Mach-NX Programming and Configuration Usage Guide

Technical Note

FPGA-TN-02231-1.0

April 2021

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

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AES	Advanced Encryption Standard
AHB	A bus protocol introduced in AMBA version 2 published by ARM Ltd company.
AHB-Lite	A subset of AHB formally defined in the AMBA 3 standard.
AMBA	Advanced Microcontroller Bus Architecture
CCLK	Configuration Clock
CFG	Configuration Flash Memory
CRC	Cyclic Redundancy Check
DSA	Digital Signature Algorithm
EBR	Embedded Block RAM
EFB	Embedded Function Block
ECDSA	Elliptic Curve Digital Signature Algorithm
ESB	Embedded Security Block
FAM	Flash Address Memory
GOE	Global Output Enable
GSR	Global Set Reset
GWDIS	Global Write Disable
I^2C	Inter-Integrated Circuit
JTAG	Joint Test Action Group
LUT	Look Up Table
PFR	Platform Firmware Resiliency
POR	Power On Reset
PROM	Programmable Read-Only Memory
RAM	Random Access Memory
SDM	Self Download Mode
SFB	SoC Function Block
SoC	System on Chip
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
SSPI	Slave Serial Peripheral Interface
SVF	Serial Vector Format
TAP	Test Access Port
TCK	Test Clock
TDI	Test Data Input
TDO	Test Data Output
TMS	Test Mode Select
UFM	User Flash Memory

1. Introduction

The Mach™-NX device is an SRAM-based Programmable Logic Device (PLD) that includes an internal Flash memory, which allows it to operate similar to a non-volatile device. Mach-NX device provides a rich set of features for the programming and configuration of the FPGA. One key feature of the Mach-NX device is the embedded security capability. An embedded security block provides the authentication and encryption of the bitstream to prevent malicious attacks. Mach-NX device also has flexible and robust access control for the configuration ports to enable the various programming and update needs. Mach-NX device provides many user options to program and configure the device in order to address every customer's needs. Each of the available options is described in detail so that you can put together the programming and configuration solution that meet your needs.

The Mach-NX device contains two types of memory, SRAM and Flash. SRAM memory contains the active configuration, essentially the fuses that define the behavior of the FPGA. The active configuration is, in most cases, retrieved from a non-volatile memory. The non-volatile memory holds the configuration data that is loaded into the FPGA's SRAM. The Mach-NX device provides an internal Flash memory that can be used to store the configuration data loaded into the Mach-NX SRAM.

2. Mach-NX Devices Features

The key programming and configuration features of Mach-NX devices are:

- Instant-on configuration from internal Flash – powers up in milliseconds
- Up to 10,000 programming cycles for the internal configuration flash memory
- Single-chip, secure solution
- Bitstream authentication using ECDSA256
- Optional bitstream encryption using AES256
- Multiple hard and soft lock controls for the Flash/SRAM access from configuration port or internal logic
- Multiple FPGA programming and configuration interfaces:
  - 1149.1 JTAG
  - Self-download
  - Slave SPI
  - Dual Boot
  - I^2C
  - AHB-Lite
- User Flash Memory (UFM) for non-volatile data storage:
  - EBR initialization data
  - Application specific data
- Transparent programming of non-volatile memory
- On-chip dual boot
- TransFR capability
  - Leave-alone I/O (non-JTAG mode)
- Security Platform SoC Function Block support
- Master SPI port for SoC Function Block configuration data and SoC Function Block firmware.

3. Definition of Terms

This document uses the following terms to describe common functions:

- AES – Advanced Encryption Standard is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.
- BIT – The BIT file is the configuration data for Mach-NX device that is stored in an external SPI Flash. It is a binary file and is programmed unmodified into the SPI Flash.
- Configuration – Configuration refers to a change in the state of the Mach-NX SRAM memory cells.
- Configuration data – This configuration data is the data read from the non-volatile memory and loaded into the FPGA SRAM configuration memory. This is also referred to as a bitstream, or device bitstream.
- Configuration mode – The configuration mode defines the method the Mach-NX device uses to acquire the configuration data from the non-volatile memory.
- Digest – A message digest is a cryptographic hash function containing a string of digits created by a one-way hashing formula.
- ECDSA – Elliptic Curve Digital Signature Algorithm is the elliptic curve analogue of the DSA.
- Internal flash memory – On-die, non-volatile flash-type memory. Mach-NX device contains multiple flash sectors for FPGA configuration image storage (up to 2), general purpose user flash (up to 3 sectors), and device feature definitions.
- JEDEC – The JEDEC file contains the configuration data programmed into the Mach-NX Configuration Flash. Format information is provided later in this technical note.
- Number formats – The following nomenclature is used to denote the radix of numbers:
  - 0x: Numbers preceded by ‘0x’ are hexadecimal.
  - b (suffix): Numbers suffixed with ‘b’ are binary.
  - All other numbers are decimal.
- Offline mode – Offline mode is a term that is applied to both non-volatile memory programming and SRAM configuration. When using offline mode programming/configuration, the FPGA no longer operates in user mode. The contents of the non-volatile or SRAM configuration memory are updated, but the Mach-NX device does not perform your logic operations until offline mode programming/configuration is complete.
- Port – A port refers to the physical connection used to perform programming and some configuration operations. Ports on the Mach-NX devices include JTAG, SPI, and I^2C physical connections.
- Programming – Programming refers to the process used to alter the contents of the internal or external non-volatile configuration memory.
- Signature – A digital signature guarantees the authenticity of an electronic document or message in digital communication and uses authentication techniques to provide proof of original and unmodified documentation.
- Transparent mode – Transparent mode is used to update the Configuration Flash and the User Flash Memory while leaving the Mach-NX devices in user mode.
- User mode – The Mach-NX device is in user mode when configuration is complete, and the FPGA is performing the logic functions it is programmed to perform.

4. Configuration Process and Flow

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

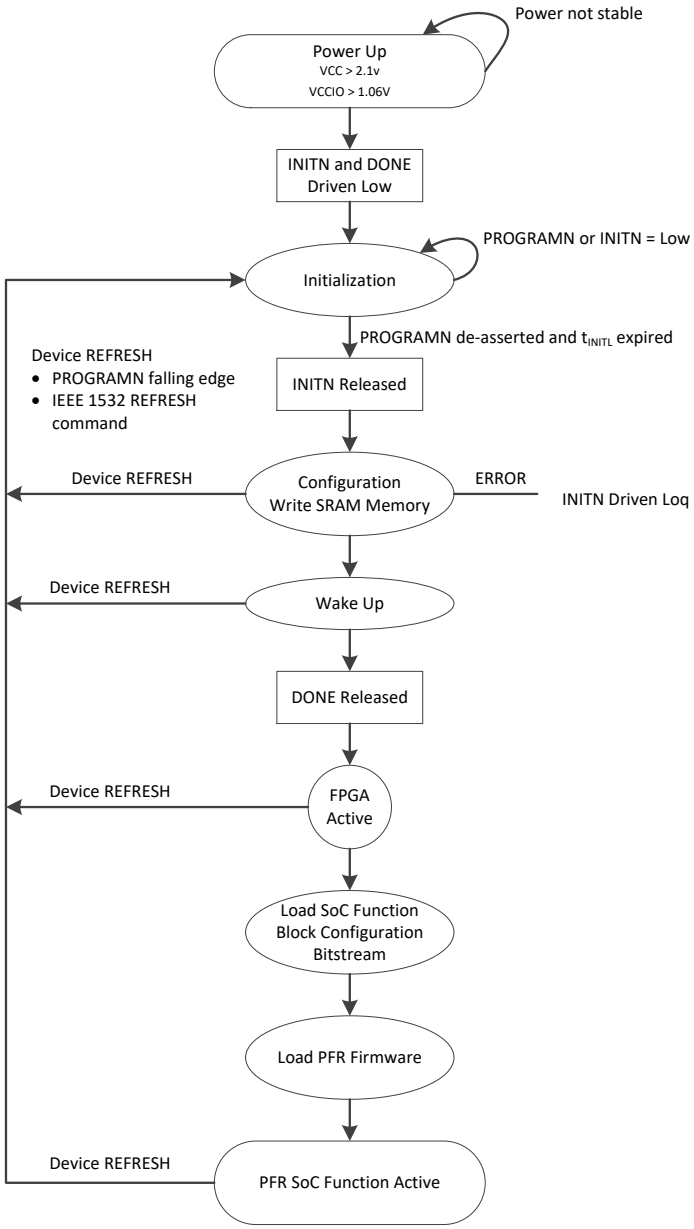


Figure 4.1. Configuration Flow

The Mach-NX sysCONFIG™ ports provide industry standard communication protocols for programming and configuring the FPGA. Each of the protocols shown in Table 4.1 provides a way to access the Mach-NX device internal Flash, or to load its configuration SRAM. The Memory Space Accessibility section provides information about the capabilities of each sysCONFIG port.

The sysCONFIG ports capable of accessing the SRAM have a priority order. The operation of the configuration logic is not defined when a low priority sysCONFIG port is interrupted by a higher priority sysCONFIG port. Do not permit simultaneous access to the configuration logic using a sysCONFIG port.

4.1. Power-up Sequence

In order for the Mach-NX 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 state becomes indeterminate.

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 VCC and VCCIO0 input rails. The POR circuit waits for the following conditions:

- VCC > 2.1 V
- VCCIO0 > 1.06 V

When these conditions are met, the POR circuit releases an internal reset strobe, allowing the device to begin its initialization process. The Mach-NX 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 4.2.

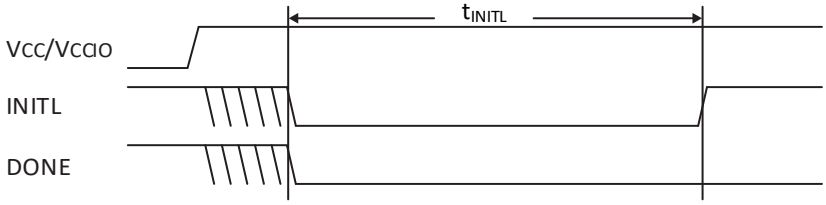


Figure 4.2. Configuration from Power-On-Reset Timing

4.2. Initialization

The Mach-NX device enters the memory initialization phase immediately after the POR circuit drives the INITN and DONE status pins low. The purpose of the initialization state is to clear all of the SRAM memory inside the FPGA.

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

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

The 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 Mach-NX device is part of a chain of devices, each device carries different t_{INITL} initialization times. 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 one or more chained devices to fail to configure intermittently.

4.3. Configuration

The rising edge of the INITN pin causes the FPGA to enter the configuration state. The FPGA is able to accept the configuration bitstream created by the Lattice Diamond® development tools.

The Mach-NX device begins fetching configuration data from non-volatile memory. Based on the user option, the data can be decrypted and/or authenticated. The memory used to configure the Mach-NX device is the internal memory. The Mach-NX device does not leave the Configuration state if there are no memories with valid configuration data or the data fails to pass the authentication. It is necessary to program the non-volatile memory internal or attached to the FPGA, or to program it using the JTAG port with correct data.

During the time the FPGA receives its configuration data, the INITN control pin takes on its final function. INITN is used to indicate that an error exists in the configuration data. When INITN is high, configuration proceeds without issue. If INITN is asserted low, an error occurred and the FPGA fails to operate.

4.4. Wake-up

Wake-up is the transition from configuration mode to user mode. The Mach-NX device fixed four-phase wake-up sequence starts when the device correctly receives all of its configuration data. 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:

- External DONE
- Global Write Disable (GWDISn)
- Global Output Enable (GOE)
- Global Set/Reset (GSR)

The first phase of the wake-up process is for the Mach-NX device to release the Global Output Enable. When it is asserted, the FPGA I/O is permitted to exit a high-impedance state and take on their 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 (GSR).

The second phase of the wake-up process releases the Global Set/Reset and the Global Write Disable 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 with the GSR enabled attribute to be set/cleared per their hardware description language definition.

The Global Write Disable is a control that overrides the write enable strobe for all RAM logic inside the FPGA. The inputs on the FPGA are always active. Keeping GWDIS asserted prevents accidental corruption of the instantiated RAM resources inside the FPGA.

The last phase of the wake-up process is to assert the external DONE pin. The external DONE is a bidirectional, opendrain I/O only when it is enabled. An external agent that holds the external DONE pin low prevents the wake-up process of the Mach-NX device 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.

4.5. FPGA Active

The FPGA fabric of the Mach-NX device begins performing the logic operations user designed. All non-SoC GPIO start to active. The FPGA fabric of Mach-NX device remains in this state until one of three events occurs:

- The PROGRAMN input pin is asserted
- A REFRESH command is received via one of the configuration ports
- Power is cycled

4.6. Load SoC Function Block Configuration Bitstream

Load the configuration bitstream for the SoC Function Block (SFB), which is manufacture supplied, signed and encrypted, from external SPI flash. The bitstream is decrypted and authenticated to ensure the authenticity and integrity of the SoC functionality.

4.7. Load PFR Firmware

Load the PFR run time firmware, which is manufacture supplied and signed, from external SPI flash. The PFR firmware is authenticated to ensure the authenticity and integrity of the SoC functionality.

4.8. PFR Soc Function Active

The Mach-NX device enters full PFR function state. All SoC I/O start to active. The Mach-NX device remains in this state until one of three events occurs:

- The PROGRAMN input pin is asserted
- A REFRESH command is received via one of the configuration ports
- Power is cycled

4.9. Clearing the Configuration Memory and Re-initialization

The current user mode configuration of the Mach-NX device remains in operation until it is actively cleared, or power is lost. Several methods are available to clear the internal configuration memory of the Mach-NX device:

- Cycle power to the Mach-NX device
- Assert the PROGRAMN input
- Issue the REFRESH command using a configuration port

Any active configuration port can be used to send a REFRESH command. Invoking one of these methods causes the Mach-NX device to drive INITN and DONE low. The Mach-NX device enters the initialization state as described earlier.

4.10. Memory Space Accessibility

The internal Flash memories and SRAM of the Mach-NX device can be read and written. Each port on the Mach-NX device has a different level of access to each memory space. Table 4.1 provides a cross-reference of the Mach-NX device ports and the memory space they can access.

As shown in Table 4.1, the JTAG port can read and write both of the internal memory spaces. No other port can read the SRAM configuration memory. The JTAG port can access the two memory spaces in Offline or Transparent mode. Every other port has limitations on the functions that can be performed.

Table 4.1. Memory Space Accessibility of Different Ports

Port	On-Chip Flash		SRAM	
	Read	Write	Read	Write
JTAG	Yes	Yes	Yes	Yes
SPI Port	Yes	Yes	No	Yes
I^2C Port	Yes	Yes	No	Yes
Internal AHB-Lite	Yes*	Yes*	No	No

*__Note__: In Transparent mode only.

When the lock policy is implemented, it further controls the accessibility. Refer to the Lock Bits and Lock Control Policy section for more details.

4.11. On-chip Flash Programming

On-chip Flash is programmed with different programming modes. These programming modes are discussed in the following sections. Within the different programming modes, there are two methods of programming the on-chip Flash: Offline and Background programming.

4.11.1. Offline Programming

Offline Programming requires the device to enter into programming mode. When in programming mode, the device stops working until the programming is completed. When using Lattice Diamond Programmer, the Offline Mode is selected using operations starting with FLASH. Unless noted by the operation, the Flash sectors accessed are Feature, Configuration, and UFM.

When bitstream authentication and encryption are enabled, the programming needs to get through the decryption with pre-programmed AES key, and pass the bitstream signature checking with pre-programmed ECDSA public key. See Mach-NX Secure Enclave Usage Guide (FPGA-TN-02252) for details.

4.11.2. Background Programming

Background Programming allows the device to continue operating in user mode, while the configuration logic programs the on-chip Flash memory. When the on-chip Flash memory programming is completed, the device can download into the SRAM with REFRESH instruction. When using Diamond Programmer, the Background Mode is selected using operations starting with XFLASH. Unless noted by the operation, the Flash sectors accessed are Configuration and UFM.

4.12. Bitstream/PROM Sizes

The Mach-NX device is an SRAM-based FPGA. The SRAM configuration memory must be loaded from a non-volatile memory that can store all of the configuration data. The size of the configuration data is variable. It is based on the amount of logic available in the FPGA, and the number of pre-initialized Embedded Block RAM (EBR) components. A Mach-NX design using the largest device, with every EBR pre-initialized with unique data values, and generated without compression turned on, requires the largest amount of storage.

For information regarding SoC Function Block (SFB) configuration bitstream and the PFR firmware, refer to Mach-NX PFR and SFB Architecture Usage Guide (FPGA-TN-02230).

Storing configuration data in the Mach-NX device internal Flash memory has special considerations. The Flash memory in the Mach-NX device provides three types of independent sectors to store the configuration data.

The first type of sector is dedicated for use in holding compressed configuration data, and is called Configuration Flash or CFG. There are two CFG sectors, CFG0 and CFG1, in the Mach-NX device. Each CFG0 and CFG1 sector can store a whole bitstream for Mach-NX configuration (not include SoC image or SoC firmware, which are stored in an external SPI Flash).

The second type of sector is called the User Flash Memory (UFM). There are three UFM sectors, UFM0, UFM1 and UFM2 in the Mach-NX device which can be used as general purpose Flash memory.

The third type of sector is the Feature Row.

Figure 4.3 shows the Flash memory space of a Mach-NX device. CFG0 and CFG1 have the same memory size. UFM0 and UFM1, likewise, have the same size. In this figure, M, N, and O represent the total page number for CFG0/1, UFM0/1, and UFM2 respectively. Refer to Table 9.1 UFM Resources in Mach-NX Devices and Table 10.1 Configuration Flash Resources in Mach-NX Devices of the Mach-NX SFB Hardware Usage Guide (FPGA-TN-02222) for their specific values in each Mach-NX device density.

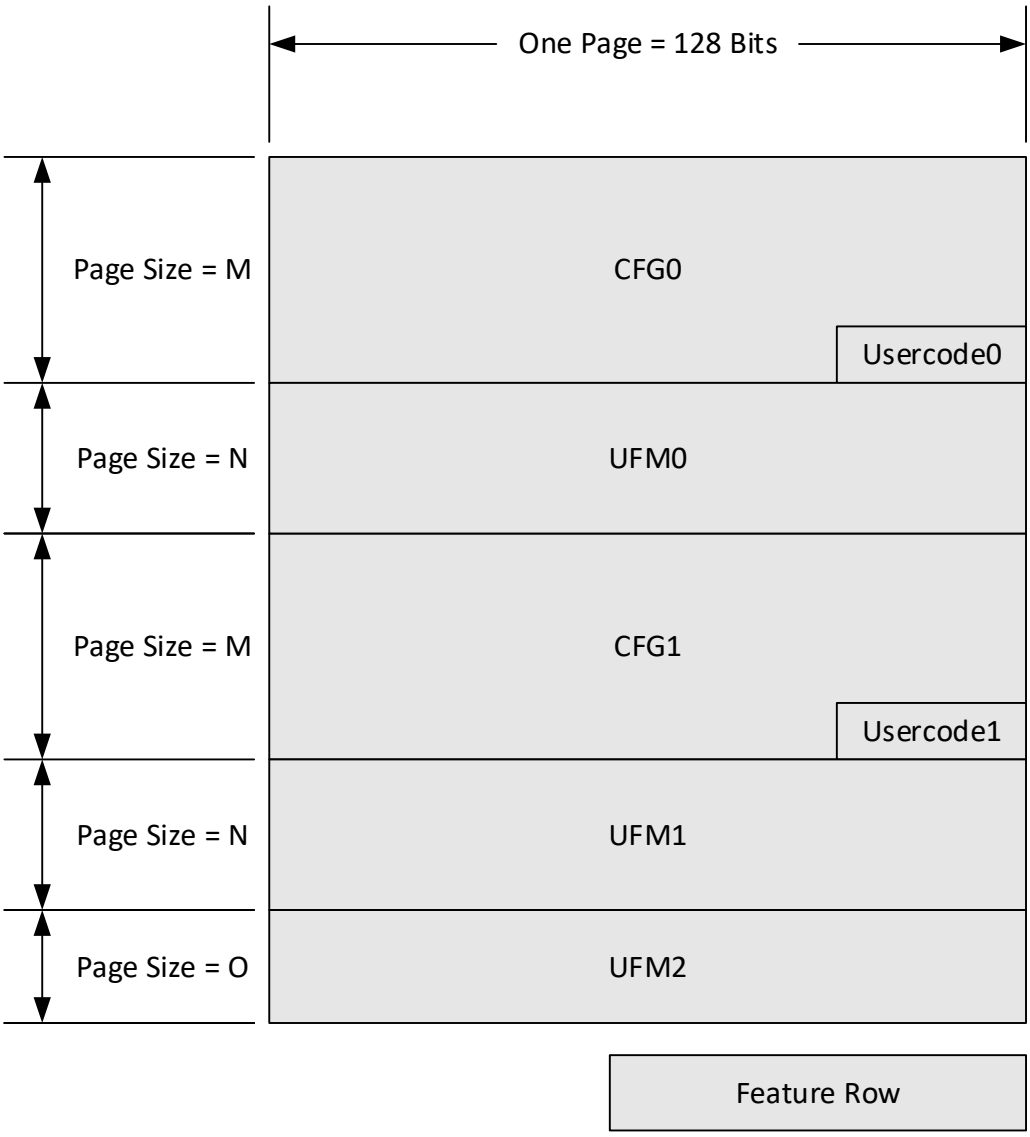


Figure 4.3. Flash Memory Space of a Mach-NX Device

The Configuration Flash CFG0 and CFG1 areused to store the compressed configuration data that is loaded into the SRAM configuration memory.

4.13. Feature Row

The Mach-NX device includes a Feature Row to control FPGA resources. The Feature Row permits more flexibility in selecting the functions available for configuration, increasing the number of available I/O on the device, and eliminating the need of making changes to your hardware. Feature Row can be erased or programmed independently.

Mach-NX Feature Row is used to determine how the Mach-NX SRAM configuration memory is loaded. When Feature Row is erased, Feature Row sets its value back to Hardware (HW) Default Mode state. The contents of Feature Row are typically specified using the Diamond Spreadsheet View. They can also be manually modified using Programming File Utility under Tools > Feature Row Editor.

