final PSR timing data is available in Radiant 2.2 SP1.

Q: Deployment Tool crashes when trying to save Deployment Tool project when using Radiant 2.2.

A: This is a bug and already fixed in Radiant 3.0.

Q: How to solve Radiant 2.2 error in programming ES?

A: When using Radiant 2.2 Programmer, use "LIFCL_ENG" as the device family instead of "LIFCL". This solves any programming issue. Also, user can vary the TCK divider to slow down the programming speed.

Q: Cannot run Radiant Programmer from the command line.

A: Read the *Setting Up the Environment to Run Command Line* in the Help file and run the following commands:

```
SET FOUNDRY=C:\lscc\radiant\3.0\ispfpga;
SET PATH=C:\lscc\radiant\3.0\bin\nt64;C:\lscc\radiant\3.0\ispfpga\bin\nt64;C:\lscc\
radiant\3.0\programmer\bin\nt64;%PATH%;
```

Q: FPGA fails to complete boot-up process using CLNX-17. It is observed that the *Data file size* in the Radiant programmer 3.0 – with access mode SPI flash, JTAG2SPI, Direct programming, Erase, Program, Verify – not being read from the file.

A: The data file size is auto-update after reproducing, no hardware required. There are times that the data file size might not be automatically updated but this can be addressed by clicking *load from file* to refresh. They are intermittent and occurrence is not usually consistent.

Q: Unable to program public key into the device (Radiant 2.2).

A: This issue is already fixed in Radiant 3.0.

Q: When using the Radiant software to generate a bitstream, the command (-secure on) has no effect.

A: This is only a text bug and already fixed in the later versions.

Q: Encrypting a compressed bitstream makes the encrypted bitstream much bigger.

A: It is not recommended to use compression with encryption as there is no guarantee that the bitstream is correctly configured when decrypted and uncompressed. Refer to page 20 of Advanced Configuration Security Usage Guide for Nexus Platform for more information (FPGA-TN-02176).

Q: How to map the DPHY primitive to one of the available Hardened D-PHY blocks on CrossLink-NX device.

A: To map the DPHY primitive using the ldc_set_location preference, refer to the example below:

PREFERENCE: ldc_set_location –site Pin_Number [get_ports Port_Name]

Example Constraint for LIFCL-17 csfBGA121 MIPI DPHY0:

ldc_set_location –site {C11} [get_ports clk_p_i]

ldc_set_location –site {C10} [get_ports clk_n_i]

ldc_set_location –site {D11} [get_ports d0_p_i]

ldc_set_location –site {D10} [get_ports d0_n_i]

For more information regarding the sysCONFIG items, refer to sysCONFIG User Guide for Nexus Platform (FPGA-TN-02099).

Appendix B. Configuration Ports Basic Concepts

The following are the two basic states that take place in the FPGA:

- Configuration mode – When the FPGA wakes up after power-up, user should configure it since it is in configuration mode with all its outputs inactive. By default, if there is no running bitstream in the FPGA, the Master SPI port is enabled. To select on which port the user wants to use between JTAG, SSPI, or I^{2}C, the level of JTAG_EN should be considered. JTAG_EN should be pulled high for JTAG and should be pulled low for SSPI and I^{2}C.
- User mode – Configuring the FPGA means downloading a stream of 0s and 1s through the SysConfig pins. Once the FPGA is properly configured, it goes into *user-mode* and becomes active. The persistence settings in the Device Constraint Editor such as MASTER_SPI_PORT, JTAG_PORT and SLAVE_SPI_PORT take effect in user mode.

The commonly used sysCONFIG ports overview is shown in Figure B.1.

