MachXO2 Programming and Configuration Usage Guide

October 2012 Technical Note TN1204

Introduction

The MachXO2™ is an SRAM-based Programmable Logic Device that includes an internal Flash memory which makes the MachXO2 appear to be a non-volatile device. The MachXO2 provides a rich set of features for programming and configuration of the FPGA. You have many options available to you for building the programming solution that fits your needs. Each of the options available will be described in detail so that you can put together the programming and configuration solution that meets your needs.

MachXO2 Features

Key programming and configuration features of MachXO2 devices are:

- Instant-on configuration from internal Flash PROM – powers up in milliseconds
- Single-chip, secure solution
- Multiple programming and configuration interfaces:
  - 1149.1 JTAG
  - Self download
  - Slave SPI
  - Master SPI
  - Dual Boot
  - I^2C
  - WISHBONE bus
- User Flash Memory (UFM) for non-volatile data storage:
  - Configuration Flash memory overflow
  - EBR Initialization data
  - Application specific data
- Transparent programming of non-volatile memory
- Optional dual boot with external SPI memory
- Optional security bits for design protection

Definition of Terms

This document uses the following terms to describe common functions:

- **Programming**: Programming refers to the process used to alter the contents of the internal or external non-volatile configuration memory.
- **Configuration**: Configuration refers to a change in the state of the MachXO2 SRAM memory cells.
- **Configuration Mode**: The configuration mode defines the method the MachXO2 uses to acquire the configuration data from the non-volatile memory.
- **Configuration Data** – This is the data read from the non-volatile memory and loaded into the FPGA's SRAM configuration memory. This is also referred to as a bitstream, or device bitstream.
- **JEDEC** – The JEDEC file contains the configuration data programmed into the MachXO2 Configuration Flash, User Flash Memory, Feature Row, and Feature Bits. Format information is provided later in this technical note.
- **BIT** – The BIT file is the configuration data for the MachXO2 that is stored in an external SPI Flash. It is a binary file and is programmed unmodified into the SPI Flash.

- **Port** – A port refers to the physical connection used to perform programming and some configuration operations. Ports on the MachXO2 include JTAG, SPI, I^2C, and WISHBONE physical connections.
- **User Mode** – The MachXO2 is in user mode when configuration is complete, and the FPGA is performing the logic functions you have programmed it to perform.
- **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 MachXO2 does not perform your logic operations until offline mode programming/configuration is complete.
- **Transparent Mode** – Transparent mode is used to update the Configuration Flash, and User Flash Memory while leaving the MachXO2 in User Mode.
- **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

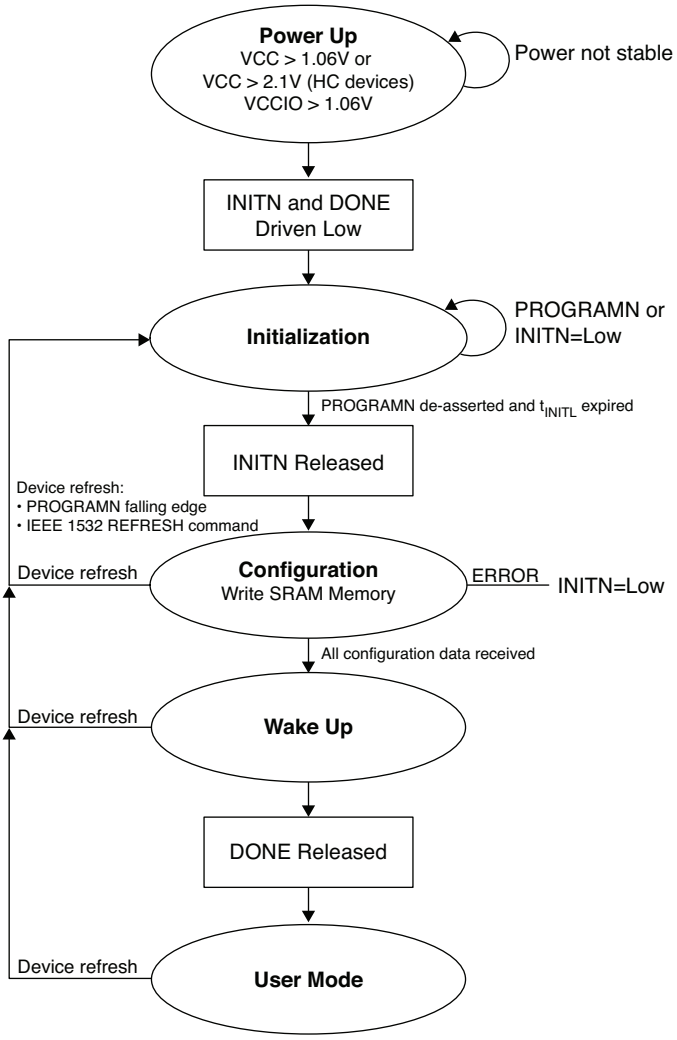
Configuration Details

MachXO2 devices contain 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 FPGAs SRAM. The MachXO2 provides an internal Flash memory that stores the configuration data loaded into the MachXO2 SRAM.

Configuration Process and Flow

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

Figure 14-1. Configuration Flow



The MachXO2 sysCONFIG ports provide industry standard communication protocols for programming and configuring the FPGA. Each of the protocols shown in Table 14-1 provides a way to access the MachXO2 device's internal Flash memory, 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 Flash memory have a priority order. Table 14-1 lists each of the sysCONFIG ports in their priority order. The MSPI configuration port does not have the ability to alter the Flash memory space, and as a result is not a factor in the sysCONFIG port priority scheme. The priority scheme is important to be aware of, as a Configuration Logic operation using a low priority sysCONFIG port can be interrupted by a higher priority sysCONFIG port. 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 Configuraiton Logic using a sysCONFIG port.