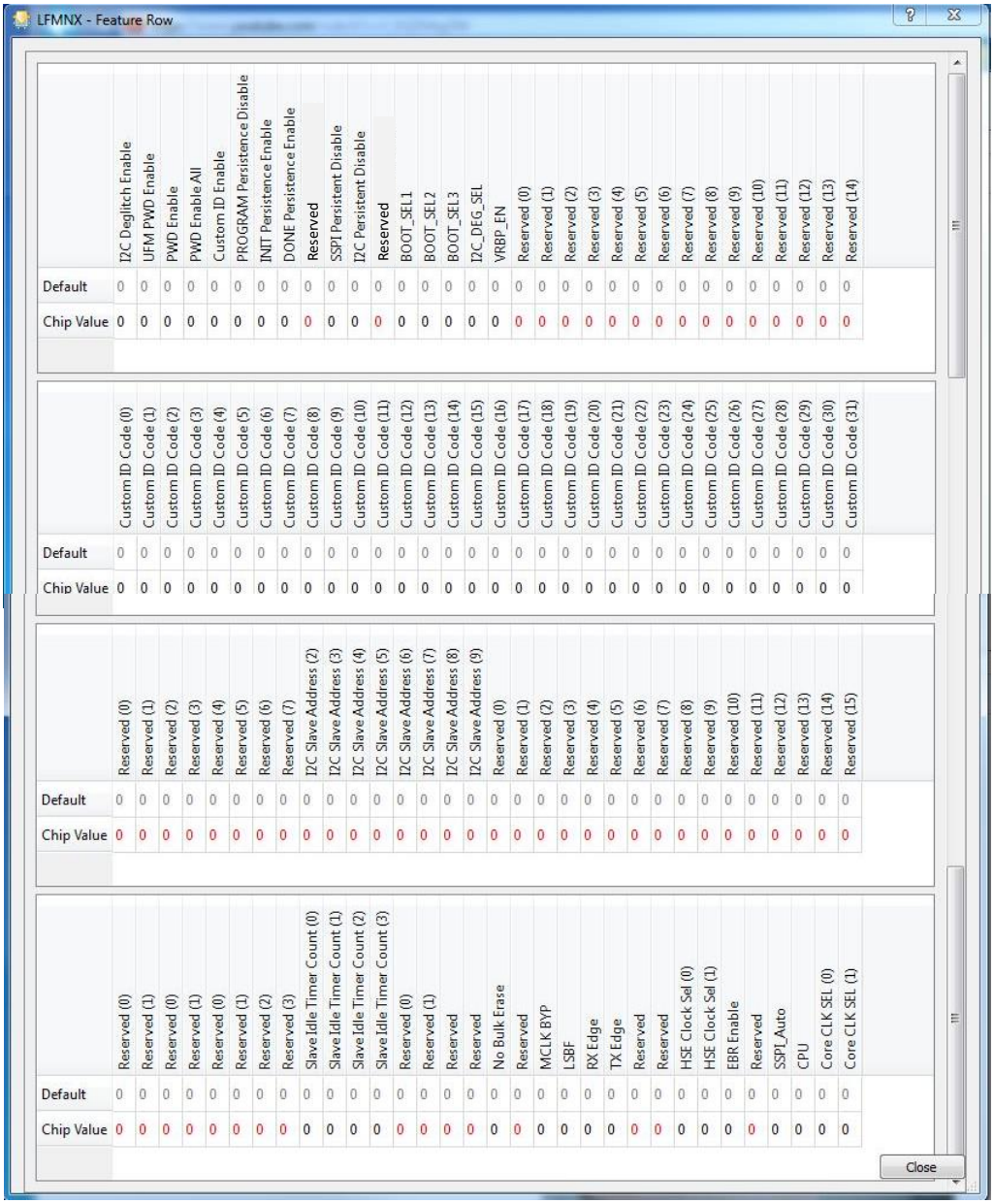


Figure 4.4. Feature Row Example

The functions controlled by the Feature Row and their default values for Mach-NX devices are shown in Table 4.2.

Table 4.2. Mach-NX Feature Row Elements

Feature	SW Default Mode State (Programmed)	HW Default Mode State (Erased)
PROGRAMN Persistence	Disabled	Enabled
INITN Persistence	Disabled	Disabled
DONE Persistence	Disabled	Disabled
Custom IDCODE	0x00000000	0x00000000
TraceID™	00000000	00000000
Security[1]	OFF	OFF
SSPI Port Persistence	Enabled	Enabled
I^2C Port Persistence	Disabled	Enabled
I^2C Programmable Primary Configuration Address[2, 3]	yyyxxxxx00	1111000000
my_ASSP Enable	OFF	OFF
Password Enable	OFF	OFF
Password Enable All	OFF	OFF
UFM Password Enable	OFF	OFF

Notes:

1. Enabled/disabled using the CONFIG_SECURE preference.
2. *y* and *x* are user-programmable from IPexpress™.
3. 1111000001 is a reserved address when the device is erased.

It is strongly recommended that the Feature Row only be modified during development, and rarely, if ever, upgraded in the field. The reason for this recommendation is that the Feature Row is responsible for controlling the availability of the Configuration Ports. It is possible to cause active Configuration Ports to become unavailable, preventing future updates.

Changing the Feature Row can also prevent the Mach-NX device from configuring. The PROGRAMN, INITN, and DONE control and status pins are enabled and disabled using the Feature Row. The PROGRAMN input pin may be recovered for use as a general purpose I/O. Erasing Feature Row state causes the PROGRAMN input to act as PROGRAMN, not as a general purpose I/O. If the general purpose I/O is driven active low, the Mach-NX device is never allowed to complete its configuration process.

Feature Row settings are specified using the Diamond Spreadsheet View, which allows you to edit the configuration settings for the Mach-NX device, and then save your settings in the Lattice Preference File (LPF). These settings are applied to the Mach-NX device configuration data during the Map, Place, and Route build phases. Alternately, the Feature Row of a device can be modified using the Program Feature Row utility in Diamond Programmer.

The key features of Feature Row are:

- Not intended to be modified in the field; only for development.
- Change in Feature Row settings may cause active configuration ports to become unavailable.
- Can be altered using Diamond Spreadsheet View.
- Can be programmed under Program Feature Row in Diamond Programmer.

4.14. Lock Bits and Lock Control Policy

Mach-NX devices contain security bits that, when set, can control the access of the SRAM configuration and Flash spaces. The Mach-NX device provides read, program, and erase permission control for each Flash sector, as well as the SRAM.

To support this, three security bits are deployed in each sector: SEC_PROG, SEC_READ, and SEC_ERASE. Once the bit is set, the corresponding operation, which is read, program, or erase, is prohibited. For the SRAM, once the SEC_PROG is set, beside blocking the configuration of the SRAM with the external configuration ports (JTAG, SSPI, I^2C, and MSPI).

Mach-NX device also provides Hard Lock and Soft Lock modes for flexibility of permission control. In Soft Lock mode, access from user logic through the internal AHB-Lite bus is not prohibited. This applies even if lock control bits are set for the external configuration ports (JTAG, SSPI, and I^2C). Also, user logic can alter the shadow register of access control bits to allow external configuration ports (JTAG, SSPI and I^2C) to access Flash memory. In Hard Lock mode, access from both the external configuration ports and the internal AHB-Lite bus is prohibited. Also, user logic cannot alter the status of access control bits. SEC_HLOCK bit is used to choose between Soft Lock and Hard Lock modes.

With the different combination of the four security bits mentioned above, different policies can be created for each Flash sector and the SRAM. Table 4.3 shows seven policies. Each line stands for one policy. *1* means Locked, and *0* means Unlocked. Polices shown in Table 4.4 are for reference purpose only. They are not limited to those listed. Desired policies must be set into the device using these four security bits. Take one example as Policy 6: SEC_HLOCK is set, so neither the external configuration ports nor the internal AHB-Lite bus can access and change the policy security bits. And since SEC_PROG, SEC_READ and SEC_ERASE are all set, neither external configuration ports nor the internal AHB-Lite bus can do anything to the Flash.

Table 4.3. Mach-NX Device Security Lock Control Bits

Mode	Policy #	Security Bits In Lock Policy Row for Each Sector				External CFG Port			Internal AHB-Lite Bus		
		SEC_HLOCK	SEC_READ	SEC_PROG	SEC_ERASE	READ	PROG	ERASE	READ	PROG	ERASE
No Security	0	0	0	0	0	Y[1]	Y[1]	Y[1]	Y[2]	Y[2]	Y[2]
Soft Lock	1	0	1	0	0	N[3]	Y[1]	Y[1]	Y[2]	Y[2]	Y[2]
	2	0	1	1	0	N[3]	N[3]	Y[1]	Y[2]	Y[2]	Y[2]
	3	0	1	1	1	N[3]	N[3]	N[3]	Y[2]	Y[2]	Y[2]
Hard Lock	4	1	1	0	0	N[4]	Y[2]	Y[2]	N[4]	Y[2]	Y[2]
	5	1	1	1	0	N[4]	N[4]	Y[2]	N[4]	N[4]	Y[2]
	6	1	1	1	1	N[4]	N[4]	N[4]	N[4]	N[4]	N[4]

Notes:

1. Accessible, but may be altered through AHB-Lite Bus.
2. Accessible.
3. Not accessible, but may be altered through AHB-Lite bus.
4. Not accessible.

A typical usage in Golden/Update image policy is:

1. Golden Image 0: It is hard locked as using policy 6.
2. User Working Image 1: It is soft locked as using policy 3.
3. To update Image 1, the command from the external configuration port is passed to the soft IP in the fabric. The fabric security IP validates and authenticates the external port lock/unlock request. If authenticated, the fabric uses the internal AHB-Lite bus to unlock Image 1.
4. After successful update and Image 1 passes the audit, the external configuration port can request the fabric IP to relock Image 1 to prevent it from being updated by any external activity.

The Mach-NX device also provides a set of permission control settings to disable access from the separate external configuration ports, such as JTAG, Slave SPI, and Slave I^2C. These configuration ports support both Hard Lock and Soft Lock modes. Each port has its control bits. Any command or part of commands sent by these ports can be blocked with the settings of the control bits.

All Mach-NX security lock control bits can be set through the external configuration port using the Diamond Programmer tool or through the internal AHB-Lite bus using user logic.

4.15. sysCONFIG Ports

Table 4.4. Mach-NX Device Programming and Configuration Ports

Interface	Port	Description
JTAG	JTAG (IEEE 1149.1 and IEEE 1532 compliant)	4-wire or 5-wire JTAG Interface
sysCONFIG	SSPI	Slave Serial Peripheral Interface (SPI)
	I^2C	Inter-integrated Circuit (I^2C) Interface
Internal	AHB-Lite Internal	AHB-Lite bus interface

4.16. sysCONFIG Pins

The Mach-NX device provides a set of sysCONFIG I/O pins that you can use to program and configure the FPGA. The sysCONFIG pins are grouped together to create ports JTAG, SSPI, and I^2C 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 you to adhere to the following guidelines:

- You must disable the unused port. You can accomplish this by using the Diamond Spreadsheet View's Global Preferences tab. Each configuration port is listed in the sysCONFIG options tree.
- You must prevent external logic from interfering with device programming. Make sure that recovered sysCONFIG pins are not asserted when the Mach-NX device is in Feature Row HW Default Mode state. One example is driving PROGRAMN with an active low signal after the Mach-NX device is in Feature Row HW Default Mode state. Failure to reprogram the Feature Row with PROGRAMN disabled prevents the FPGA from configuring and entering user mode.
- Use care when using JTAGENB to selectively enable and disable the JTAG port. Any external logic connected to the JTAG I/O must not contend with the JTAG programming port.

Table 4.5 lists the default state of the shared sysCONFIG pins. An HW default Mode Feature Row device has the JTAG, SPI Slave and I^2C ports enabled. Upon entry to user mode, the Mach-NX device, the default state of the SSPI, and I^2C sysCONFIG pins become general purpose I/O. This means you lose the ability to program the Mach-NX device using I^2C when using the default sysCONFIG port settings. To retain the I^2C sysCONFIG pins in user mode, be sure to enable them using the Diamond Spreadsheet View editor.

Unless specified otherwise, the sysCONFIG pins are powered by the VCCIO0 voltage. It is crucial you take this into consideration when provisioning other logic attached to Bank 0.

The function of each sysCONFIG pin is described in detail in Table 4.5 and Table 4.6.

Table 4.5. Default State of the sysCONFIG Pins

Pin Name	Associated sysCONFIG Port	Pin Function in Feature Row Erased Mode (Configuration/HW Default Mode)	Pin Direction (Configuration Mode)	Default Function in User Mode (SW Default Mode)
PROGRAMN	SDM	PROGRAMN	Input with weak pull-up	User-defined I/O
INITN	SDM	I/O	I/O with weak pull-up	User-defined I/O
DONE	SDM	I/O	I/O with weak pull-up	User-defined I/O
CCLK	SSPI	SSPI	Input with weak pull-up	SSPI
SN	SSPI	SSPI	Input with weak pull-up	SSPI
SI	SSPI	SSPI	Input	SSPI
SO	SSPI	SSPI	Output	SSPI
SDA	I^2C^*	I^2C	Bidirectional	User-defined I/O

Table 4.6. sysCONFIG Port Default Settings in Lattice Diamond

sysConfig Port	Diamond Default*
SDM_PORT	DISABLE
SLAVE_SPI_PORT	ENABLE
I2C_PORT	DISABLE
JTAG_PORT	ENABLE

*__Note__: This default setting can be modified in the Diamond Spreadsheet View, Global Preferences tab.

4.16.1. Self-Download Port Pins

PROGRAMN

The PROGRAMN is an input used to configure the FPGA. The PROGRAMN pin, when enabled, is sensitive to a high-to-low transition, 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 Mach-NX device from leaving the Initialization phase. The PROGRAMN has a minimum pulse width assertion period, t_{PRGMJ} (Figure 4.6), for it to be recognized by the FPGA. You can find this minimum time in the sysCONFIG Timing Specification section of the Mach-NX Device Family Data Sheet (FPGA-DS-02084).

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

- If the device is currently being programmed via JTAG, then PROGRAMN is ignored until the JTAG mode programming sequence is complete.
- Toggling the PROGRAMN pin during device configuration interrupts the process and restarts the configuration cycle.
- Asserting PROGRAMN on a device in Feature Row HW Default Mode state disables the SSPI and I^2C ports. Start SSPI or I^2C programming operations after PROGRAMN is deasserted.
- PROGRAMN is active during power-up, even when it is reserved as a general purpose I/O. Do not allow any input signal attached to PROGRAMN to transition from high to low at a frequency greater than the VCC (min) to INITN rising edge time period. High to low PROGRAMN assertions more frequently prevent the Mach-NX device from configuring, causing the FPGA to remain in a continuous RESET condition. See Figure 4.5.

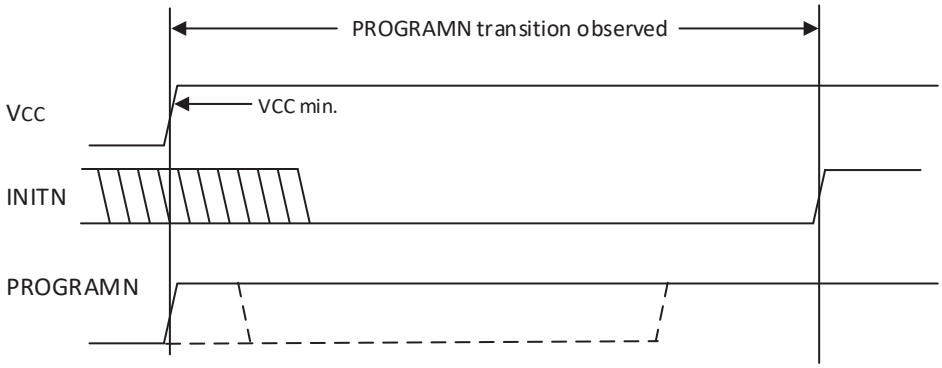


Figure 4.5. Period PROGRAMN is Always Observed

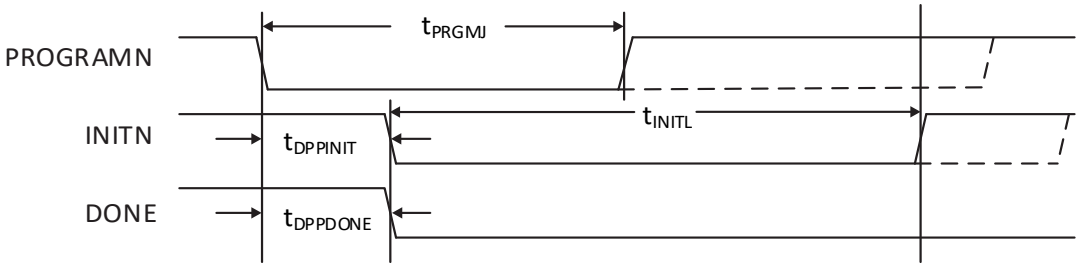


Figure 4.6. Configuration from PROGRAMN Timing

INITN

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

- After power is applied, after a PROGRAMN is asserted, or after a REFRESH command is transmitted, INITN goes low to indicate the SRAM configuration memory is being erased. The low time assertion is specified with the t_{INITL} parameter.
- After the t_{INITL} time period elapses, the INITN pin is deasserted, that is active high, to indicate that the Mach-NX device is ready for its configuration bits. The Mach-NX device begins loading configuration data from the internal Flash.
- INITN can be asserted low by an external agent before the t_{INITL} time period elapses 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} elapses and the INITN pin goes high, any subsequent INITN assertion signals that an error is detected by the Mach-NX device during configuration.

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

- Power is applied
- PROGRAMN falling edge occurs
- The IEEE 1532 REFRESH command is sent using a slave configuration port (JTAG, SSPI or I^2C)

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.

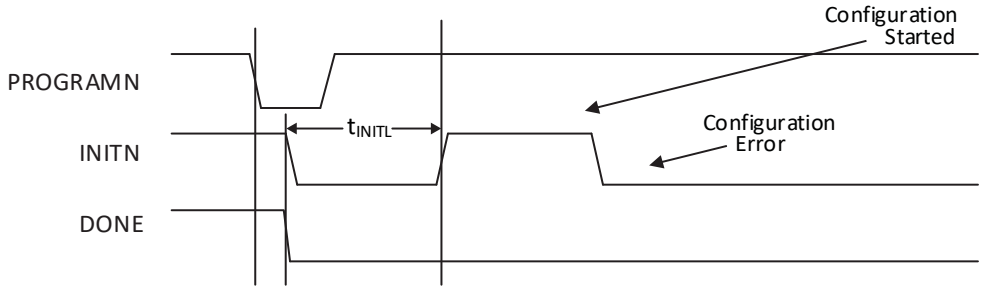


Figure 4.7. Configuration Error Notification

The INITN pin of a Mach-NX device is not visible external to the device when in the Feature Row HW Default Mode state. The INITN pin, when in this mode, is pulled high by default. The INITN behavior described in Figure 4.7 is only visible outside the Mach-NX device when the INITN pin is enabled.

The INITN can be recovered as a general purpose I/O. By default, the INITN pin is disabled. You can use the Diamond Spreadsheet View to enable it.

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 fails the configuration when the following happens:

- The bitstream CRC error is detected
- The invalid command error detected
- A time out error is encountered when loading from the on-chip Flash
- The program done command is not received when the end of on-chip SRAM configuration or on-chip Flash memory is reached

DONE

The DONE pin is a bidirectional 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 DONE output pin is controlled by the SDM_PORT configuration parameter that is modified in the Diamond Spreadsheet View. By default the DONE pin is a general purpose I/O when the Mach-NX device is in the Feature Row HW Default Mode state. The default mode causes the Mach-NX device to automatically sequence through the wake-up sequence after the internal DONE bit is asserted. The FPGA does not stall waking up waiting for the DONE pin to be asserted high.

The FPGA can be held from entering user mode indefinitely by having an external agent keep the DONE pin asserted low. In order to use DONE to stall entering user mode, the SDM_PORT must enable the DONE I/O, and the FPGA Feature Row must be programmed. This feature is supported in Diamond 3.5 and later. Earlier versions of Diamond do not enable the stall feature when SDM_PORT enables DONE I/O. 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 via an active configuration port.

Sampling the DONE pin is a way for an external device to tell if the FPGA configuration is complete. However, when using IEEE 1532 JTAG to configure SRAM the DONE pin is driven by a boundary scan cell, so the state of the DONE pin has no meaning during IEEE 1532 JTAG configuration. Once configuration is complete, the DONE pin takes on the behavior defined by the SDM_PORT setting in the Feature Row. The DONE pin is also pulled high when the FPGA is in the Feature Row HW Default Mode state. This behavior can make a part appear to be successfully configured to other logic monitoring the DONE pin.

4.16.2. Slave SPI Configuration Port Pins

Table 4.7. Slave SPI Configuration Port Pins

Pin Name	Function	Direction	Description
CCLK	CCLK	Input with weak pull-up	Clock used to time data transmission/reception from an external SPI master device to the Mach-NX Configuration Logic.
SI	SI	Input	SI carries output data from the external SPI master to the Mach-NX Configuration Logic
SO	SO	Output	SO carries output data from the Mach-NX configuration logic to the external SPI master
SN	SN	Input with weak pull-up	Mach-NX configuration logic slave SPI chip select input. SN is an active low input.

CCLK

The CCLK, when active, are clocks used to sequentially load the configuration data for the FPGA. The pin functions as follows:

The Mach-NX CCLK pin's default state in the Feature Row HW Default Mode state acts as the configuration clock, CCLK. This allows an external SPI master controller to program the Mach-NX device. The maximum CCLK frequency and the data setup/hold parameters are found in the sysCONFIG Port Timing Specifications of the Mach-NX Device Family Data Sheet (FPGA-DS-02084). The Feature Row must be configured to enable the Slave SPI Port if you want to use the port to reprogram the Mach-NX device after it enters the user mode.

SN

The SN pin is the Slave SPI ports chip select. An external SPI bus master asserts the SN pin active low to perform actions using the Mach-NX device programming and configuration logic. The SN pin is available when the Mach-NX device is in the Feature Row HW Default Mode state, and in user mode when the Slave SPI port is set to the ENABLE setting. The SN pin is a general purpose I/O in user mode when the Slave SPI port is set to the DISABLE setting.

Proper operation of the Mach-NX device depends upon maintaining the SN pin in the correct state:

- SN must be deasserted (that is, held high) when configuring using Master SPI mode
- SN must be deasserted when the Mach-NX device is in user mode
- SN must be deasserted when accessing the configuration logic in the Mach-NX device using I^2C
- When SN is asserted, CSSPIN must be deasserted. Deasserting CSSPIN places the shared SPI pins into a high impedance state.
    - The Master SPI port and the Slave SPI port share three common pins, SI/SISPI, SO/SPISO, and MCLK/CCLK. The Mach-NX device permits both ports to be available at the same time. They are not permitted to be accessed at the same time. The Slave SPI and the Master SPI port must be time multiplexed when both ports are enabled.

Lattice recommends the SN pin be pulled high externally to augment the weak internal pull-up.

SI

The SI is an input data pin when using the Slave SPI mode.

The sysCONFIG preferences, SLAVE_SPI_PORT, must be set to ENABLE to preserve this pin as SI and allow access to the SPI interface.

SO

The SO pin is an output data pin when using the Slave SPI mode.

The sysCONFIG preferences, SLAVE_SPI_PORT, must be set to ENABLE to preserve this pin as SO and allow access to the SPI interface.

4.16.3. I²C Configuration Port Pins

SCL

The Mach-NX device provides an I²C configuration port. The SCL is the I²C Serial Clock pin, and is used to initiate and time transactions on the I²C bus. It is a bidirectional, open-drain signal that is an output when the Mach-NX I²C controller is mastering transactions on the bus, and is an input when an external I²C master is accessing resources inside the Mach-NX device. SCL requires an external pull-up resistor to operate.