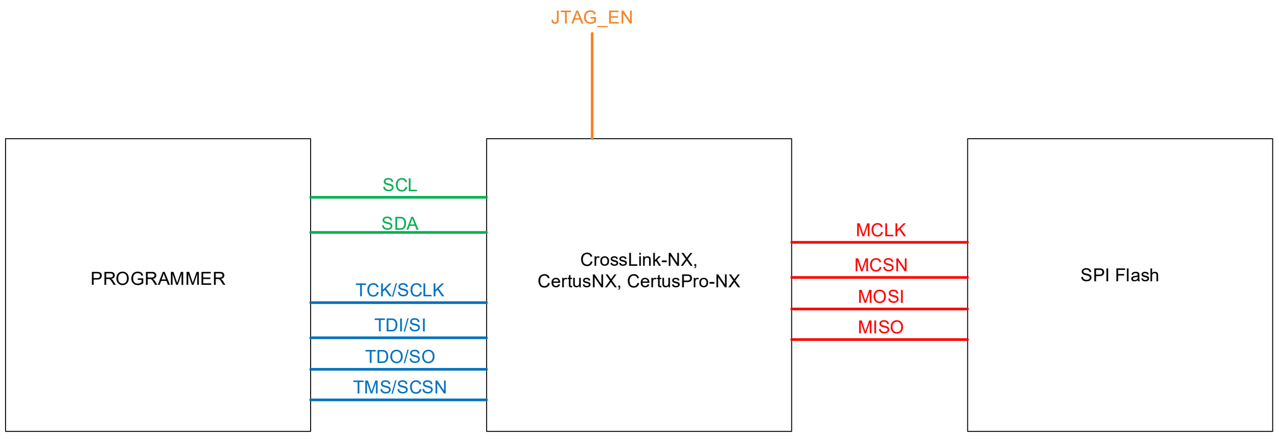


Figure B.1. sysCONFIG Port Top-Level Diagram

Figure B.1 shows the top-level diagram of the sysCONFIG ports:

- Green – the Slave I^{2}C port wherein the programmer acts as a Master and the FPGA as a Slave.
- Blue – the JTAG port and Slave SPI port wherein the programmer acts as a Master and the FPGA as a Slave. Note that the JTAG and Slave SPI share the same pins.
- Red – the Master SPI port wherein the FPGA acts as a Master and the external SPI Flash as a Slave.

The Crosslink-NX, Certus-NX and CertusPro-NX devices support JTAG2SPI and SSPI2MSPI bridging.

When the device is in configuration mode, the following conditions should be satisfied:

- To enable the JTAG2SPI bridging, the level of JTAG_EN should be pulled to high.
- To enable the SSPI2MSPI bridging, the level of JTAG_EN should be pulled to low.

When the device is in user mode, the following conditions should be satisfied:

- To enable the JTAG2SPI bridging, the level of JTAG_EN should be pulled to high and the JTAG_PORT and MASTER_SPI_PORT setting in the Device Constraint Editor should be enabled.

MASTER_SPI_PORT	SERIAL
JTAG_PORT	ENABLE

- To enable the SSPI2MSPI bridging, the level of JTAG_EN should be pulled to low and the SLAVE_SPI_PORT and MASTER_SPI_PORT setting in the Device Constraint Editor should be enabled.

SLAVE_SPI_PORT	SERIAL
MASTER_SPI_PORT	SERIAL

If the user encounters any configuration port access issues, refer to the top-level debugging flow as shown in Figure B.2. For easy debugging, the user should test this flow using the SRAM Display ID and SPI Flash Display ID Operations first in the Radiant Programmer.