The SCL pin is available when the Mach-NX device is in the Feature Row HW Default Mode state. You must ENABLE the I2C_PORT and instantiate the EFB for the I²C port to continue to be available in user mode (see the I2C Configuration Mode section for details). The SCL pin becomes a general purpose I/O if you do not set to ENABLE the I2C_PORT.

SDA

The SDA pin is the I²C serial data input/output pin. It is bidirectional, open-drain, and requires an external pull-up resistor to operate. The pin changes direction dynamically during data transactions on the I²C bus. The current state depends on the current bus master and the operation being performed by that master.

The SDA pin is available when the Mach-NX device is in the Feature Row HW Default Mode state. You must ENABLE the I2C_PORT and instantiate the EFB if you want the I²C port to continue to be available in user mode (see the I2C Configuration Mode section for details). The SDA pin becomes a general purpose I/O if you do not set to ENABLE the I2C_PORT.

4.16.4. JTAG Configuration Port Pins

The JTAG pins provide a standard IEEE 1149.1 Test Access Port (TAP). The JTAG port is the only configuration port on the Mach-NX device that is capable of performing configuration, programming, and multi-device configuration functions. Programming and configuration over the JTAG port uses IEEE 1532 compliant commands. In addition to the IEEE 1532 capabilities, the Mach-NX device provides all of the mandatory IEEE 1149.1 Test Access Port commands allowing printed circuit board assembly verification.

The JTAG port is enabled by default when the Mach-NX device is in the Feature Row HW Default Mode state. Like all of the other configuration port pins the JTAG pins can become general purpose I/O. Unlike the other ports, the default state for the JTAG port is to remain active in user mode, that is, in ENABLE state. The JTAG pins can be recovered to be general purpose I/O by setting the JTAG_PORT preference to the DISABLE state. It is recommended the JTAG port remain dedicated programming pins.

The JTAG port, when set to DISABLE state, enables the JTAGENB input. JTAGENB permits the JTAG pins to be multiplexed. Asserting JTAGENB high causes the JTAG pins to take on the IEEE 1149.1 personality. De-asserting JTAGENB, that is, driven low causes the JTAG port pins to become general purpose I/O. Design the JTAG port circuitry carefully when taking advantage of JTAG port pin multiplexing. Avoid bus contention between logic attached to the JTAG port.

When the device is programmed through IEEE 1149.1 control, the sysCONFIG programming pins, such as DONE, cannot be used to determine programming progress. This is because the state of the boundary scan cell drives the pin, per the IEEE JTAG standard, rather than normal internal logic.

Table 4.8. JTAG Port Pins

Pin Name	Pin Function (Configuration Mode)	Pin Direction (Configuration Mode)	Default Function (User Mode)
TDI	TDI	Input with weak pull-up	TDI
TDO	TDO	Output with weak pull-up	TDO
TCK	TCK	Input	TCK
TMS	TMS	Input with weak pull-up	TMS
JTAGENB	I/O	Input/output with weak pull-down	I/O

TDO

The Test Data Output (TDO) pin is used to shift out serial test instructions and data. When TDO is not being driven by the internal circuitry, the pin is in a high impedance state. The only time TDO is not in a high impedance state is when the JTAG state machine is in the Shift IR or Shift DR state. This pin should be wired to TDO of the JTAG connector, or to TDI of a downstream device in a JTAG chain. An internal pull-up resistor on the TDO pin is provided. The internal resistor is pulled up to VCCIO Bank 0.

TDI

The Test Data Input (TDI) pin is used to shift in serial test instructions and data. This pin should be wired to TDI of the JTAG connector, or to TDO of an upstream device in a JTAG chain. An internal pull-up resistor on the TDI pin is provided. The internal resistor is pulled up to VCCIO of Bank 0.

TMS

The Test Mode Select (TMS) pin is an input pin that controls the progression through the 1149.1 compliant state machine states. The TMS pin is sampled on the rising edge of TCK. The JTAG state machine remains in or transitions to a new TAP state depending on the current state of the TAP, and the present state of the TMS input. An internal pull-up resistor is present on TMS per the JTAG specification. The internal resistor is pulled to the VCCIO of Bank 0.

TCK

The test clock pin (TCK) provides the clock used to time the other JTAG port pins. Data is shifted into the instruction or data registers on the rising edge of TCK and shifted out on the falling edge of TCK. The TAP is a static design permitting TCK to be stopped in either the high or low state. The maximum input frequency for TCK is specified in the DC and Switching Characteristics section of Mach-NX Device Family Data Sheet (FPGA-DS-02084). The TCK pin does not have a pull-up. An external pull-down resistor of 4.7 kΩ is recommended to avoid inadvertently clocking the TAP controller as power is applied to the Mach-NX device.

JTAGENB

The JTAG ENABLE pin, also known as the IEEE 1149.1 conformance pin, is an input pin that can be used to multiplex the JTAG port. The JTAGENB pin is only active in user mode. The JTAGENB pin is a user I/O while the JTAG port is in the ENABLE state. Figure 4.8 shows the default behavior of the JTAG port of a Mach-NX device.

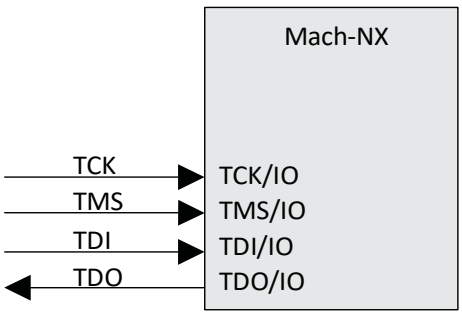


Figure 4.8. Default JTAG Port with JTAG_PORT = ENABLE

The JTAG port can become general purpose I/O by setting the JTAG_PORT preference in the Diamond Spreadsheet View to the DISABLE state. When the JTAG port is in the DISABLE state, the JTAGENB pin becomes a dedicated input. Driving the JTAGENB low disables the JTAG port and the four JTAG pins become general purpose I/O. Driving the JTAGENB input high enables the JTAG port. Figure 4.9 shows JTAG port behavior under the control of the JTAGENB.

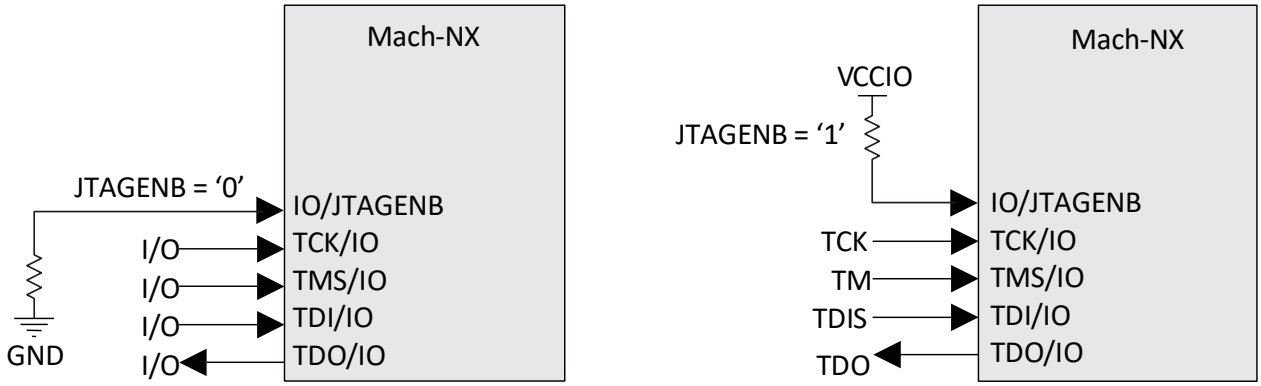


Figure 4.9. JTAG Port Behavior with JTAG_PORT = DISABLE

It is critical that the external logic attached to the JTAG I/O pins does not contend with a JTAG programming system when using the JTAGENB feature. The external logic must ignore any JTAG transactions performed by an external programming system.

Lattice parallel port or USB download cables provide an output called ispEN. The ispEN signal can be attached to the JTAGENB input to control the availability of the JTAG port. An alternate mechanism to control the JTAGENB input is to use a shunt that can be installed or removed as required.

5. Configuration Modes

The Mach-NX device provides multiple options for loading the configuration SRAM from a non-volatile memory. The previous section describes the physical interface necessary to interact with the Mach-NX device configuration logic. This section focuses on describing the functionality of each of the different configuration modes. Descriptions of important settings required in the Diamond Spreadsheet View are also discussed.

5.1. SDM Mode

The advantages of Self Download Configuration Mode (SDM) include:

- Speed: The Mach-NX device is ready to run in a few milliseconds depending on the density of the device.
- Security: The configuration data is never seen outside the device during the load to SRAM. You can prevent the internal memory from being read.
- Reduced cost: There is no need to purchase a PROM specifically reserved for programming the Mach-NX device.
- Reduced board space: Elimination of an external PROM allows your board to be smaller.

The Mach-NX device retrieves the configuration data from the internal Flash, CFG0 or CFG1, when it is using Self Download Mode. To set the Mach-NX device operation using the SDM Configuration Mode, you must:

- Store the entire configuration data in CFG0 or CFG1.
- Set the preference as shown in Table 5.1.

Table 5.1. SDM Configuration Software Settings

Preference	Setting
CONFIGURATION	CFG | CFG_EBRUFM | CFGUFM
PRIMARY_BOOT	IMAGE_0 | IMAGE_1
SECONDARY_BOOT	NONE

SDM is triggered when power is applied, a REFRESH command is received, or by asserting the PROGRAMN pin.

5.2. Dual Boot Configuration Mode

The Mach-NX device, when set up in Dual Boot Configuration Mode, has two types of configurations: Golden image dual configuration and Version-based dual configuration.

5.2.1. Golden Image Dual Configuration

In Golden image dual configuration, Mach-NX device tries to configure first from the primary image stored in an internal Flash memory sector. If the first configuration fails, the Mach-NX device attempts to configure itself from the golden image stored in another internal Flash memory sector. The dual boot sequence may be changed if you want to use the CONFIGURATION/PRIMARY_BOOT/SECONDARY_BOOT options in the Diamond software spreadsheet view. Mach-NX device supports seven golden image dual boot sequences. The primary image and/or the golden image can be flexibly stored in the internal Flash memory. These dual boot sequences can be only from the internal Flash memory, such as CFG0_CFG1, CFG1_CFG0.

The first boot attempt is from the primary configuration image. If the primary configuration fails, the second boot attempt is from the golden/failsafe configuration image. If both fail, device stays at the un-programmed state. The primary image can fail in one of two ways:

- A bitstream CRC error is detected
- A time-out error is encountered when loading

A CRC error is caused by incorrect or corrupt data. Data is read from the primary image in rows. As each row enters the configuration engine, the data is checked for CRC consistency. Before the data enters the configuration SRAM, the CRC must be correct. Any incorrect CRC causes the device to erase the configuration SRAM and retrieve configuration data from the golden/failsafe image location.

It is possible for the data to be correct from a CRC calculation perspective, but not be functionally correct. In this instance, the internal DONE bit is never active. The Mach-NX device counts the number of master clock pulses provided after the Power On Reset signal is released. When the count expires without DONE becoming active, the FPGA attempts to get its configuration data from the golden/failsafe image location.

Dual boot Configuration Mode typically requires two configuration data files. One of the two configuration data files is a golden or fail-safe image that is rarely, if ever, updated. The second configuration data file is a primary or working image that is routinely updated. One Diamond project can be used to create both the working and the fail-safe configuration data files. Configure the Diamond project with an implementation named working, and an implementation named failsafe. Read the Diamond Online Help for more information about using Diamond implementations.

Use the following preferences shown in Table 5.2 to build a dual-boot design.

Table 5.2. Dual Boot Configuration Software Settings

Preference	Setting
CONFIGURATION	CFG|CFG_EBRUFM|CFGUFM
PRIMARY_BOOT	IMAGE_0|IMAGE_1| LATEST|FORMER
SECONDARY_BOOT	NONE|IMAGE_0|IMAGE_1| LATEST|FORMER

The legal combinations of CONFIGURATION, PRIMARY_BOOT and SECONDARY_BOOT settings are listed in Table 5.3.

Table 5.3. Legal combination of CONFIGURATION, PRIMARY_BOOT and SECONDARY_BOOT Settings

CONFIGURATION	PRIMARY_BOOT	SECONDARY_BOOT	Boot	Boot Sequence
CFG|CFG_EBRUFM|CFGUFM	IMAGE_0	NONE	Single	CFG0 (SW default)
		IMAGE_1	Dual	CFG0_CFG1
	IMAGE_1	NONE	Single	CFG1
		IMAGE_0	Dual	CFG1_CFG0
	LATEST	FORMER	Dual	Latter_Former
	FORMER	LATEST	Dual	Former_Latter

5.2.2. Version Based Dual Configuration

For each of the configuration images stored in the internal CFG0 and CFG1, there is one 4-bit version tag for it. After programming an image into one of CFG0 and CFG1, a new version number is generated by increasing the version number for the other of CFG0 and CFG1 by one. And it is automatically programmed and used for the new image. Version tag *0000* is assumed to be larger than *1111*. It means the later programmed bitstream has the larger version number. In this type of configuration, the dual boot configurations are from CFG0 and CFG1. And the boot sequence depends on the version number. The first configuration can be from the latter image of the two images stored in CFG0 and CFG1 with PRIMARY_BOOT = LATEST and SECONDARY_BOOT = FORMER, or from the former bitstream with PRIMARY_BOOT = FORMER and SECONDARY_BOOT = LATEST. If this fails, configure from the other image. If both fail, device stays at the un-programmed state.

5.3. Slave SPI Mode (SSPI)

The Mach-NX device provides a Slave SPI configuration port that allows you to access features provided by the configuration logic. You can reprogram the SRAM, Flash and Feature Row, and access status/control registers within the configuration logic block. Reprogramming the Flash can be done using offline or transparent operations.

Table 5.4. Slave SPI Port Pins

Pin Name	Description
CCLK	Configuration clock input that is driven by an SPI master controller.
SI	Serial Data Input to the Mach-NX device configuration logic for command and data.
SO	Serial Data Output from the Mach-NX device configuration logic.
SN	Chip select to enable the Mach-NX device configuration logic.

In the Slave SPI mode, the MCLK/CCLK pin becomes configuration clock (CCLK). Input data is read into the Mach-NX device on the SI pin at the rising edge of CCLK. Output data is valid on the SO pin at the falling edge of CCLK. The SN acts as the chip select signal. When SN is high, the SSPI interface is deselected and the SO/SPISO pin is tri-stated. Commands can be written into and data read from the Mach-NX device when SN is asserted. The Mach-NX device SSPI port only accepts Mode 0 bus transactions to the configuration logic.

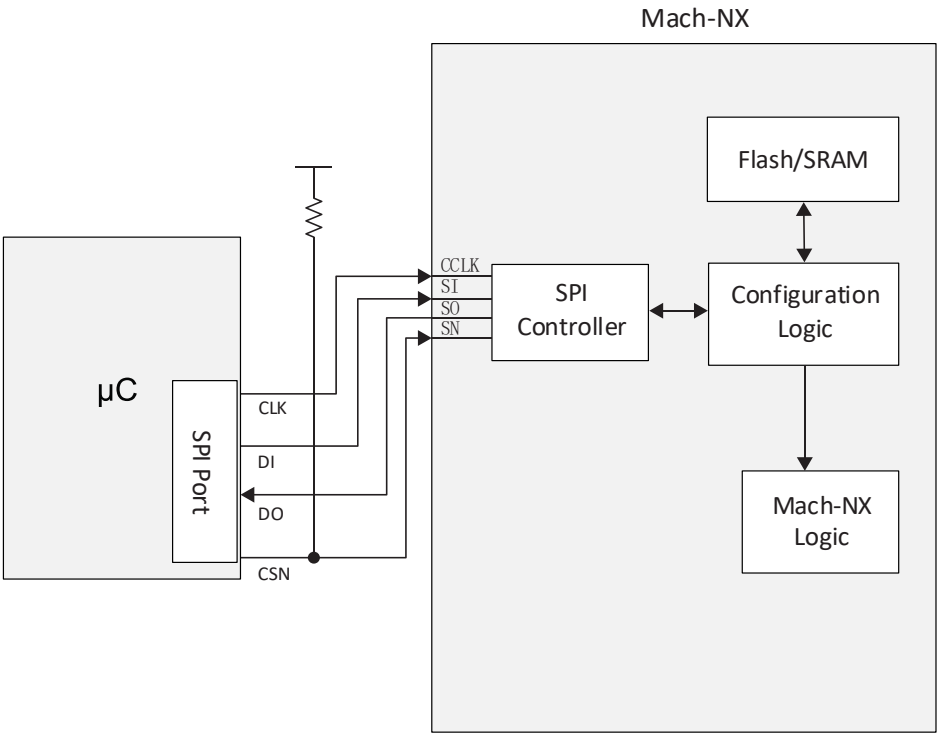


Figure 5.1. Slave SPI Configuration Mode

The SSPI port is active when the Mach-NX device is in Feature Row HW Default Mode state. Diamond default preference for the SLAVE_SPI_PORT is to ENABLE the port. Use the Spreadsheet View to disable the SLAVE_SPI_PORT preference in your design to keep the SSPI port to be used as general purpose I/O in user mode. Lattice recommends you keep a secondary programming port active in the event the SSPI port is accidentally disabled.

The SSPI port is used to erase, program, and verify the Configuration Flash, User Flash Memory, and the Feature Row. It is not capable of directly accessing the configuration SRAM. To prevent unintentional erasure of the Feature Row, it is recommended the SSPI port be used to perform transparent updates of the Flash memory. The SSPI port can issue a REFRESH command to make a newly programmed image active. The REFRESH command can be safely used when the Mach-NX device is using External or Dual Boot Configuration Mode, because the REFRESH operation does not begin until SN is deasserted.

Programming the Mach-NX device using the SSPI port is complex. As such, Lattice provides C source code called SSPIEMbedded to insulate you from the complexity of programming the Mach-NX device. Use SSPIEmbedded to reprogram the Flash or SRAM.

Accessing the status registers is less complex and does not require the use of the SSPIEmbedded code.

5.4. I²C Configuration Mode

The Mach-NX device has an I²C configuration port for use in accessing the configuration logic. An I²C master can communicate to the configuration logic using 10-bit or 7-bit addressing modes. The I²C SCL input can accept a clock frequency up to 400 kHz. You can reprogram the SRAM, Flash and Feature Row, and access status/control registers within the configuration logic block. Reprogramming the Flash can be done in offline or in transparent operations.

Table 5.5. I²C Port Pins

Pin Name	Description
SCL	I²C bus clock
SDA	I²C bus data line

The I²C configuration port is available when the Mach-NX device is in Feature Row erased state. The default state set for the I2C_PORT in the Diamond design software is to place the I2C_PORT in the DISABLE state. You must make sure the I2C_PORT is set to the ENABLE state to leave the I²C interface active in user mode. Lattice recommends making a second configuration port available, for example, JTAG, to recover from erroneously disabling the I²C port.

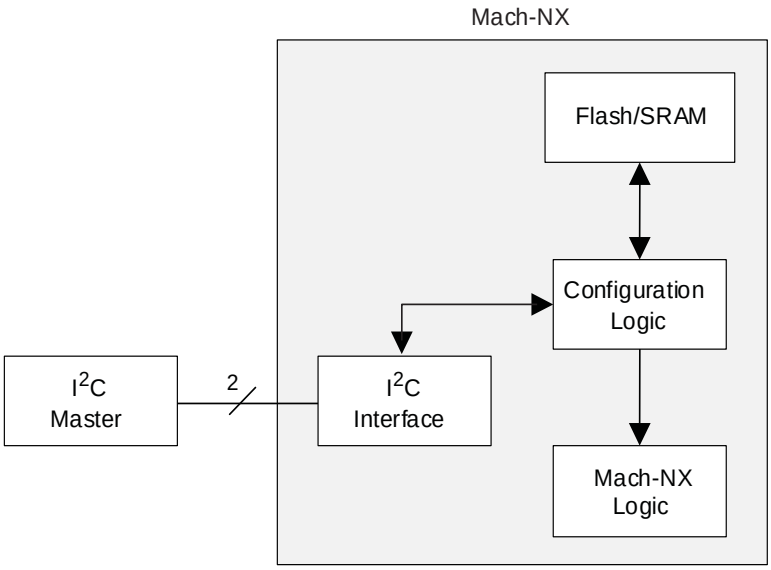


Figure 5.2. I²C Configuration Logic

There are two hardened I²C controllers in a Mach-NX device, a primary and a secondary. The primary controller provides an interface to the Mach-NX configuration logic. The primary I²C controller is the only one that permits access to the configuration logic or can be a user mode I²C controller. The Secondary I²C controller is always a user mode I²C controller.

When the Mach-NX device is in Feature Row HW Default Mode state, the I²C port is enabled. You may interact with the primary I²C controller. Whenever the I²C port is enabled, access to the configuration logic is possible. It is necessary to instantiate the Embedded Function Block (EFB) to preserve access to the configuration logic in user mode. Moreover, when instantiated, the EFB 'wb_clk_i' input must be connected to a valid clock source of at least 7.5 times the I²C bus rate, for example, >3.0 MHz when I²C rate = 400 kHz.

An external I²C master accesses the configuration logic using address 1000000 in 7-bit mode or 1111000000 in 10-bit mode unless the EFB I²C base address is modified. Use IPexpress, not Spreadsheet View, to modify the address to which the Primary and Secondary I²C controllers respond. It is necessary to instantiate the EFB to change the address. The address is shared by the Primary and Secondary I²C controllers.

Table 5.6 shows the address decoding used to access the I²C resources in the Mach-NX device.

Table 5.6. Slave Addresses for I²C Ports

Slave Address	I²C Function
yyyxxxxx00	Primary I²C controller configuration logic address. Always responds to 7-bit or 10-bit addresses.
yyyxxxxx01	User mode primary I²C controller address.
yyyxxxxx10	User mode secondary I²C controller address.
yyyxxxxx11	Primary I²C configuration logic Reset. Always responds to 7-bit or 10-bit addresses.

The fourth I²C resource in the Mach-NX device is located at offset 3. In some instances, an I²C memory transaction to the configuration logic may be interrupted or abandoned. It is possible for a command to be accepted by the configuration logic that causes the configuration logic to respond with data. In the event that the I²C memory transaction is interrupted or abandoned, the configuration logic continues to return the queued data. New incoming I²C commands may be considered padding bytes or may be misinterpreted. Clear this condition by writing any value to offset 3. The configuration logic command interpreter resets, any queued data is flushed, and subsequent I²C memory transactions to the configuration logic operates correctly.

5.5. AHB-Lite Configuration Mode

The Mach-NX device can access the Configuration Flash, User Flash Memory, and the Feature Row from an internal AHB-Lite bus. To use the AHB-Lite bus, the Embedded Function Block (EFB) must be inserted into your design. You can design the logic to interface to the EFB, then perform AHB-Lite bus transactions to access resources attached to the configuration logic.

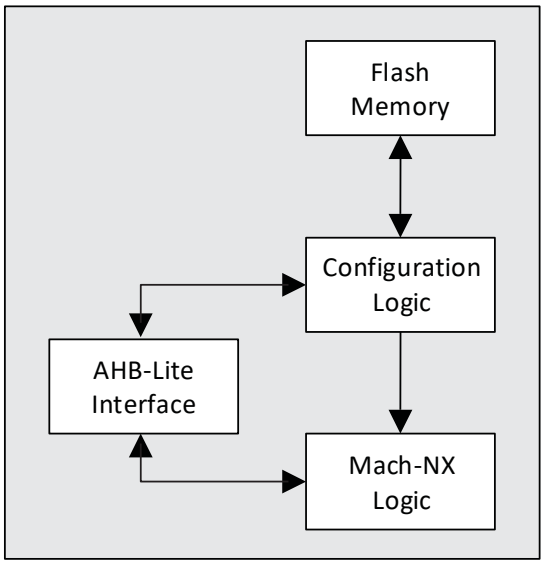


Figure 5.3. AHB-Lite Configuration Mode

In order to access the AHB-Lite interface, the Mach-NX device must be in user mode. Accessing and updating the resources made available by the configuration logic must be completed in Transparent mode. Attempting accesses to the configuration logic in offline mode causes a deadlock because the Mach-NX device leaves user mode.

You can get more detailed information about the Mach-NX AHB-Lite interface from Mach-NX SFB Hardware Usage Guide (FPGA-TN-02222).

5.6. JTAG Mode

The JTAG port is the most flexible configuration and programming port available on the Mach-NX device. The JTAG provides:

- Offline Flash programming
- Transparent Flash memory programming
- Offline SRAM configuration
- Full access to the Mach-NX configuration logic
- Device chaining
- IEEE 1149.1 testability
- IEEE 1532 compliant programming

The JTAG port is available when the Mach-NX device is in Feature Row HW Default Mode state. The Mach-NX JTAG port pins are not dedicated to performing the IEEE 1149.1 TAP function. The JTAG port may be recovered for use as general purpose I/O or vice versa. See the

sysCONFIG Pins section for details.

The Mach-NX JTAG port is a valuable asset due to its flexibility. It provides the best capabilities for system and device debug. Lattice recommends the JTAG port remain accessible in every Mach-NX design. Advantages for keeping the JTAG port active include:

- Multi-chain Architectures

  The JTAG port is the only configuration and programming port that permits the Mach-NX device to be combined in a chain of other programmable logic.

- Reveal Debug

  The Lattice Reveal debug tool is an embeddable logic analyzer tool. It allows you to analyze the logic inside the Mach-NX device in the same fashion as an external logic analyzer permits analysis of board level logic. Reveal access is only available via the Mach-NX JTAG port.

- SRAM Readback

  The JTAG port is the only sysCONFIG port able to directly access the Mach-NX device configuration SRAM.

- Boundary Scan Testability

  Board level connectivity testing performed using IEEE 1149.1 JTAG is a key capability for assuring the quality of assembled printed-circuit-boards. Preserving the Mach-NX JTAG port is vital for boundary scan testability. Lattice provides Boundary Scan Description Language files for the Mach-NX device on the Lattice website.

5.7. TransFR Operation

Mach-NX device, like other Lattice FPGAs, provides for the TransFR™ capability. TransFR is only supported for non-SoC GPIO. The following is an example of how you can update bitstream in Mach-NX device by using the TransFR feature.

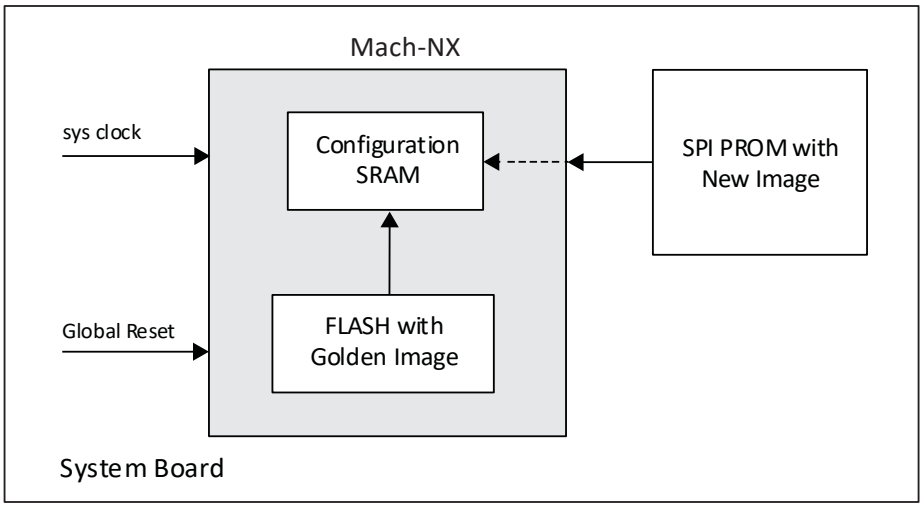
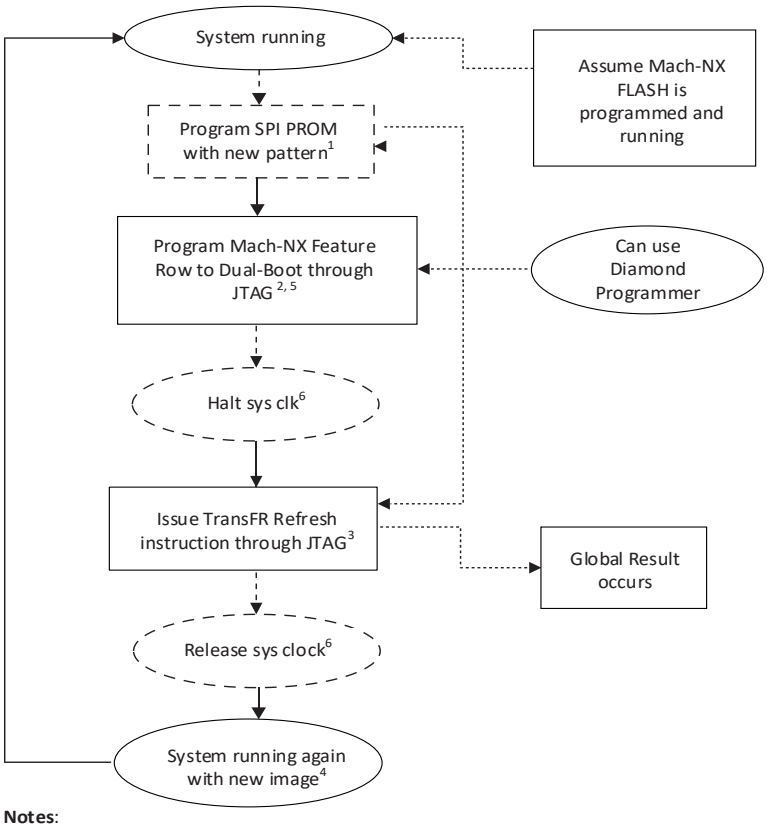


Figure 5.4. Bitstream Update Using TransFR

The example assumes that you have the golden image stored in Flash to initiate the system, and then use SPI PROM as a resource for image updates without disturbing the system. Figure 5.5 shows the process flow for performing this task.



Notes:
1. You can use operations such as *SPI Flash Background Erase, Program, Verify* for this.
2. You can use operations such as *Program Feature Row* for this.
3. You can use operations like *XFLASH TransFR* for this.
4. If new image fails to configure MachNX device, the golden image in FLASH still configures Mach-NX device, so system still runs with original image.
5. Feature Row only needs to be programmed if changes need to be made, for instance, disable or enable JTAG, Slave Port. If no changes need to be made, skip this step.
6. This step is optional.

Figure 5.5. Example Process Flow

Caution when using the above process flow:

Since a Global Reset is triggered during device wake-up after REFRESH instruction is issued, attention needs to be given in designing I/O with following conditions:

- Register output pins
- Impact on the system board level when value changes (may shut off the board, for instance)
- Register is set/reset by global reset

For the I/O in the example above, the state of the I/O is not changed during the TransFR REFRESH, but may change once the device gets into user mode right after the TransFR REFRESH. Following are design tips to avoid this:

- For critical I/O, try not to use global reset.
- For critical I/O, if you have to use global reset, try to use the set/reset option so that when GSR occurs, the state of the I/O pin does not trigger a system crash.

5.8. Password

Mach-NX device supports a password-based security access feature also known as Flash Protect Key. The Flash Protect Key feature provides a method of controlling access to the Configuration and Programming modes of the device. When enabled, the Configuration and Programming edit mode operations including Write, Verify, and Erase operations are allowed only when coupled with a Flash Protect Key, which matches that expected by the device.

The Flash Protect Key feature requires that a device accessing a Mach-NX device through a sysConfig port (JTAG, SSPI, I^2C or AHB-Lite) provides a valid digital Password, also known as the Flash Protect Key, to unlock the device and allow configuration or programming operations to proceed. Without a valid Flash Protect Key, you can perform only rudimentary non-configuration operations such as Read Device ID.

The 128-bit Flash Protect Key is stored in the Feature Row. Three additional Feature Row fuses are specified for enabling the feature: PWD_EN, PWD_ALL, and PWD_UFM.

You can read more about the Password feature in the Using Password Security with Mach-NX Devices technical note.

6. Software Selectable Options

The operation of the Mach-NX device configuration logic is managed by options selected in the Diamond design software. Other FPGAs provide dedicated I/O pins to select the configuration mode. The Mach-NX device uses the non-volatile Feature Row to select how it configures. The Feature Row default state needs to be modified in almost every design. You use the Diamond Spreadsheet View to make the changes to the operation of the Mach-NX Feature Row which alters the operation of the configuration logic.

The configuration logic preferences are accessed using Spreadsheet View. Click on the Global Preferences tab, and look for the sysCONFIG tree. The sysCONFIG section is shown in Figure 6.1. The sysCONFIG preferences are divided into three categories:

- Configuration mode and port related
- Bitstream generation related
- Security related

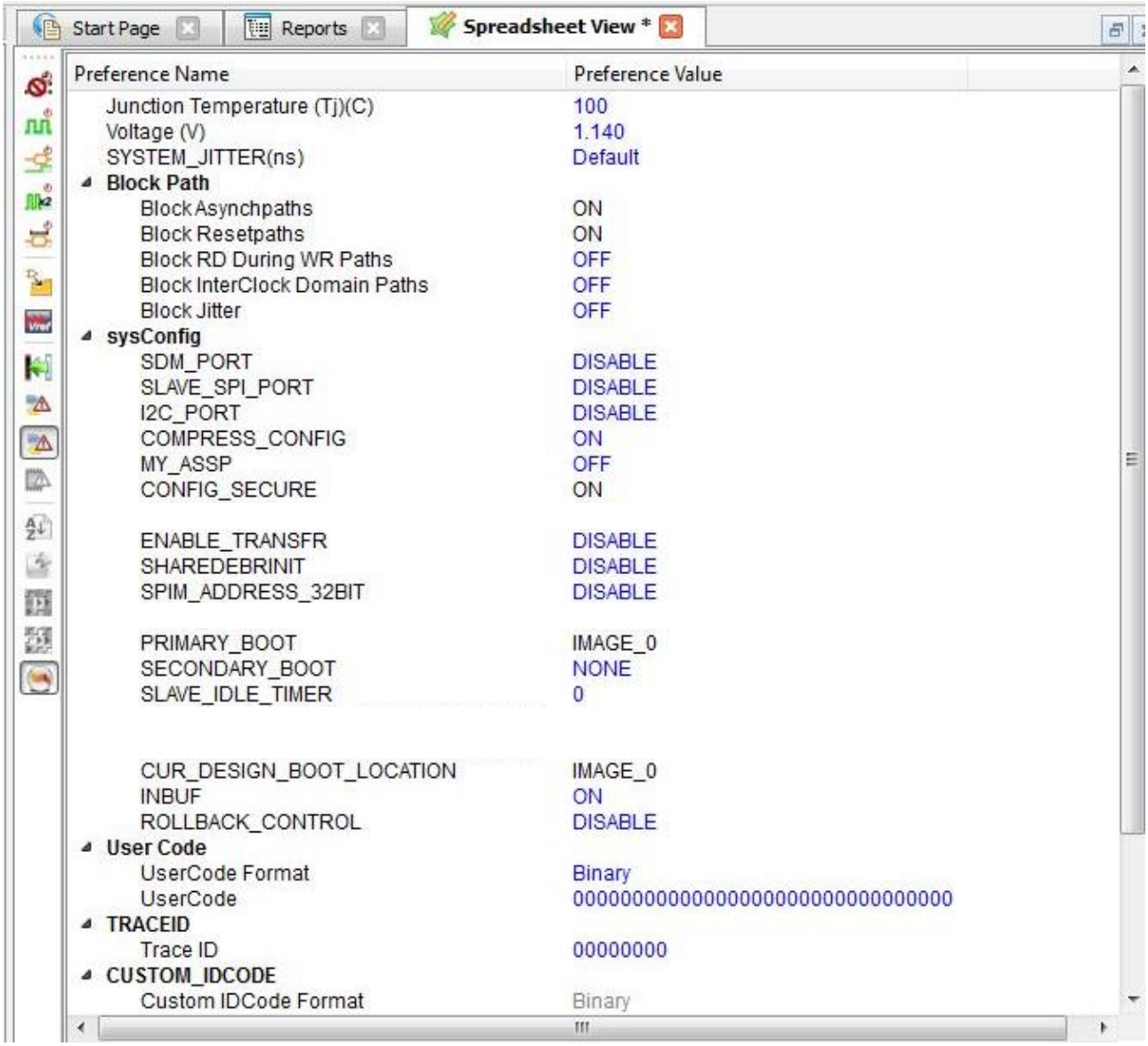


Figure 6.1. sysCONFIG Preferences in Global Preferences Tab, Diamond Spreadsheet View

6.1. Configuration Mode and Port Options

The configuration and port options allow you to decide which configuration ports continue to operate after the Mach-NX device is in user mode. You can also control the availability of status pins, as well as the speed at which configuration data is read from an external PROM. The selections made here are saved in the Feature Row and remain in effect until the Feature Row is erased. The only exception is the MCCLK_FREQ parameter, which is stored in the configuration data.

The configuration and port options can be used in any combination.

Table 6.1. Configuration Mode/Port Options

Option Name	Default Setting	All Settings
SLAVE_SPI_PORT	DISABLE	ENABLE, DISABLE
I2C_PORT	DISABLE	ENABLE, DISABLE
SDM_PORT	DISABLE	DISABLE, PROGRAMN, PROGRAMN_DONE, PROGRAMN_DONE_INITN

6.1.1. JTAG Port

The JTAG_PORT is always enabled

6.1.2. Slave SPI Port

The SLAVE_SPI_PORT allows you to preserve the Slave SPI configuration port after the Mach-NX device enters user mode. There are two states to which the SLAVE_SPI_PORT preference can be set:

- ENABLE – This setting preserves the SPI port I/O when the Mach-NX device is in user mode. When the pins are preserved, an external SPI master controller can interact with the configuration logic. The preference also prevents you from over-assigning I/O to the port pins.
- DISABLE – This setting disconnects the SPI port pins from the configuration logic. By itself it does not make the port pins general purpose I/O. Both the SLAVE_SPI_PORT and MASTER_SPI_PORT must be in the DISABLE state for the SPI port pins to be general purpose I/O.

The SLAVE_SPI_PORT can be enabled at the same time as the MASTER_SPI_PORT. It is necessary to guarantee that the internal SPI master controller not perform SPI transactions at the same time as an external SPI master. It is your responsibility to prevent two SPI masters from operating simultaneously.

6.1.3. I^2C Port

The I2C_PORT allows you to preserve the I^2C configuration port after the Mach-NX device enters user mode. There are two states to which the I2C_PORT preference can be set:

- ENABLE – This setting preserves the I^2C port I/O when the Mach-NX device is in user mode. When the pins are preserved, and the EFB is instantiated with wb_clk_i input connected to a valid clock source of at least 7.5 times the I^2C bus rate, an external I^2C master controller can interact with the configuration logic. The preference also prevents you from over-assigning I/O to the port pins.
- DISABLE – This setting disconnects the I^2C port pins from the configuration logic. The port pins become general purpose I/O.

To use the primary and secondary I^2C controllers in the EFB, the I2C_PORT must be in the ENABLE state.

6.1.4. SDM Port

The SDM_PORT allows you to select the programming status pins after the Mach-NX device enters user mode. There are six states to which the SDM_PORT preference can be set:

- DISABLE – This setting causes the PROGRAMN, DONE, and INITN status pins to become general purpose I/O.
- PROGRAM – This setting preserves the PROGRAMN pin when the Mach-NX device is in user mode. Asserting this pin active low causes the Mach-NX device to reconfigure. The DONE and INITN pins are general purpose I/O.

- DONE – This setting preserves the DONE pin when the Mach-NX device is in user mode. The PROGRAMN and INITN pins are general purpose I/O.
- INITN – This setting preserves the INITN pin when the Mach-NX device is in user mode. The PROGRAMN and DONE pins are general purpose I/O.
- PROGRAM_DONE – This setting preserves the PROGRAMN and DONE pins when the Mach-NX device enters user mode. INITN is a general purpose I/O.
- PROGRAM_DONE_INITN – This setting preservers PROGRAM, DONE, and INITN in user mode.

Lattice recommends setting the SDM_PORT to PROGRAMN when using Master SPI or Dual Boot Configuration Mode. The PROGRAMN pin is the only way to perform a *warm* reconfiguration of the Mach-NX device, unless another configuration port is available to transmit a REFRESH command.

6.1.5. ENABLE_TRANSFR

The TransFR function used by the Mach-NX device requires the configuration data loaded into the configuration SRAM, and any future configuration data file loaded into the internal Flash memory have the ENABLE_TRANSFR set to the ENABLE state. See the TransFR Operation section for more information about using TransFR with the Mach-NX device.

6.1.6. SLAVE_IDLE_TIMER

When downloading the bitstream from the slave port, SSPI or I^2C, the Mach-NX device configuration module starts running a timer on configuration clock domain. This timer is enabled to recover from any hang-up caused by the external SPI master or I^2C master. When the timer expires, the configuration engine goes back to the reset state. It allows the configuration engine to wait and receive the new coming bitstream from any slave port. There are 16 options for the setting of this timer. The default option is for the infinite timer, or the disabled timer. The other 15 options define the timer from 10 ms to 128 s.

Table 6.2. SLAVE_IDLE_TIMER Values

Option	Timeout Period	Option	Timeout Period
0	Infinite (disabled)	8	1000 ms
1	128000 ms	9	640 ms
2	64000 ms	10	320 ms
3	32000 ms	11	160 ms
4	16000 ms	12	80 ms
5	8000 ms	13	40 ms
6	4000 ms	14	20 ms
7	2000 ms	15	10 ms

6.2. Bitstream Generation Options

The Bitstream Generation options allow you to decide how the Diamond development tools create the configuration data for the Mach-NX device. The CONFIGURATION, USERCODE, CUSTOM_IDCODE, and SHAREDEBRINIT settings are saved in the Feature Row and remain in effect until the Feature Row is erased. The other options allow you to control the JEDEC and BIT files that are generated by Diamond.

6.2.1. COMPRESS_CONFIG

The COMPRESS_CONFIG preference alters the way JEDEC and BIT files are generated. The COMPRESS_CONFIG default setting is ON.

JEDEC files, when they are built, are always compressed. The configuration time is slightly reduced when reading configuration data from the external PROM and the Diamond tool creates a JEDEC file you can program into the internal memory.

6.2.2. PRIMARY_BOOT

The PRIMARY_BOOT preference allows you to control which configuration memory sector is used for the single image configuration mode or the first boot in the Dual Boot Configuration Mode. The PRIMARY_BOOT preference has five possible settings:

- IMAGE_0 – This preference is the default setting. It uses CFG0 or CFG0 plus UFM0 sectors for the primary boot. The CONFIGURATION preference determines whether or not UFM0 is used to store the configuration image.
- IMAGE_1 – This preference uses CFG1 or CFG1 plus UFM1 sectors for the primary boot. The CONFIGURATION preference determines whether or not UFM1 is used to store the configuration image.
- LATEST – This preference is one of the settings for the version-based Dual Boot Configuration Mode. The configuration starts from one of the CFG0 and CFG1 that has the later configuration image or the larger version number first. If the process fails, the former configuration image in the other internal Flash sector is loaded.
- FORMER – This preference is one of the settings for the version-based Dual Boot Configuration Mode. The configuration starts from one of the CFG0 and CFG1 that has the former configuration image or the smaller version number first. If the process fails, the later configuration image in the other internal Flash sector is loaded.

6.2.3. SECONDARY_BOOT

The SECONDRY_BOOT preference allows you to control which configuration memory sector is used for the secondary boot in the Dual Boot Configuration Mode. The SECONDARY_BOOT preference has six possible settings:

- NONE – This preference is the default setting. It is used to select the single image configuration mode.
- IMAGE_0 – This preference is one of the settings for the golden image Dual Boot Configuration Mode. It uses CFG0 or CFG0 plus UFM0 sectors to store the golden configuration image for the secondary boot. The CONFIGURATION preference determines whether or not UFM0 is used to store the configuration image.
- IMAGE_1 – This preference is one of the settings for the golden image Dual Boot Configuration Mode. It uses CFG1 or CFG1 plus UFM1 sectors to store the golden configuration image for the secondary boot. The CONFIGURATION preference determines whether or not UFM1 is used to store the configuration image.
- LATEST – This preference is one of the settings for the version-based Dual Boot Configuration Mode. The configuration starts from one of the CFG0 and CFG1 that has the former configuration image or the smaller version number first. If the process fails, the later configuration image in the other internal Flash sector is loaded.
- FORMER – This preference is one of the settings for the version-based Dual Boot Configuration Mode. The configuration starts from one of the CFG0 and CFG1 that has the later configuration image or the larger version number first. If the process fails, the former configuration image in the other internal Flash sector is loaded.

6.2.4. USERCODE

The Mach-NX device configuration Flash sector contains a 32-bit register for storing a user-defined value. The USERCODE is also stored in the generated bitstream for redundancy. The default value stored in the register is 0x00000000. Using the USERCODE preference, you can assign any value to the register you desire. Suggested uses include the configuration data version number, a manufacturing ID code, date of assembly, or the JEDEC file checksum.

The format of the USERCODE field is controlled using the USERCODE_FORMAT preference. Data entry can be performed in either Binary, Hex, or ASCII formats.

6.2.5. USERCODE_FORMAT

The USERCODE_FORMAT preference selects the format for the data field used to assign a value in the USERCODE preference. The USERCODE_FORMAT has three options:

- Binary – USERCODE is set using 32 ‘1’ or ‘0’ characters.
- Hex – USERCODE is set using eight hexadecimal digits that is 0-9A-F.
- ASCII – USERCODE is set using up to four ASCII characters.