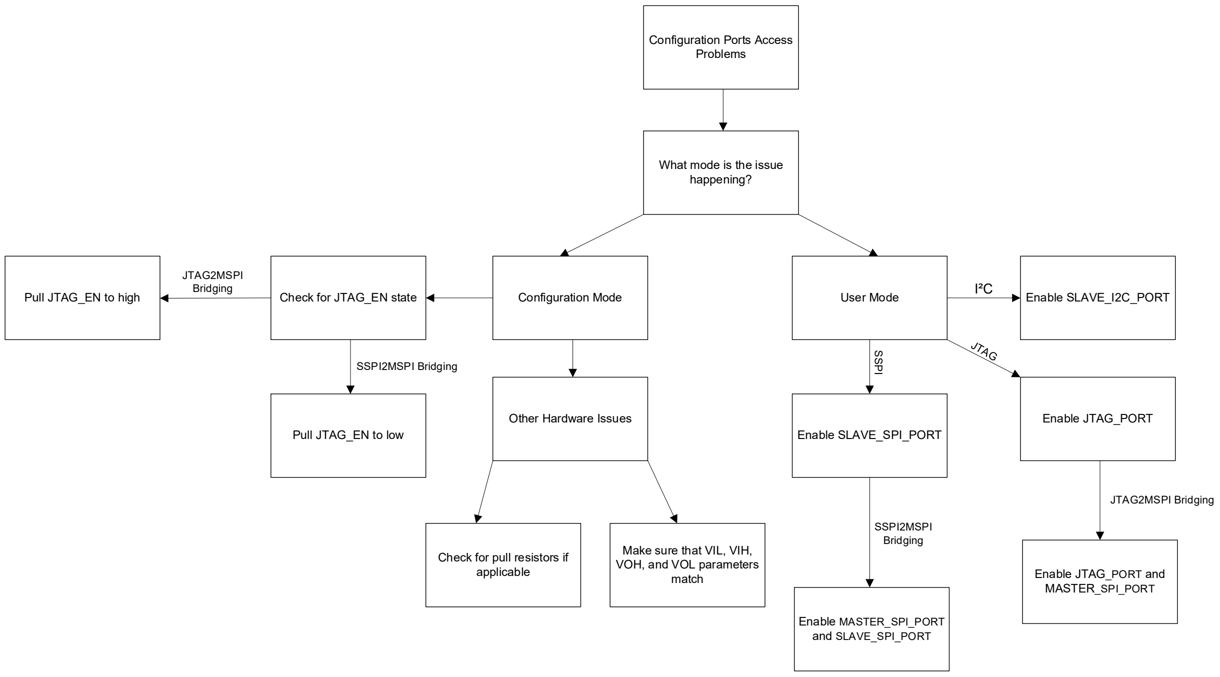


Figure B.2. Top-level Debugging Flow Diagram

References

For more information refer to:

- CrossLink-NX Family Data Sheet (FPGA-DS-02049)
- Certus-NX Family Data Sheet (FPGA-DS-02078)
- CertusPro-NX Family Data Sheet (FPGA-DS-02086)
- Certus-NX Versa Evaluation Board (FPGA-EB-02032)
- Programming Cable User Guide (FPGA-UG-02042)
- sysCONFIG User Guide for Nexus Platform (FPGA-TN-02099)
- 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 www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.2, July 2023

Section	Change Summary
Introduction	Rephrased the purpose of the document.
Boot Process	Updated Section 3.2 FAQ.
	Updated Figure 3.3. Feature Rows Programming Operation and Figure 3.4. Boot Select Bits
Appendix B. Configuration Ports Basic Concepts	Updated Figure B.2. Top-level Debugging Flow Diagram
References	Added this section.

Revision 1.1, January 2023

Section	Change Summary
All	• Minor adjustments in formatting across the document. • Changed document name from Common Issues Related to Configuration and Programming for Nexus Platform to *Common Programming and Configuration FAQs with Supplementary Concepts for CrossLink-NX, Certus-NX, and CertusPro-NX*.
Acronyms in This Document	Added definition for SVF.
PROGRAMN/INITN/DONE as GPIO	• Updated note in the Feature-Row bits and settings question. • Updated Figure 2.3. Lattice Radiant Software Global Settings and Figure 2.4. Generated .fea File.
ID Check Read Error	Updated Flash ID check question to add CrossLink-NX device, added note, and updated Slave SPI device reading question.
Embedded-related	Added FAQ on creating SVF file through Radiant Programmer.
Appendix B. Configuration Ports Basic Concepts	Added this section.
Technical Support Assistance	Added reference link to Lattice Answer Database.

Revision 1.0, April 2022

Section	Change Summary
All	Initial release

www.latticesemi.com