6.2.6. CUSTOM_IDCODE

The CUSTOM_IDCODE preference is used to assign a 32-bit register that resides in the Feature Row. The CUSTOM_IDCODE field is only active when the MY_ASSP preference is in the ON state. The value assigned can be entered in binary or hexadecimal, according to the CUSTOM_IDCODE_FORMAT preference. See the MY_ASSP section for more information about how to assign a value to the CUSTOM_IDCODE preference.

6.2.7. CUSTOM_IDCODE_FORMAT

The CUSTOM_IDCODE_FORMAT preference selects the format for the data field used to assign a value in the CUSTOM_IDCODE preference. The CUSTOM_IDCODE_FORMAT has two options:

- Binary – CUSTOM_IDCODE is set using 32 '1' or '0' characters.
- Hex – CUSTOM_IDCODE is set using eight hexadecimal digits that is 0-9A-F.

6.2.8. SHAREDEBRINIT

When set to ENABLE, this preference allows one copy of a unique memory initialization file to be stored in the internal memory. This copy of the initialization values can be shared among multiple EBRs. Doing so reduces the bitstream size of the design and saves internal memory space for other applications.

6.2.9. CUR_DESIGN_BOOT_LOCATION

The CUR_DESIGN_BOOT_LOCATION preference is used to specify the location of the configuration image generated by the current Diamond project in Dual Boot applications. When set to IMAGE_0, it is targeted to the CFG0 space of the internal Flash. When set to IMAGE_1, it is targeted to the CFG1 space of the internal Flash.

6.2.10. ROLLBACK_CONTROL

When the ROLLBACK_CONTROL preference is set to ENABLE, the Mach-NX device provides a mechanism to prevent a bitstream from updating to an earlier version. The version number is stored in USERCODE and is treated as 32-bit unsigned integer in this mechanism. In addition, this mechanism is applicable only to the dual configuration mode, except the boot sequence = EXT_EXT. Authentication is required to be enabled. The flow of the mechanism is different for the various boot sequences.

For the dual boot sequences, using both CFG0 and CFG1, after programming the bitstream to CFG0 or CFG1, you need to send the LSC_PROG_AUTH_DONE command to try to program the Pre-authentication Done bit or AUTH_DONE bit. The configuration engine then authenticates this bitstream and checks to see whether its version number is newer than the one in the other internal memory that contains a bitstream. If both authentication and version checking pass, the Pre-authentication Done bit is programmed. Otherwise, it is not be programmed. You can run Flash Programming Mode > FLASH Version Rollback Protect in Diamond Programmer to program both the bitstream and the Pre-authentication Done bit. When booting, the configuration engine check the Pre-authentication Done bit when the preference is enabled. It continues to boot only when the bit is programmed.

When booting from the bitstream in the internal memory, the Pre-authentication Done bit is checked. Only the programmed bit allows continuous booting. Hence, you need to program it by running Advanced Security Keys Programming > Security Program Auth Done FlashA or Security Program Auth Done FlashB in Diamond Programmer after programming its bitstream.

6.3. Security Options

The Security options allow you to select from a range of options for tracking or securing the Mach-NX device. Table 6.3 provides a summary of these options.

Table 6.3. Security Options

Option Name	Default Setting	All Settings
TRACEID	<all zero>	8-bit arbitrary
MY_ASSP	OFF	OFF, ON
CONFIG_SECURE	OFF	OFF, ON

6.3.1. TRACEID

TraceID stamps each Mach-NX device with a unique 64-bit ID. No two Mach-NX devices have the same TraceID value even when they are loaded with the same configuration data. This differs from a USERCODE which is present in the configuration data. Every device that receives the configuration data using a USERCODE receives the same USERCODE value.

The TraceID is 64 bits long with the least significant 56 bits being immutable data. The 56 bits are a combination of the wafer lot, the wafer number, and the X/Y coordinates locating the die on the wafer. The most significant eight bits are provided by you and are stored in the Feature Row. The TraceID is changed using the Diamond Spreadsheet View. You enter a unique 8-bit binary value in the TraceID field and generate configuration data.

You can read more about the TraceID feature in Using TraceID (FPGA-TN-02084).

6.3.2. MY_ASSP

Every Lattice device has its own identification code identifying the device family, device density, and other parameters, for example voltage, device stepping. The code is accessible from any Mach-NX device configuration port. The value stored in the IDCODE register allows you to uniquely identify a Lattice device.

The MY_ASSP preference permits you to change the value returned when the IDCODE is read from the FPGA. Set the MY_ASSP preference to the ON state. Turning MY_ASSP ON enables the CUSTOM_IDCODE preference.

6.3.3. CUSTOM_IDCODE

The CUSTOM_IDCODE is the value you assign to override the default IDCODE in the Mach-NX device. You are only allowed to enter a 32-bit hexadecimal or binary value when the MY_ASSP preference is ON.

Overriding the IDCODE prevents the Lattice programming software from being able to identify the Mach-NX device, and as a result, prevents Diamond Programmer from directly programming the Mach-NX device. It is necessary to migrate to generating Serial Vector Format (SVF) files to program MY_ASSP enabled Mach-NX devices.

6.3.4. CONFIG_SECURE

When this preference is set to ON, the read-back of the SRAM memory and the Flash memory are blocked. The Mach-NX device cannot be read back, nor can it be programmed without erasing. The device must be erased to reset the security setting. The CONFIG_SECURE fuse and the Flash are erased in tandem. Once the security fuses are reset, the device can be programmed again.

7. Device Wake-up Sequence

When configuration is completed, that is, the SRAM is loaded, the device wakes up in a predictable fashion. If the Mach-NX device is the only device in the chain, or the last device in a chain, the wake-up process should be initiated by the completion of configuration. Once configuration is completed, the internal DONE bit is set and then the wake-up process begins. Figure 7.1 shows the wake-up sequence using the internal clock.

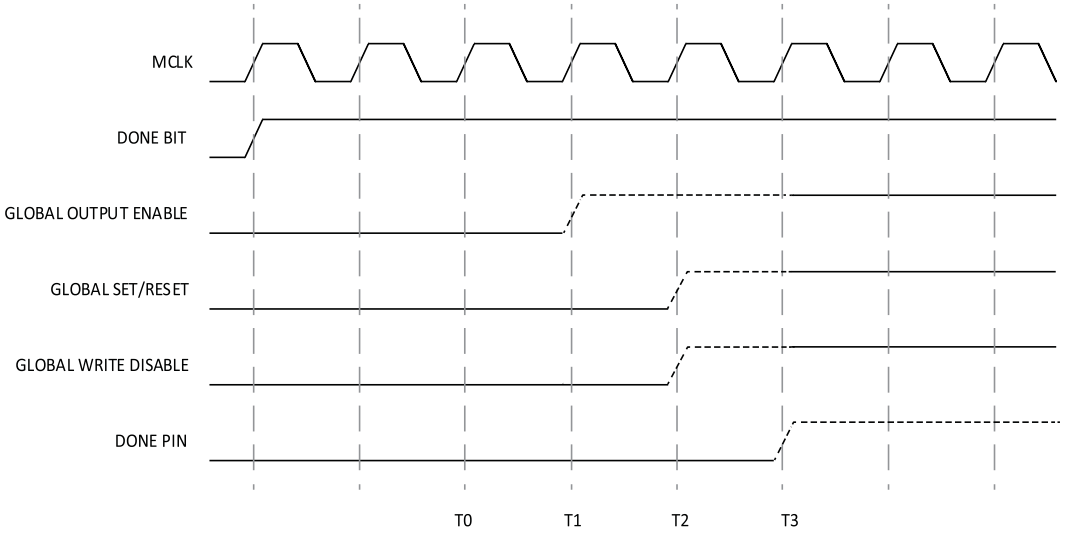


Figure 7.1. Wake-up Sequence Using Internal Clock

7.1. Wake-up Signals

Three internal signals, GSR, GWDIS, and GOE, determine the wake-up sequence.

- GSR – GSR is used to set and reset the core of the device. GSR is asserted low during configuration and de-asserted high in the wake-up sequence.
- GWDIS – When the GWDIS signal is low, it safeguards the integrity of the RAM Blocks and LUTs in the device. This signal is low before the device wakes up. This control signal does not control the primary input pin to the device but controls specific control ports of EBR and LUTs.
- GOE – When low, GOE prevents the device I/O buffers from driving the pins. The GOE only controls output pins. Once the internal DONE is asserted, the Mach-NX device responds to input data.

  When high, the DONE pin indicates that configuration is complete and that no errors are detected.

7.2. Wake-up Clock Selection

The clock source used to complete the four state transitions in the wake-up sequence is user-selectable. Once the Mach-NX device is configured, it enters the wake-up state, which is the transition between the configuration mode and user mode. This sequence is synchronized to a clock source, which defaults to MCLK/CCLK when sysCONFIG is used, or TCK when JTAG is used.

You can change the clock used by instantiating the START macro in your Verilog or VHDL file. The clock must be supplied on an external input pin, because the Mach-NX device does not begin internal operations until the wake-up sequence is completed. There is no external indication the device is ready to perform the last four state transitions. You must either provide a free running clock frequency, or you must wait until the device is guaranteed to be ready to wake up. Using the START macro provides another mechanism for holding off configuring one or more programmable devices and then starting them synchronously.

Verilog

```
module START (STARTCLK);
input STARTCLK;
endmodule

START u1 (.STARTCLK(<clock_name>)) /* synthesis syn_noprune=1 */;
```

VHDL

```
COMPONENT START
     PORT(
           STARTCLK      :     IN STD_ULOGIC
         );
END COMPONENT;
attribute syn_noprune: boolean;
attribute syn_noprune of START: component is true;

begin
u1: START port map (STARTCLK =><clock name>);
```

8. Advanced Configuration Information

8.1. Flash Programming

The Mach-NX device internal Flash is the heart of the FPGA configuration system. It is flexible, allowing you to store the FPGA configuration data, as well as storing design specific data in the internal memory. It is also a resource that uses a precise erasing and programming sequence. Lattice provides several methods for programming the Mach-NX Flash:

- JTAG or Slave SPI programming
- VMEmbedded: C source for use with an embedded microprocessor controlling the JTAG port
- SSPIEmbedded: C source for use with an embedded microprocessor controlling the SSPI port
- Custom: The information in this section, and information from the Mach-NX SFB Hardware Usage Guide (FPGA-TN-02222), permits creation of a custom solution.

The Flash space can be accessed by the JTAG, I^2C, and SPI ports. These configuration ports may use offline or transparent modes to erase, program, and verify the Mach-NX Flash resources. The AHB-Lite interface is only permitted to use transparent programming operations. The sequence and timing of the commands presented to the configuration logic are identical across all of the configuration ports. There are slight differences due to communication protocol standards when transmitting commands and data. The command and timing flow common to all configuration ports is described first. Protocol variances are described afterward.

Each Mach-NX device contains a certain quantity of internal memory. The amount of memory depends on the Mach-NX device density. Refer to Table 9.1 and Table 10.1 in Mach-NX SFB Hardware Usage Guide (FPGA-TN-02222) for the number of internal memory pages available for each Mach-NX device density.

For programming the Mach-NX device internal Flash memory using Lattice Diamond Programmer (Rev 3.12 or later) through HW-USBN-2B cable, refer to the example shown in the Mach-NX Internal Flash Programming section.

8.2. Mach-NX Device JEDEC File Format

All Lattice non-volatile devices support JEDEC files. Utilities are available in the Deployment Tool software for converting the JEDEC file into other programming file formats, such as STAPL, SVF, or bitstream no matter in hex or binary format. Relevant details about the JEDEC file are provided in Table 8.1 or completeness.

Table 8.1. Mach-NX Device JEDEC File Format

JEDEC Field	Syntax	Description
Don't Care	My design	Characters appearing before the ^B character are don't care. All character sets or internal language can be used here except ^B.
Start-of-text	^B	^B (Control-B 0x02) marks the beginning of the JEDEC file. Only ASCII characters are legal after ^B. The character * is the delimiter to mark the ending of a JEDEC field. The CR and LF are treated as regular white spaces and have no delimiter function in a JEDEC file.
Header	My design	The first field is the header, which does not have an identifier to indicate its start. Only ASCII characters are legal after ^B. The header is terminated by an asterisk character *.
Field Terminator	*	Each field in the JEDEC file is terminated with an asterisk.
Note (Comment)	NOTE my design	The key word N marks the beginning of the comment. It can appear anywhere in the JEDEC file. Lattice JEDEC files add OTE to the N keyword to make it a more meaningful word NOTE.
Fuse Count	QF3627736	The keyword QF identifies the total real fuse count of the device.
Default Fuse State	F0 or F1	The keyword F identifies the fuse state of those fuses not included in the link field. F0 = fill them with zeroes (0), F1 = fill them with ones (1). It is defined for the purpose of reducing JEDEC file size. It has no meaning in Lattice JEDEC file. Lattice recommends using compression to reduce file size instead.
Security Setting	G0 or G1	JEDEC standard defines G<0,1> to program security <0=no, 1=yes>

JEDEC Field	Syntax	Description
Link Field	L0000000 101011…100011 …… 111111…101100 110 101011…100011 …… 111111…101100 110 …… …… 101011…100011 …… 111111…101100 110*	The keyword L identifies the first fuse address of the fuse pattern that follows after the white space. The number of digit shown following the L keyword must be the same as that on the QF field. In this example, QF3627736 has seven digits, thus L0000000 should have seven zeroes. The fuse address traditionally starts counting from 0. The link field is the most critical portion of the JEDEC file where the programming pattern is stored. The programming data is written into this field in the manner mirroring exactly the fuse array layout of the silicon physically. Row address is written from top to bottom in ascending order: Top = Row 0, Bottom = Last Row. The column address is written from left to right in ascending order: Left most = bit 0, Right most = last bit. Row 0 is selected first by the INIT_ADDRESS command. The first bit to shift into the device is bit 0 for programming. The first to shift out from the device is also bit 0 when verify. The end of the configuration data is marked by “NOTE END CONFIG DATA*”. It is not necessary to program any page data containing all ‘0’ values. UFM pages, if present in the JEDEC, are preceded by a “NOTE USER MEMORY DATA UFM0*” line or a “NOTE USER MEMORY DATA UFM1*” line. If the JEDEC file is encrypted, all the data in the link field are encrypted. The column size increases accordingly to include filler bits to make the column size packet, 128 bits or 16 bytes per packet, bounded.
Fuse Checksum	CC1B9	The checksum of all the fuses = Fuse count. The fuse state of all the fuses are found from the Link field. If it is not specified in the link field, then use the Default Fuse State in their places. If the JEDEC file is encrypted, the fuse checksum is calculated after encryption. The fuse checksum prior to encryption are found in one of the comments.
U Field	UA Home	This is the place to store the 32-bit USERCODE. The 32-bit USERCODE can be expressed in UA = ASCII, UH = ASCII Hex, U = Binary. Lattice enhanced this field for storing the CRC value of encrypted JEDEC.
End-of-text	^C	^C (CTLC) marks the ending of the JEDEC file.
Transmission Checksum	ABCD	This is the checksum of the whole file starting from ^B to ^C. All characters and white space, including the ^B and ^C, are included in the checksum calculation.

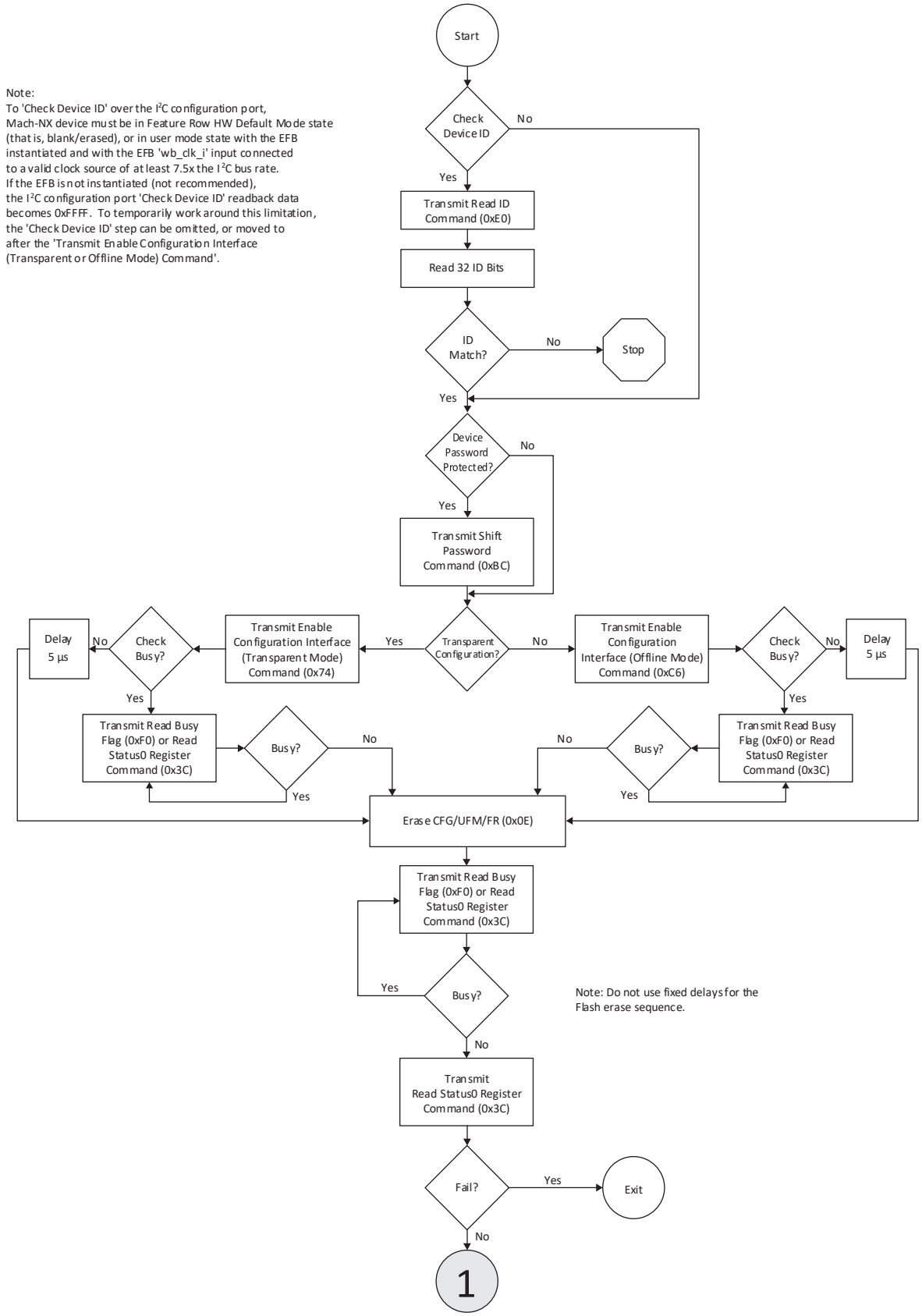
An example of a Mach-NX device JEDEC file is shown in Figure 8.1.

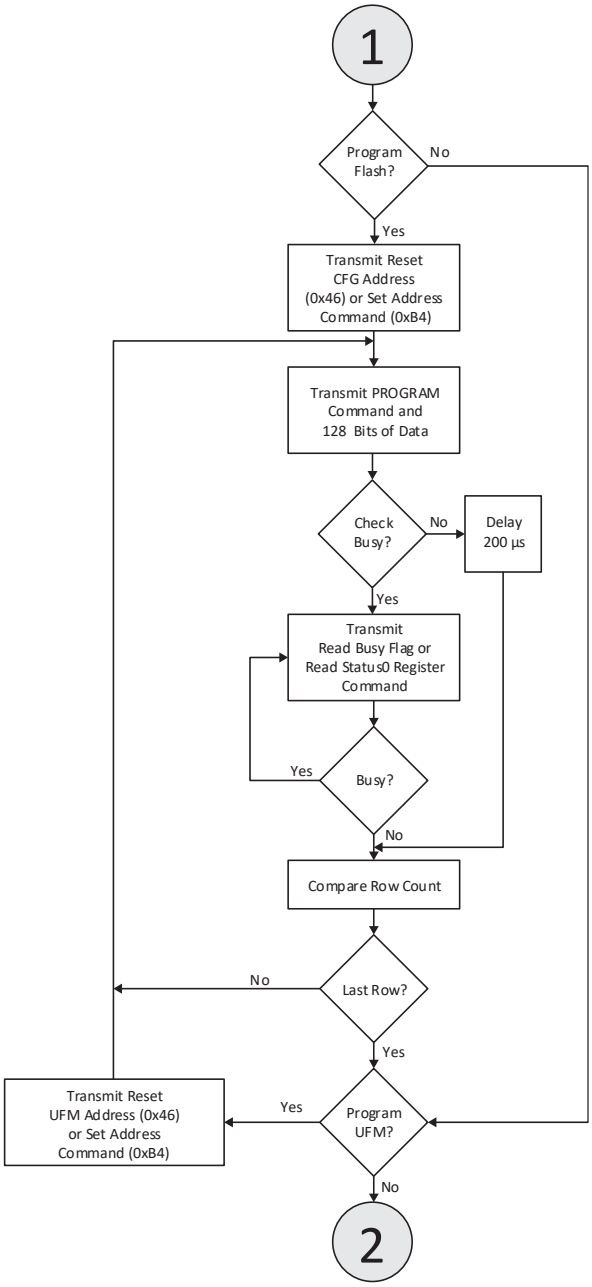
```
*
NOTE Diamond (64-bit) 3.11.0.187.1 JEDEC Compatible Fuse File.*
NOTE Copyright (C), 1992-2010, Lattice Semiconductor Corporation.*
NOTE All Rights Reserved.*
NOTE DATE CREATED:     Tue Jun 26 09:24:31 2018*
NOTE DESIGN NAME:      SimpleDesign_impl1.ncd*
NOTE DEVICE NAME:      LCMXO3D-6900HC-5TQFP144*
NOTE JEDEC FILE STATUS: Advanced   Version 1.6*
NOTE PIN ASSIGNMENTS*
NOTE PINS DOUT[3] : 82 : out*
NOTE PINS DOUT[4] : 83 : out*
NOTE PINS DOUT[5] : 91 : out*
NOTE PINS DOUT[6] : 77 : out*
NOTE PINS DOUT[7] : 95 : out*
NOTE PINS DOUT[2] : 84 : out*
NOTE PINS DOUT[1] : 94 : out*
…
NOTE PINS wb_dat_i[1] : 13 : in*
NOTE PINS wb_dat_i[0] : 1 : in*
QP144*
QF2063872*
G0*
F0*
L0000000
11111111111111111111111111111111111111111111111111111011110110110011111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111001110110000000000000000000000000111000100000000
000000000000000000010001001011100011000001000011000000100000000000000000000000000000101000000101001001000001010000000001010100 1000
0011000000000000110000100001000000000000100000100000000000100010000000000000110000000000100010000000000000000000000000000000000000
000000000000000000100011000000000000000000000000000010001010000000000000000000000000011111111111111111111111111111111111111101000110
0000000000000000000000000101110001110000000000001101111000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
…
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000101011100000110111111111111111111111111111111111111111111111111111111111111111000010100000000000000000000000000011001010
1111111010111101010111110001100010001001100100011000000000000000000000000001100010100000000000000000000000000011111111111111111111111
*
NOTE EBR_INIT DATA*
L275072
11111111111111111111111111111111111111011000000000000000000000000000000000000000000000000010000000000000101100100001000000000000010000000
000000000000000001000000010000000011000000100000000101000000110000000111000000100000000100100000101000000101100000110000000110100
0001110000000111100001000000000100010000100100000100110000101000000010101000010110000010111000011000000011001000011010000011011000 0
…
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0010001100000000000000000000000011000101000000000000000000000000000101111000000000000000000000000001111111111111111111111111111111
*
NOTE END CONFIG DATA*
L284544
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
…
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
*
NOTE USER MEMORY DATA UFM0*
L2062592
00000000000000001000000010000000110000001000000001010000001100000011100000100000000100100001010000001011000011000000110100000111000000 1111
0001000000001000100010010000100110001010000001010100010110000101110001100000011001000110100001101100011100000111010001111000011111
00000000000000001000000010000000110000001000000001010000001100000011100000100000000100100001010000001011000011000000110100000111000000 1111
0001000000001000100010010000100110001010000001010100010110000101110001100000011001000110100001101100011100000111010001111000011111
00000000000000001000000010000000110000001000000001010000001100000011100000100000000100100001010000001011000011000000110100000111000000 1111
0001000000001000100010010000100110001010000001010100010110000101110001100000011001000110100001101100011100000111010001111000011111
00000000000000001000000010000000110000001000000001010000001100000011100000100000000100100001010000001011000011000000110100000111000000 1111
0001000000001000100010010000100110001010000001010100010110000101110001100000011001000110100001101100011100000111010001111000011111
00000000000000001000000010000000110000001000000001010000001100000011100000100000000100100001010000001011000011000000110100000111000000 1111
0001000000001000100010010000100110001010000001010100010110000101110001100000011001000110100001101100011100000111010001111000000000
*
C83B5*
NOTE User Electronic Signature Data*
UHCAFEBABE*
A6FE
```

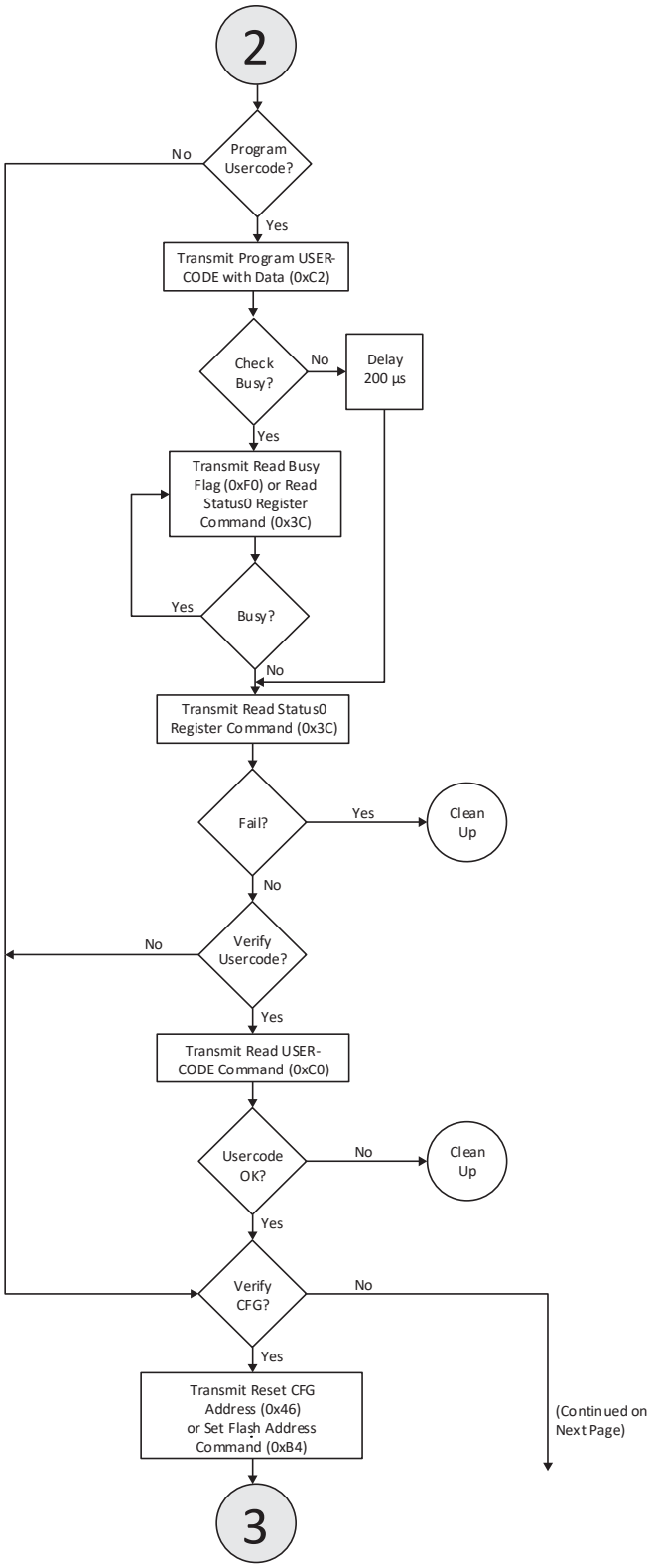
Figure 8.1. JEDEC File Example

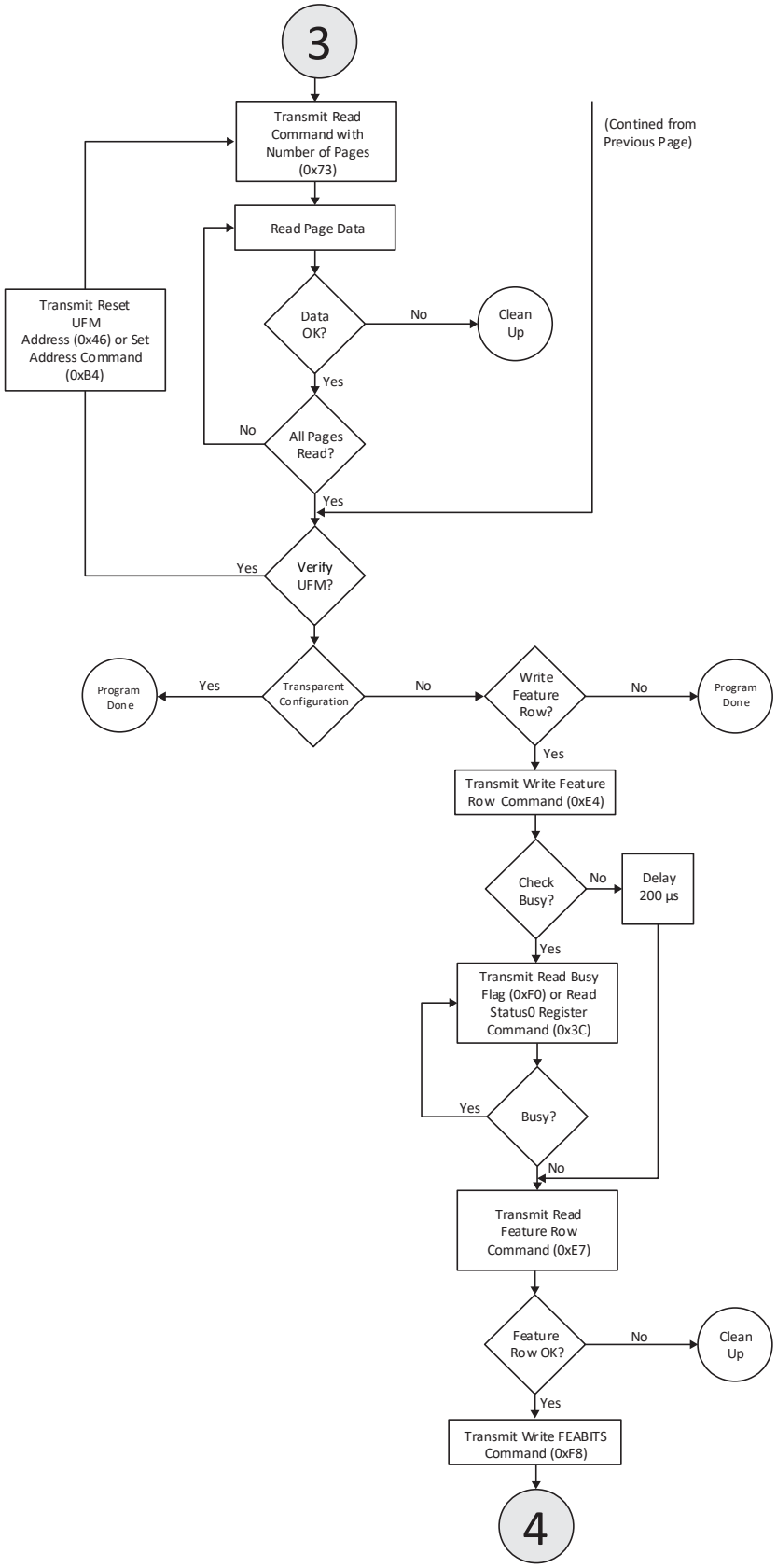
8.3. Mach-NX Flash Programming Flow

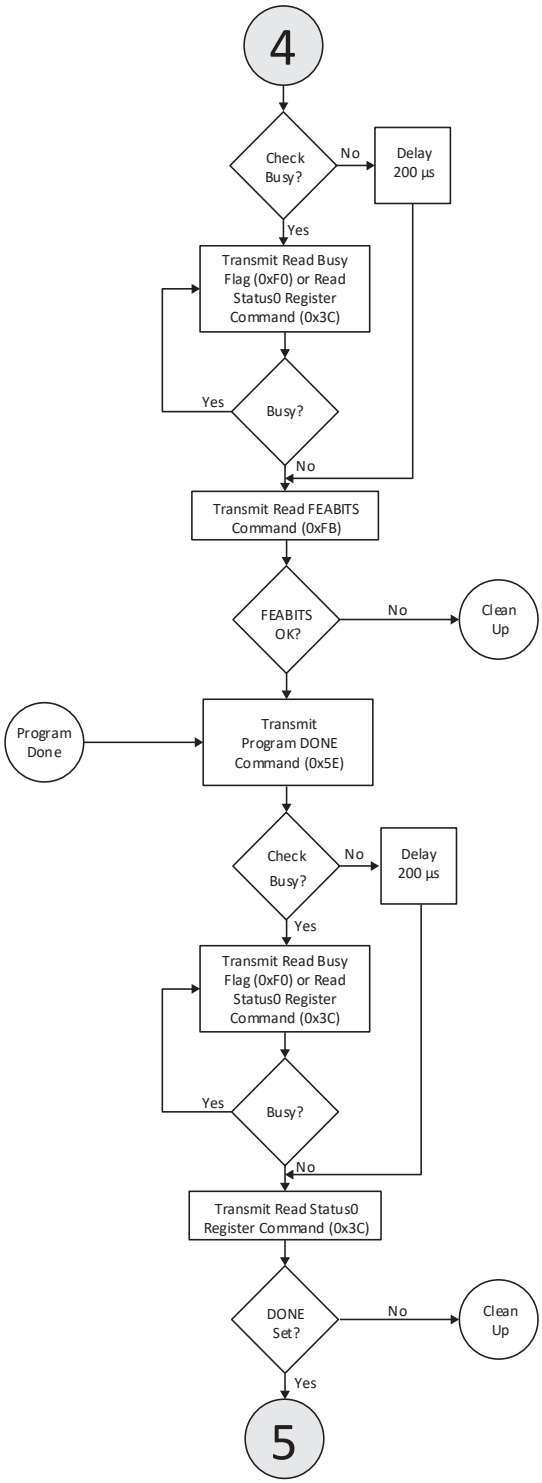
The Mach-NX Flash memory erasing and programming requires a specific set of steps and timing. The flowchart in this section describes the command sequences and the timing required for successful Flash programming. The commands and timing are common among all of the configuration ports. There are some minor variations in the protocol, but not the timing, based on the configuration port used.

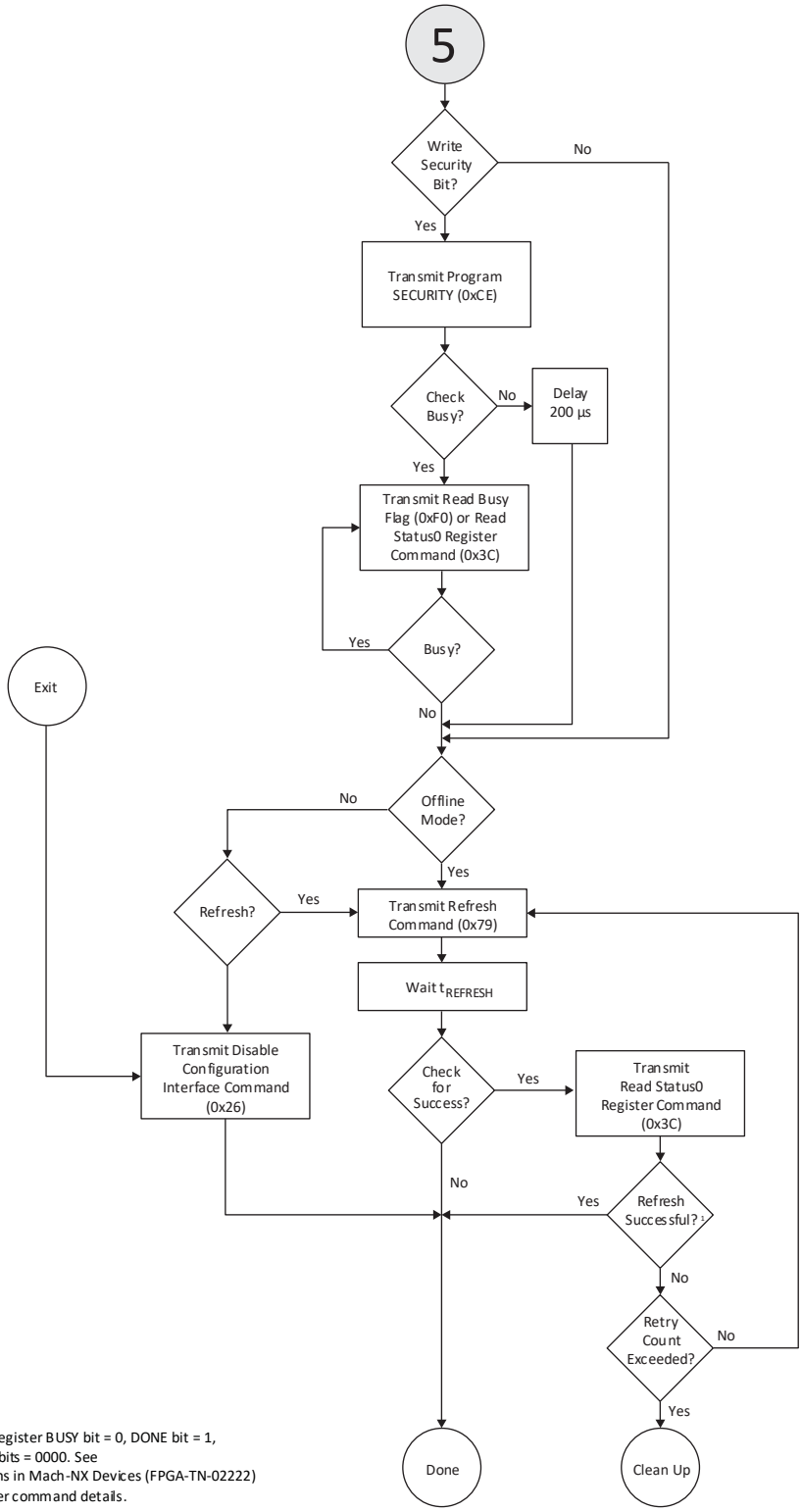












Notes:
Refresh is successful if Status0 Register BUSY bit = 0, DONE bit = 1, and Configuration Check Status bits = 0000. See Using Hardened Control Functions in Mach-NX Devices (FPGA-TN-02222) for complete Read Status Register command details.

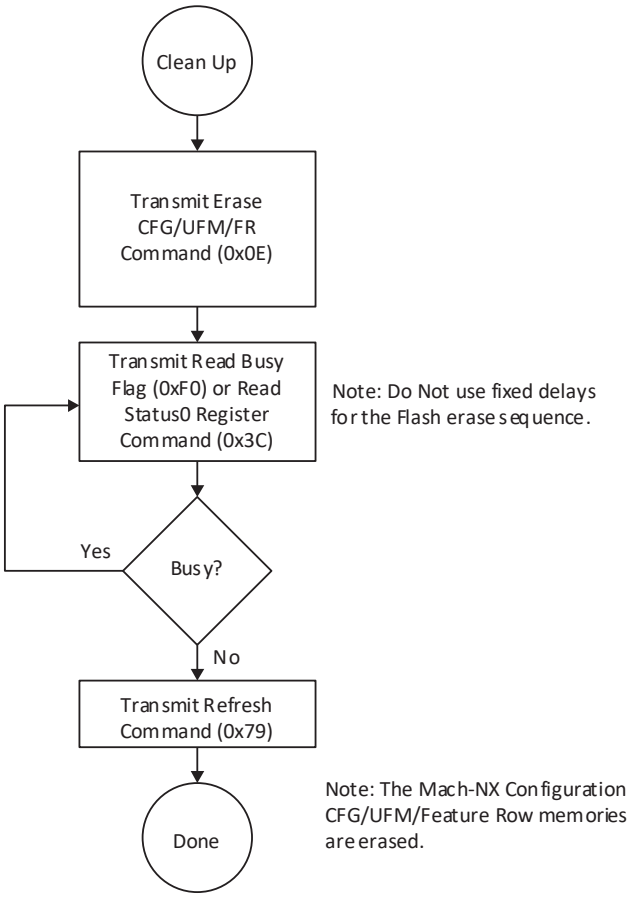


Figure 8.2. Mach-NX Flash Memory Programming Flow

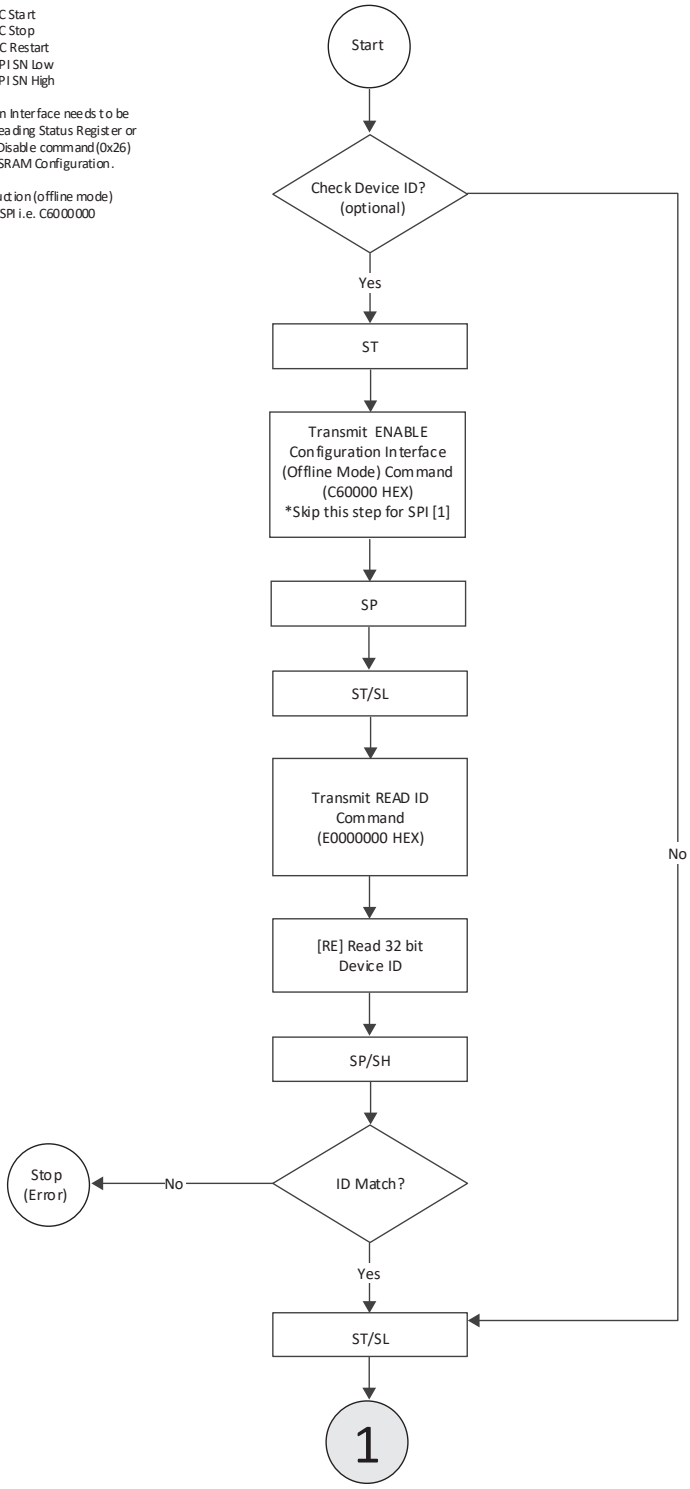
8.4. Mach-NX Slave SPI/I²C SRAM Configuration Flow

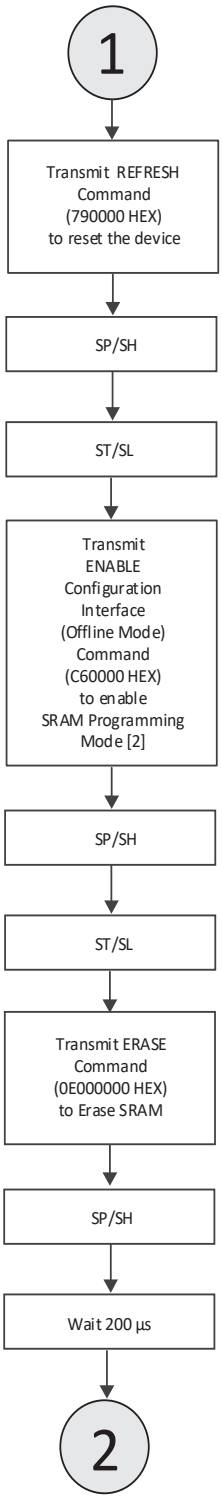
Mach-NX Slave SPI/I²C SRAM configuration requires a specific set of steps and timing. The flow chart in this section describes the command sequences and the timing required for successful SSPI/I²C SRAM configuration.

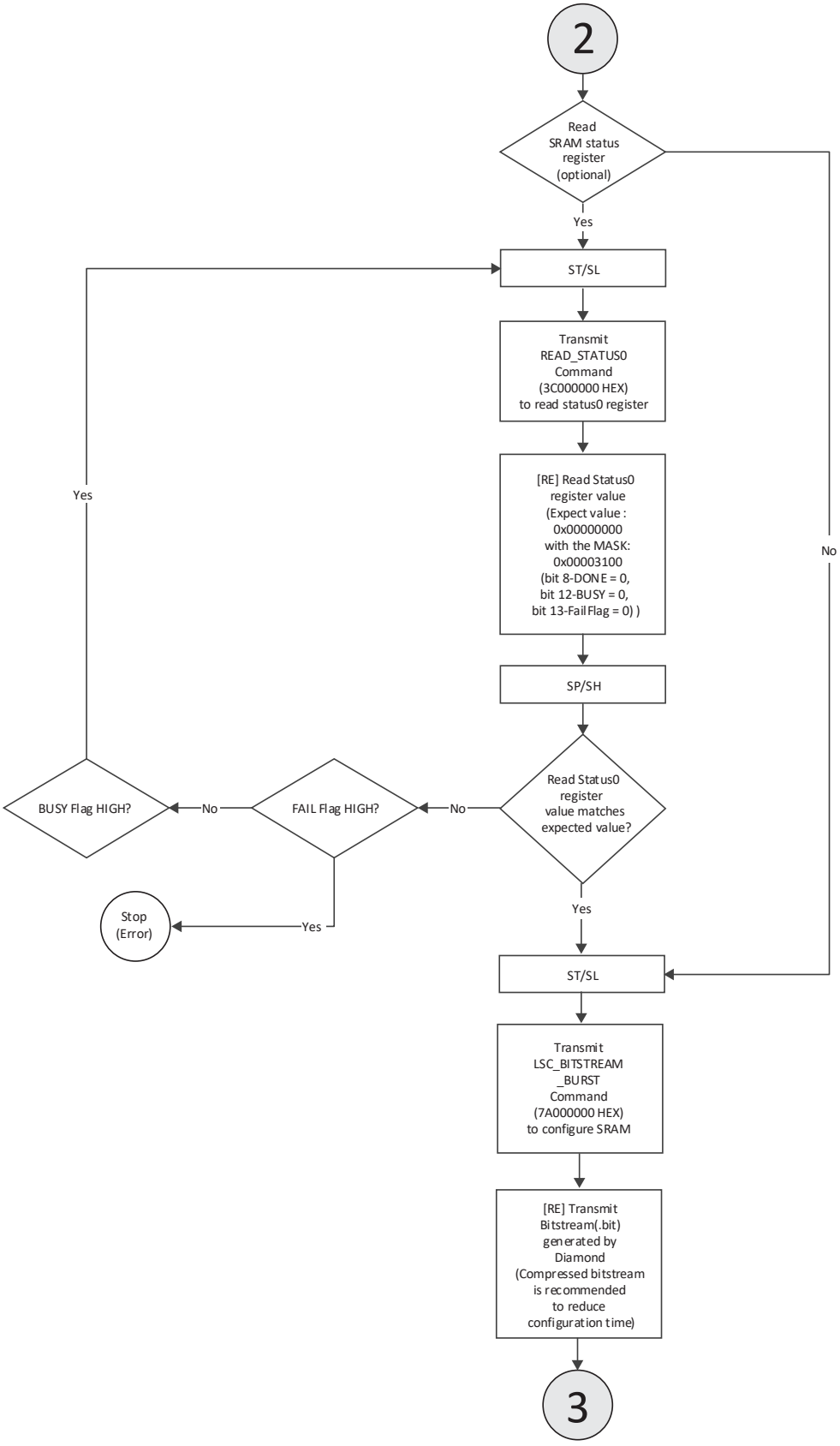
Note: ST -> I²C Start
SP -> I²C Stop
RE -> I²C Restart
SL -> SPI SN Low
SH -> SPI SN High

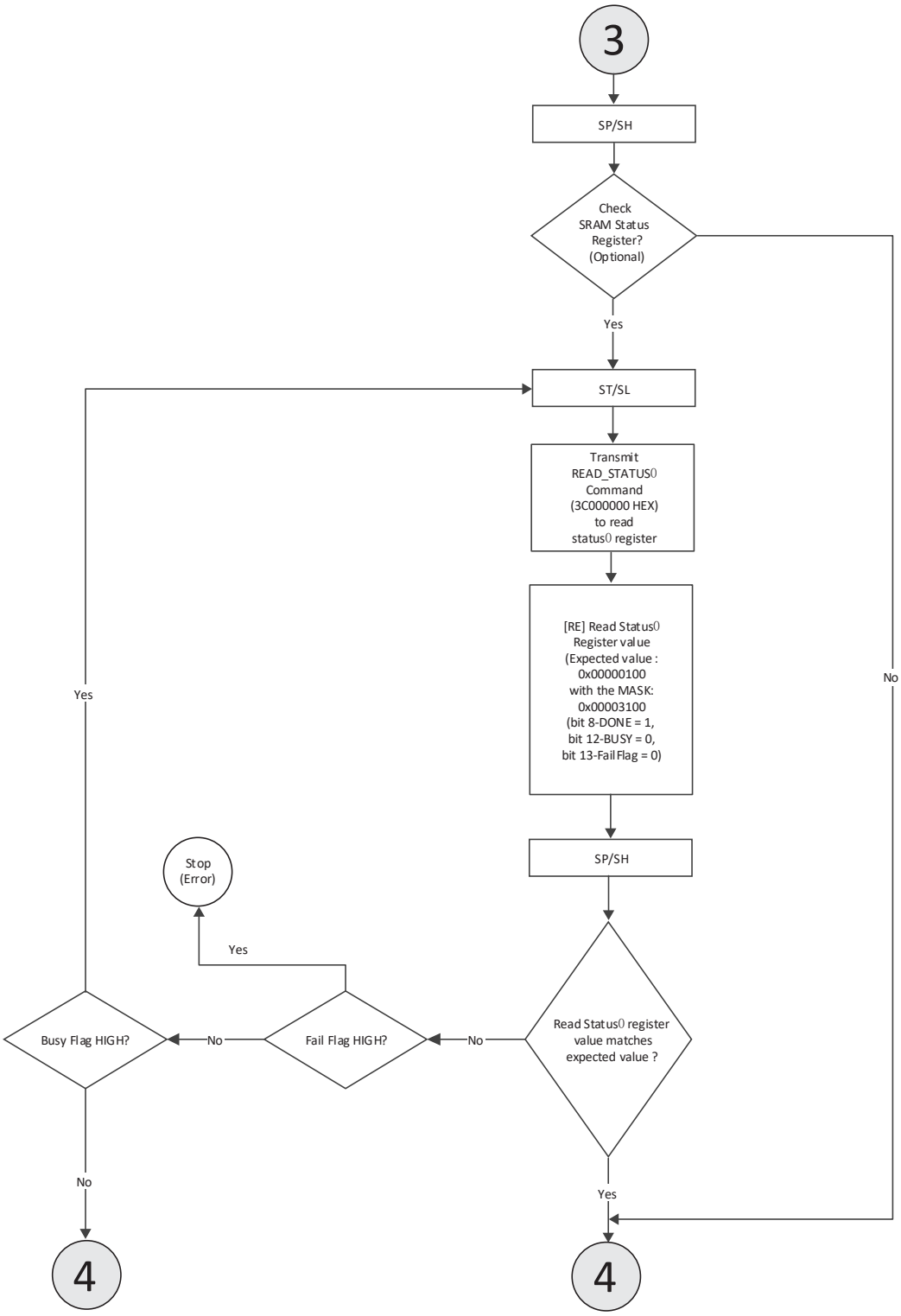
[1] Configuration Interface needs to be re-enabled for reading Status Register or Device ID after Disable command (0x26) is issued for I²C SRAM Configuration.

[2] Enable instruction (offline mode) is of 4 bytes for SPI i.e. C6000000









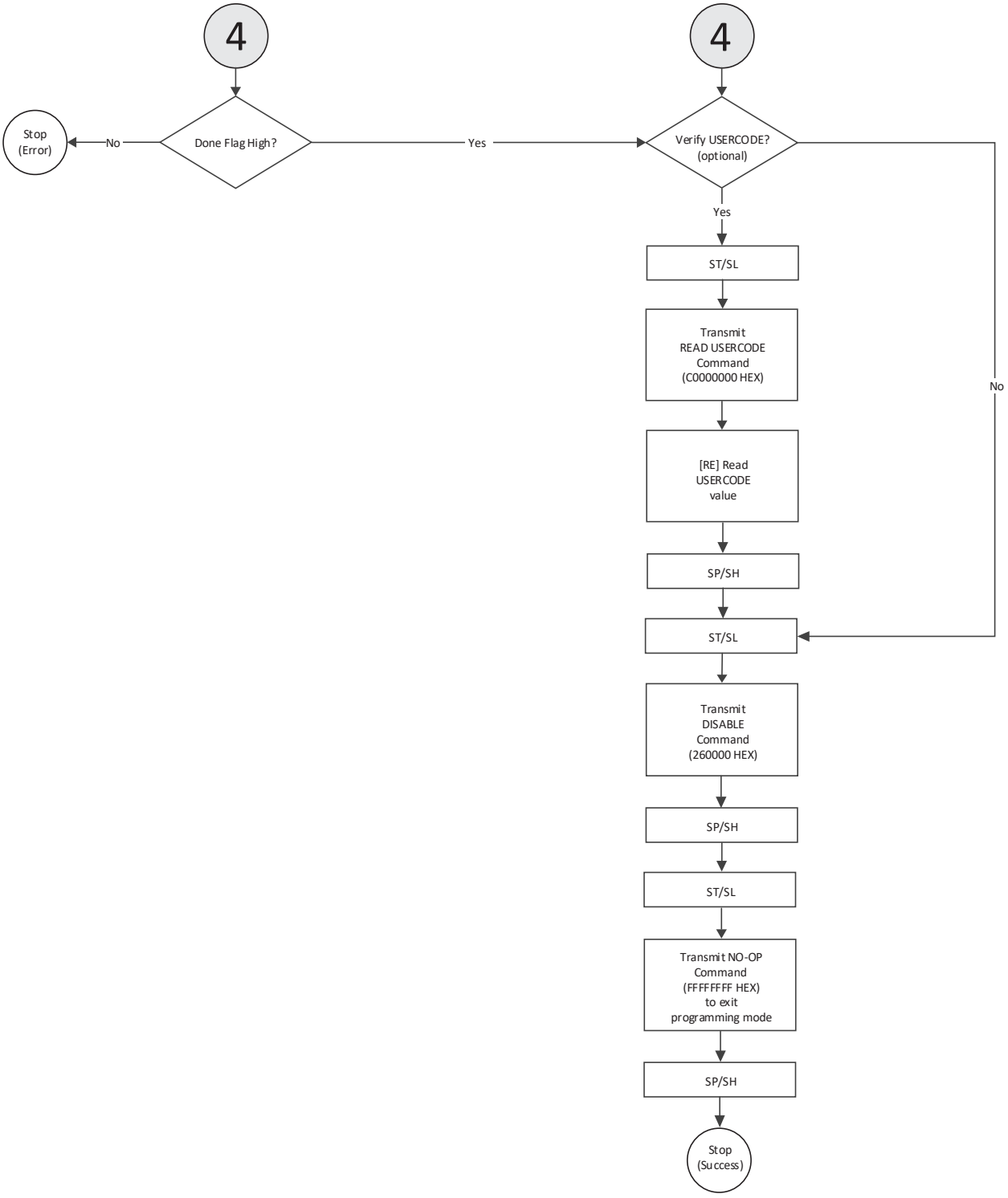
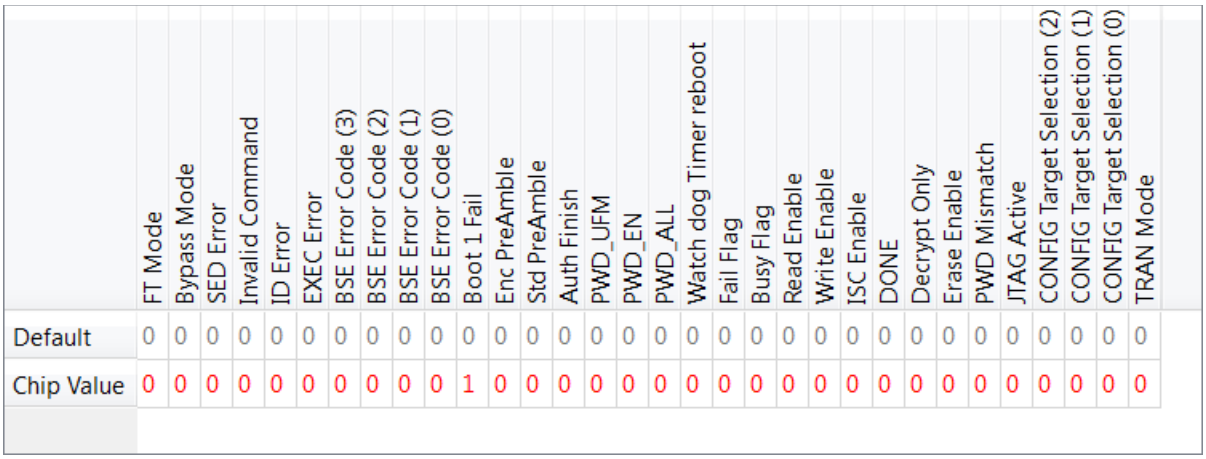


Figure 8.3. Mach-NX Slave SPI/I²C SRAM Configuration Flow



	FT Mode	Bypass Mode	SED Error	Invalid Command	ID Error	EXEC Error	BSE Error Code (3)	BSE Error Code (2)	BSE Error Code (1)	BSE Error Code (0)	Boot 1 Fail	Enc PreAmble	Std PreAmble	Auth Finish	PWD_UFM	PWD_EN	PWD_ALL	Watch dog Timer reboot	Fail Flag	Busy Flag	Read Enable	Write Enable	ISC Enable	DONE	Decrypt Only	Erase Enable	PWD Mismatch	JTAG Active	CONFIG Target Selection (2)	CONFIG Target Selection (1)	CONFIG Target Selection (0)	TRAN Mode
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
Chip Value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

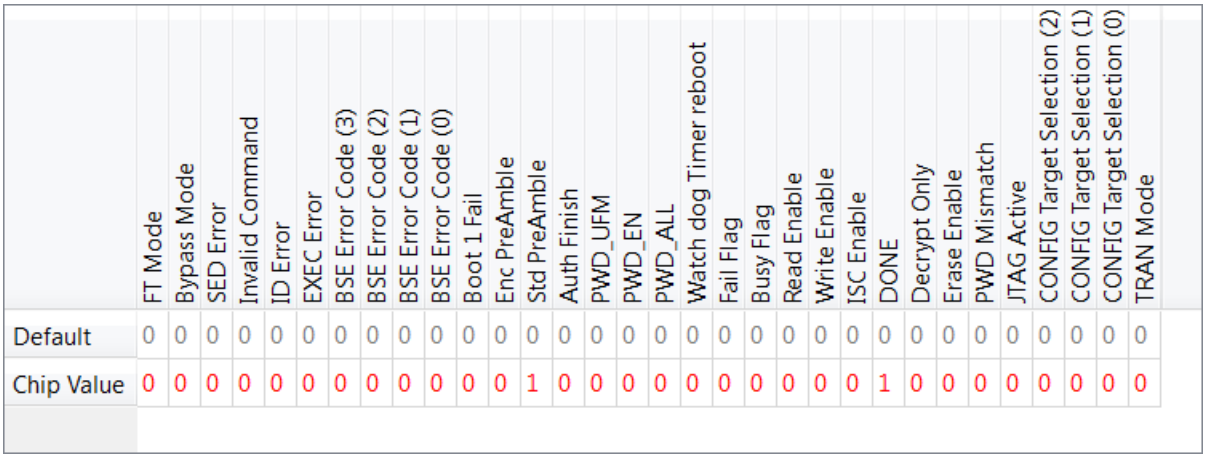
Figure 8.4. Status0 Register Value after Erasing

Expected value from SO: 0x00000000 with the MASK: 0x00003100

(bit 8-DONE = 0, bit 12-BUSY = 0, bit 13-FailFlag = 0)

Mask = 0 means don't care

Mask = 1 means care



	FT Mode	Bypass Mode	SED Error	Invalid Command	ID Error	EXEC Error	BSE Error Code (3)	BSE Error Code (2)	BSE Error Code (1)	BSE Error Code (0)	Boot 1 Fail	Enc PreAmble	Std PreAmble	Auth Finish	PWD_UFM	PWD_EN	PWD_ALL	Watch dog Timer reboot	Fail Flag	Busy Flag	Read Enable	Write Enable	ISC Enable	DONE	Decrypt Only	Erase Enable	PWD Mismatch	JTAG Active	CONFIG Target Selection (2)	CONFIG Target Selection (1)	CONFIG Target Selection (0)	TRAN Mode
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	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

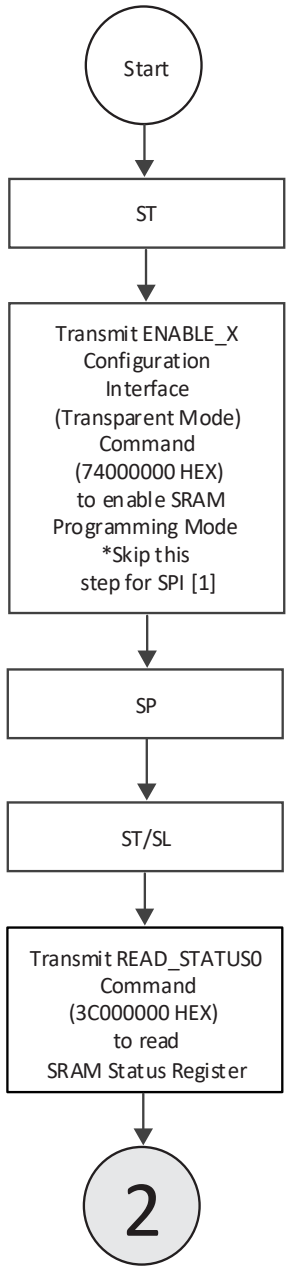
Figure 8.5. Status0 Register Value after Programming

Expected value from SO: 0x00000100 with the MASK: 0x00003100

(bit 8-DONE = 1, bit 12-BUSY = 0, bit 13-FailFlag = 0)

Mask = 0 means don't care

Mask = 1 means care



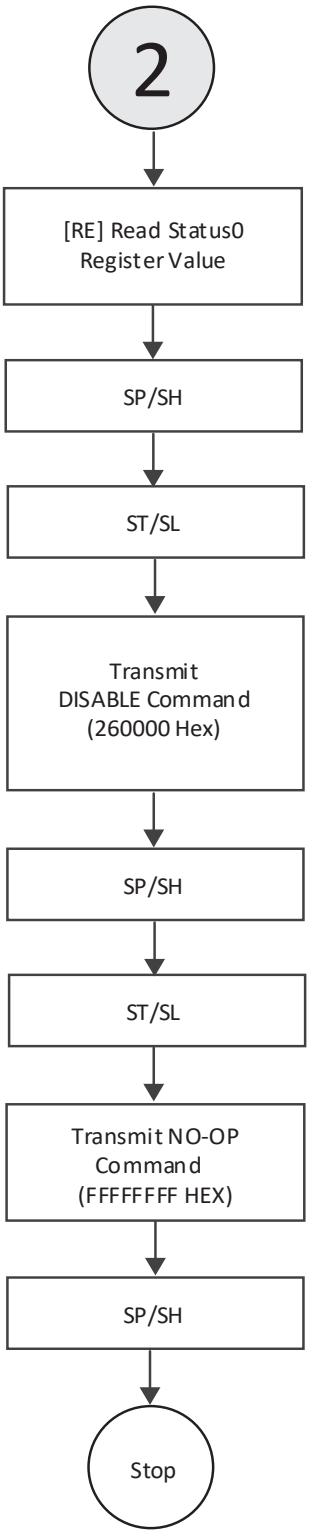


Figure 8.6. Slave SPI/I²C SRAM Read Status0 Register Flow

8.5. Mach-NX Programming Commands

Table 8.2. Mach-NX sysCONFIG Programming Commands

Command Name [SVF Synonym]	Command	Operands	Write Data	Read Data	Notes
Read Device ID [IDCODE_PUB]	0xE0	00 00 00	N/A	YY YY YY YY	YY characters represent the device-specific ID code.
Enable Configuration Interface (Transparent Mode) [ISC_ENABLE_X]	0x74	08 00 001	N/A	N/A	Enable the configuration logic for device programming in Transparent mode.1
Enable Configuration Interface (Offline Mode) [ISC_ENABLE]	0xC6	08 00 001	N/A	N/A	Enable the configuration logic for device programming in Offline mode.1
Read Busy Flag [LSC_CHECK_BUSY]	0xF0	00 00 00	N/A	YY	Bit 1 0 7 Busy Ready
Read Status0 Register [LSC_READ_STATUS0]	0x3C	00 00 00	N/A	YY YY YY YY	Bit 1 0 12 Busy Ready 13 Fail OK
Erase [ISC_ERASE]	0x0E	0Y YY 00	N/A	N/A	Erase the different internal memories. The bit in YYY defines which memory is erased in Flash access mode. Bit 1=Enable 8 Erase CFG0 9 Erase CFG1 10 Erase UFM0 11 Erase UFM1 12 Erase UFM2 13 Reserved 14 Erase CSEC 15 Erase USEC 16 Erase PUBKEY 17 Erase AESKEY 18 Erase FEA 19 Reserved In SRAM access mode this command is used to erase SRAM with all bits of operands reserved.
Erase UFM [LSC_ERASE_TAG]	0xCB	00 YY 00	N/A	N/A	Erase the UFM sectors. The bit in YY defines which sector is erased in Flash access mode. Bit 1=Enable 8 Reserved 9 Reserved 10 Erase UFM0 11 Erase UFM1 12 Erase UFM2 13 Reserved 14 Reserved 15 Erase USEC
Reset Memory Address [LSC_INIT_ADDRESS]	0x46	YY YY 00	N/A	N/A	Set Page Address pointer to the beginning of the different internal Flash sectors. The bit in YYYY defines which sector is selected. Bit Flash sector selected

Command Name [SVF Synonym]	Command	Operands	Write Data	Read Data	Notes
					8 CFG0 9 CFG1 10 FEA 11 PUBKEY 12 AESKEY 13 CSEC 14 UFM0 15 UFM1 16 UFM2 17 Reserved 18 USEC 19 Reserved 20 Reserved 21 Reserved 22 Reserved Only one of the above bits is allowed to be set at a time. Otherwise, no sector is selected. Bit 23 is CRC check bit, 0 for no CRC check and 1 for CRC check.
Set Address [LSC_WRITE_ADDRESS]	0xB4	00 00 00	00 0P PP PP	N/A	Set the Page Address pointer to the Flash page specified by the least significant 18 bits of the P PP PP field. Bit17 ~ Bit14 selects the Flash space to access. Bit[17:14] Flash sector selected 4'h0 CFG0 4'h1 UFM0 4'h2 Reserved 4'h3 FEA 4'h4 CFG1 4'h5 UFM1 4'h6 PUBKEY 4'h7 CSEC 4'h8 UFM2 4'h9 Reserved 4'hA ASEKEY 4'hB USEC Bit13 ~ Bit0 defines the page address.
Program Page [LSC_PROG_INCR_NV]	0x70	00 00 01	YY * 16	N/A	Program one Flash page. Can be used to program the CFG or UFM.
Reset UFM Address [LSC_INIT_ADDR_UFM]	0x47	00 YY 00	N/A	N/A	Set the Page Address Pointer to the beginning of the UFM sectors. The bit in YY defines which sector is selected. Bit UFM sector selected 8 Reserved 9 Reserved 10 UFM0 11 UFM1 12 UFM2 13 Reserved 14 Reserved 15 USEC

Command Name [SVF Synonym]	Command	Operands	Write Data	Read Data	Notes
					Only one of the above bits is allowed to be set at a time. Otherwise, no sector is selected.
Program UFM Page [LSC_PROG_TAG]	0xC9	00 00 01	YY * 16	N/A	Program one UFM page.
Program USERCODE [ISC_PROGRAM_USERCODE]	0xC2	00 00 00	YY * 4	N/A	Program the USERCODE.
Read USERCODE [USERCODE]	0xC0	00 00 00	N/A	YY * 4	Retrieves the 32-bit USERCODE value.
Write Feature Row [LSC_PROG_FEATURE]	0xE4	00 00 00	YY * 8	N/A	Program the Feature Row bits.
Read Feature Row [LSC_READ_FEATURE]	0xE7	00 00 00	N/A	YY * 8	Retrieves the Feature Row bits.
Write FEABITS [LSC_PROG_FEABITS]	0xF8	00 00 00	YY * 2	N/A	Program the FEABITS.
Read FEABITS [LSC_READ_FEABITS]	0xFB	00 00 00	N/A	YY * 2	Retrieves the FEABITS.
Read Flash [LSC_READ_INCR_NV]	0x73	M0 PP PP	N/A	See the Reading Flash Pages section.	Retrieves PPPP count pages. Only the least significant 14 bits of PP PP are used. The *M* field must be set based on the configuration port being used to read the Flash. 0x0 I^2C 0x1 JTAG/SSPI
Read UFM [LSC_READ_UFM]	0xCA	M0 PP PP	N/A	See the Reading Flash Pages section.	Retrieves PPPP count UFM pages. Only the least significant 14 bits of PP PP are used for the page count. The *M* field must be set based on the configuration port being used to read the UFM. 0x0 I^2C 0x1 JTAG/SSPI
Program DONE [ISC_PROGRAM_DONE]	0x5E	00 00 00	N/A	N/A	Program the DONE status bit enabling SDM.
Program AES feature bits [LSC_PROG_AES_FEA]	0xF9	00 00 00	PP	N/A	Program AES feature decryption bits Bit 1=Enable DEC_ONLY RAND_AES RAND_NOISE
Read AES feature bits [LSC_READ_AES_FEA]	0xFA	00 00 00	N/A	PP	Read AES feature decryption bits Bit 1=Enable DEC_ONLY RAND_AES RAND_NOISE
Disable Configuration Interface [ISC_DISABLE]	0x26	0000	N/A	N/A	Exit Offline or Transparent programming mode. ISC_DISABLE causes the Mach-NX device to automatically reconfigure when leaving Offline programming mode. Thus, when leaving Offline programming mode the Configuration

Command Name [SVF Synonym]	Command	Operands	Write Data	Read Data	Notes
					SRAM must be explicitly cleared using ISC_ERASE (0x0E) prior to transmitting ISC_DISABLE. The recommended exit command from Offline programming mode is LSC_REFRESH (0x79), wherein ISC_ERASE and ISC_DISABLE are not necessary. See Figure 8.2.
Bypass [ISC_NOOP]	0xFF	FF FF FF	N/A	N/A	No Operation and Device Wakeup.
Refresh [LSC_REFRESH]	0x79	00 00	N/A	N/A	Force the Mach-NX device to reconfigure. Transmitting a REFRESH command reconfigures the Mach-NX device in the same fashion as asserting PROGRAMN.
Program SECURITY [ISC_PROGRAM_SECURITY]	0xCE	00 00 00	YY	N/A	Programs security or lock bits to current sector of Flash memory. Bit 1=Enable 0 SEC_ERASE 1 SEC_PROG 2 SEC_READ 3 SEC_HLOCK See Section 4.11
Program AUTH_DONE [LSC_PROG_AUTH_DONE]	0xCC	00 00 00	N/A	N/A	Program AUTH_DONE bit in CFG0 or CFG1 if pre-authentication feature is required.
Read TraceID code [UIDCODE_PUB]	0x19	00 00 00	N/A	YY*8	Read 64-bit TraceID.
Configure SRAM [LSC_BITSTREAM_BURST]	0x7A	00 00 00	Compressed bitstream	N/A	Shift in bitstream (.bit) generated by Diamond. Recommend using compressed bit-stream to reduce configuration time. Number of bits varies depending on compression ratio.
Program Flash Protect Key [LSC_PROG_PASSWORD]	0xF1	00 00 00	YY*16	N/A	Program the 128-bit Password into the device.
Read Flash Protect Key [LSC_READ_PASSWORD]	0xF2	00 00 00	N/A	YY*16	Read the 128-bit Password from the device.
Shift Flash Protect Key [LSC_SHIFT_PASSWORD]	0xBC	00 00 00	YY*16	N/A	Present the 128-bit Password. When enabled (PWD_EN = 1), the write data is compared to the Password contained into the Feature Row. If the values match, the device is unlocked for programming and configuration operations. The device remains unlocked until a Disable Configuration command is received, a Refresh command is issued, or a power cycle event occurs.

Note: Transmit the command opcode and first two operand bytes when using the I^2C port. The final operand byte must not be transmitted.

8.6. Reading Flash Pages

Reading the CFG and UFM pages requires a specific procedure. The CFG and UFM pages are accessible from any of the Mach-NX device configuration ports. The JTAG and Slave SPI configuration ports all behave identically when performing read operations. The I²C port requires a modified access protocol. A high-level representation of the data flow, by port, is shown in Figure 8.8.

All ports start the read process in the same way, by sending a Read Flash/Read UFM command. The Mach-NX device begins the read process once the command byte is accepted by the configuration logic. The Page Address Pointer determines the first page returned from the Mach-NX device. For the first returned page to be valid (for example, for single-page read operations), a Retrieval delay of 240 ns must be observed. The Retrieval delay time is from the end of the Command byte transmission to the end of the first Operand byte transmission as shown in Figure 8.7. Note that for slower interface clock rates, 240 ns may be consumed entirely by the normal transmission of the first Operand and no additional delay may be necessary.

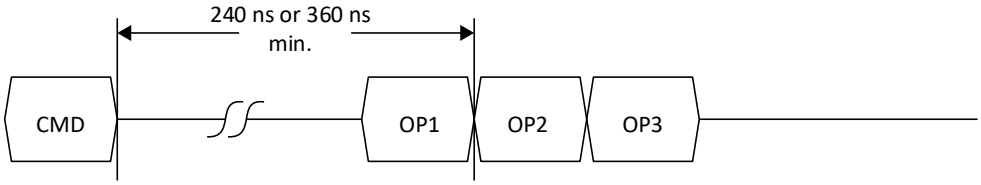
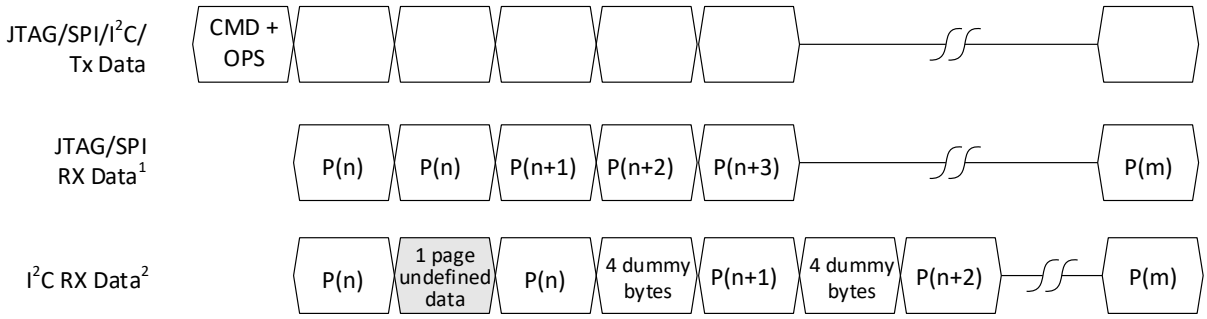


Figure 8.7. Retrieval Delay Timing Requirement for Single-Page Reads



Notes:
1. JTAG/SSPI must transmit data to read data back. The data sent by the JTAG/SSPI master is not specified (that is, don't care).
2. The I²C must use RESTART between sending the CMD and reading the data. (Issuing a STOP terminates a CMD and resets the I²C state machine.)
CMD + OPS = Read CFG or Read UFM command byte + 3 operand bytes.

Figure 8.8. Flash Page Command and Data Sequence

Figure 8.8 shows a multiple page read sequence. The Read CFG/Read UFM command is transmitted to the Mach-NX device. As can be seen in Figure 8.8, all interfaces return the page at the Page Address Pointer immediately. For single-page read operations, all configuration ports are allowed to terminate the read immediately following the transfer of the final byte of the first page. The I²C interface differs only in the Read CFG/Read UFM operand bytes.

Reading more than one page requires special handling. The multiple page read duplicates the page selected by the Page Address Pointer. The result of this behavior is that the page count must be one greater than the desired number of pages. For example, reading two pages requires the page count supplied in the Read CFG/Read UFM command to be assigned a value of 3. If the Page Address Pointer is 0000, the Mach-NX device returns three pages, Page 0, Page 0, and Page 1. A restriction must be observed when using the AHB-Lite interface to read the Flash. When reading 13 or more pages, the page count must be set to the maximum (16383 decimal or 0x3FFF). The user logic is not required to read this number of pages and may safely truncate the read operation after the desired number of pages are read.

The I^2C interface has additional overhead when reading Flash pages. Reviewing Figure 8.8 shows how the data is presented during a multiple page read request. When the page count is three, and the Page Address Pointer is 0000, the I^2C interface returns Page 0, 16 undefined bytes, Page 0, 4 dummy bytes, and Page 1. Reading the final four dummy bytes is optional.

Appendix A. Mach-NX Device Specific

Table A.1. Mach-NX Device ID

Device Name	32-bit Device ID Code
LFMNX-50	412E3043

Appendix B. Mach-NX Internal Flash Programming

To program the Mach-NX internal flash memory using Lattice Diamond® Programmer (Rev3.12 or later) through the HW-USBN-2B cable, launch Lattice Diamond Programmer. Two devices show up in the JTAG scan chain, as show in Figure B.1. The first item is the Mach-NX FPGA unit. The second item is the SoC Function Block unit of the Mach-NX device.

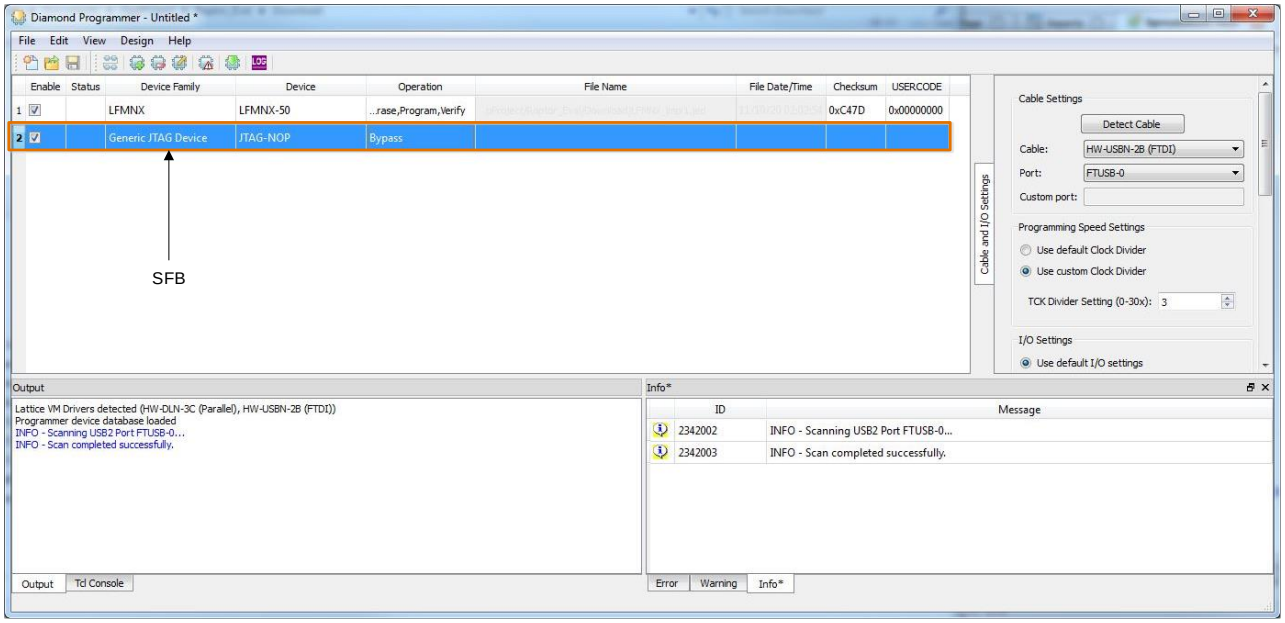


Figure B.1. Lattice Diamond Programmer for Mach-NX Device

To setup the JTAG chain, click the Operation field of the SoC Function Block unit. In the pop-up Device Properties dialog, set the Instruction register length to 8, as show in Figure B.2. Click OK.

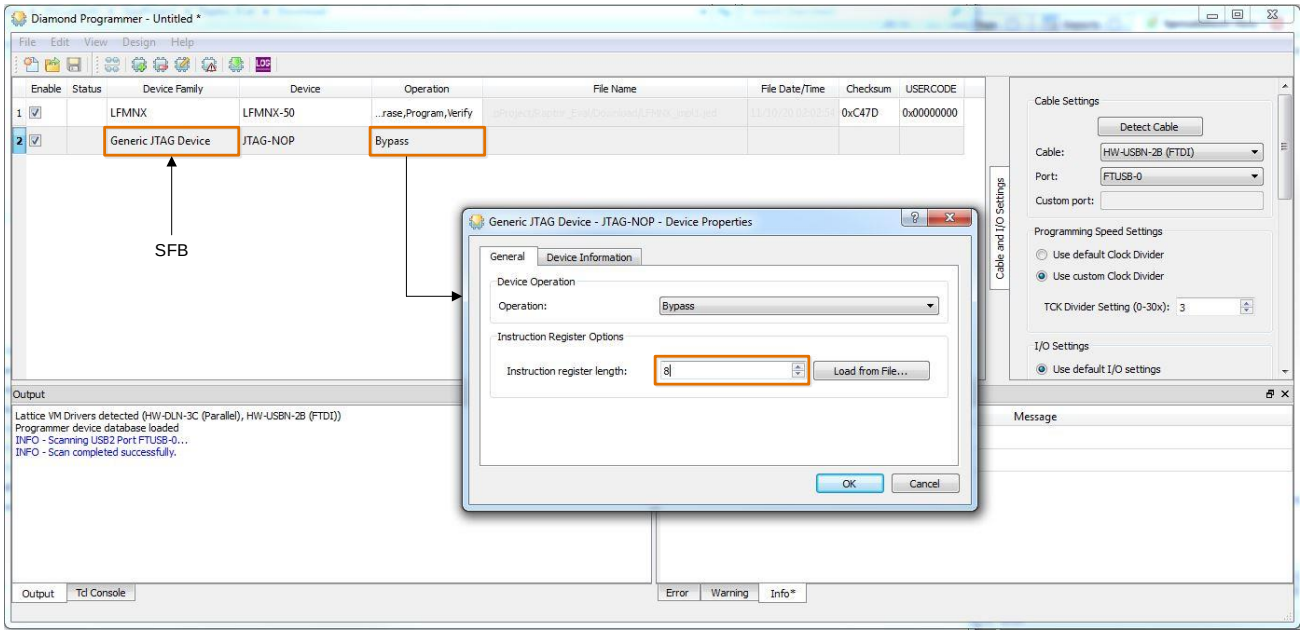


Figure B.2. Lattice Diamond Programmer JTAG Chain Setup

To setup the Mach-NX Flash contents for programming, click on the Operation field of the Mach-NX FPGA unit. In the pop-up Device Properties dialog, select the bitstream file (.jed) for CFG0 and the Flash Address Memory (FAM) Programming file of the SoC Function Block unit for the design, as shown in Figure B.3. Optionally, you can set up the bitstream file for CFG1, and/or contents for UFM0, UFM1, and UFM2. Click OK.

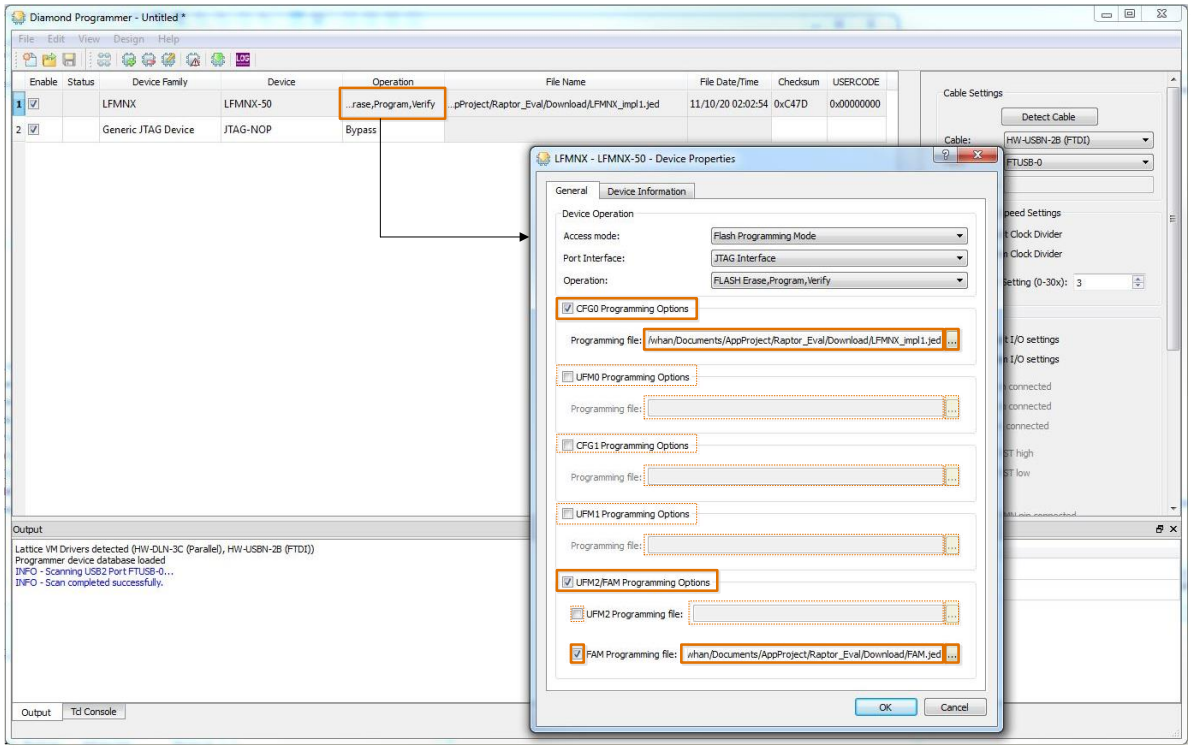


Figure B.3. Programming Files Setup for Diamond Programmer

Once you finish the setup, click the **Program** button from the toolbar to execute the flash programming, as shown in Figure B.4.

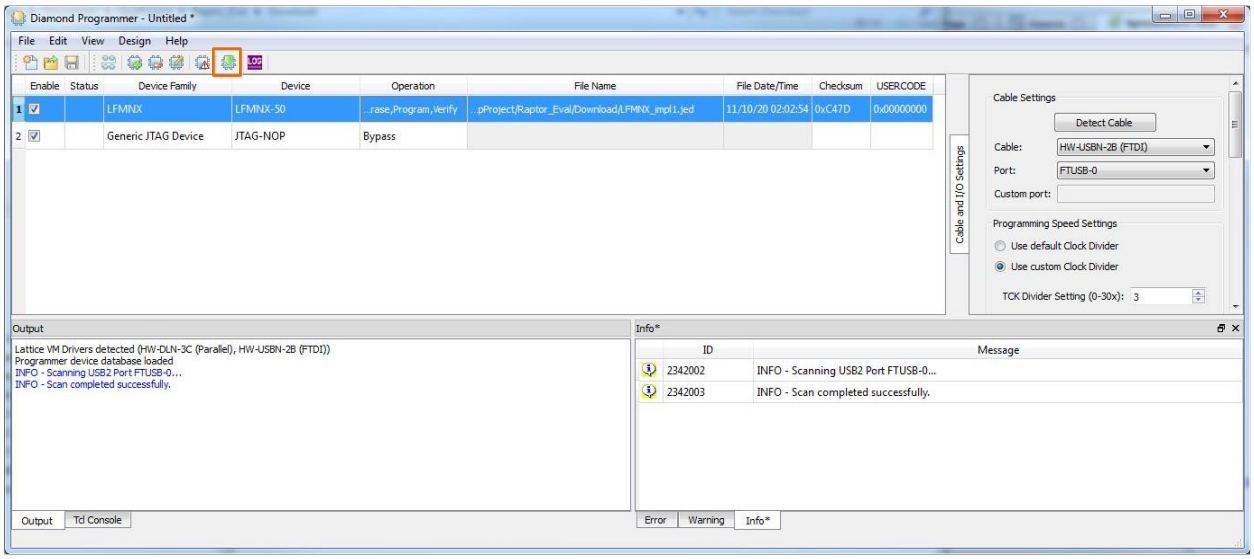


Figure B.4. Flash Programming Execution

Optionally, you can save the Diamond Programmer setup into a .xcf file for later programming application.

References

- Mach-NX Device Family Data Sheet (FPGA-DS-02084)
- Mach-NX PFR and SFB Architecture Usage Guide (FPGA-TN-02230)
- Mach-NX SFB Hardware Usage Guide (FPGA-TN-02222)
- Mach-NX Secure Enclave Usage Guide (FPGA-TN-02252)

Technical Support Assistance

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

Revision History

Revision 1.0, April 2021

Section	Change Summary
All	Production release.

www.latticesemi.com