Power-up Sequence

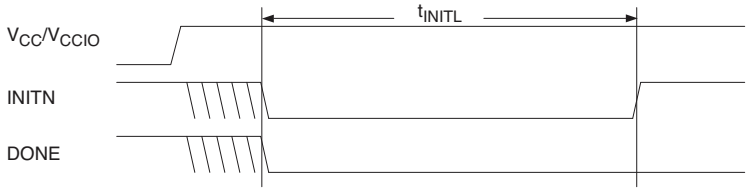
In order for the MachXO2 to operate, power must be applied to the device. During a short period of time, as the voltages applied to the system rise, the FPGA will have an indeterminate state.

As power continues to ramp, a Power On Reset (POR) circuit inside the FPGA becomes active. The POR circuit, once active, makes sure the external I/O pins are in a high-impedance state. It also monitors the V_{CC} and V_{CCIO0} input rails. The POR circuit waits for the following conditions:

- V_{CC} > 1.06V (or 2.1V for HC devices)
- V_{CCIO0} > 1.06V

When these conditions are met the POR circuit releases an internal reset strobe, allowing the device to begin its initialization process. The MachXO2 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 14-1.

Figure 14-2. Configuration from Power-On-Reset Timing



Initialization

The MachXO2 enters the memory initialization phase immediately after the Power On Reset circuit drives the INITN and DONE status pins low. The purpose of the initialization state is to clear all 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 has elapsed
- The PROGRAMN pin is deasserted
- The INITN pin is no longer asserted low by an external master

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

During the t_{INITL} time period the FPGA is clearing the configuration SRAM. When the MachXO2 is part of a chain of devices each device will have different t_{INTIL} initialization times. The FPGA with the slowest t_{INTIL} parameter can prevent other devices in the chain from starting to configure. Premature release of the INITN in a multi-device chain may cause configuration of one or more chained devices to fail to configure intermittently.

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

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 Diamond development tools.

The MachXO2 begins fetching configuration data from non-volatile memory. The memory used to configure the MachXO2 is either the internal Flash, or an external SPI Flash. The MachXO2 does not leave the Configuration

state if there are no memories with valid configuration data. It is necessary to program the non-volatile memory internal or attached to the FPGA, or to program it using the JTAG port.

During the time the FPGA receives its configuration data the INITN control pin takes on its final function. INITN is used to indicate an error exists in the configuration data. When INITN is high, configuration proceeds without issue. If INITN is asserted low, an error has occurred and the FPGA will not operate.

Wake-up

Wake-up is the transition from configuration mode to user mode. The MachXO2's fixed four-phase wake-up sequence starts when the device has correctly received 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 MachXO2 to release the Global Output Enable. When it is asserted, permits the FPGA's I/O to exit a high-impedance state and take on 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 that have 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, as mentioned in the Global Output Enable section. 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 bi-directional, open-drain I/O only when it is enabled. An external agent that holds the external DONE pin low prevents the wake-up process of the MachXO2 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.

User Mode

The MachXO2 enters User Mode immediately following the Wake-Up sequence has completed. User Mode is the point in time when the MachXO2 begins performing the logic operations you designed. The MachXO2 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

Clearing the Configuration Memory and Re-initialization

The current user mode configuration of the MachXO2 remains in operation until it is actively cleared, or power is lost. Several methods are available to clear the internal configuration memory of the MachXO2. The first is to

remove power and reapply power. Another method is to toggle the PROGRAMN pin. Lastly you can reinitialize the memory through a Refresh command. Any active configuration port can be used to send a Refresh command.

- Assertion of the PROGRAMn input
- Cycling power to the MachXO2
- Sending the Refresh command using a configuration port

Invoking one of these methods causes the MachXO2 to drive INITN and DONE low. The MachXO2 enters the initialization state as described earlier.

Memory Space Accessibility

The two internal memories, Flash and SRAM, of the MachXO2 have the ability to be read and written. Each port on the MachXO2 has a different level of access to each memory space. Table 14-2 provides a cross-reference of the MachXO2 ports and the memory space they can access.

As can be seen from Table 14-1, the JTAG port has the ability to read and write both of the internal memory spaces. No other port has ability to read the SRAM configuration memory. The JTAG port has the ability to access the two memory spaces in either Offline or Transparent mode. Every other port has some limitation on the functions that can be performed.

Table 14-1. 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	Refresh[2]
I^2C Port	Yes	Yes	No	Refresh[2]
Internal WISHBONE	Yes[1]	Yes[1]	No	No

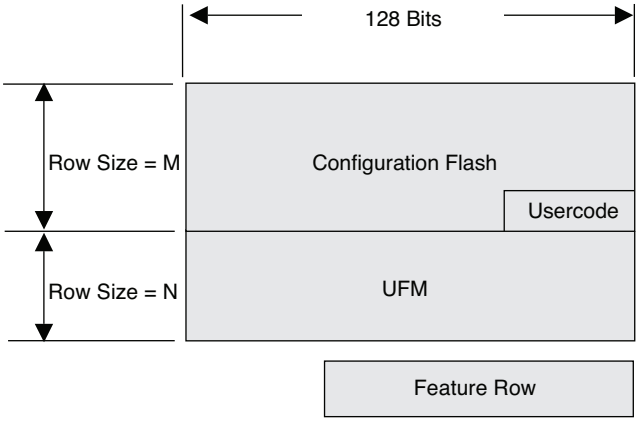
1. In Transparent mode only.
2. See "Clearing the Configuration Memory and Re-initialization" on page 14-5.

Bitstream/PROM Sizes

The MachXO2 is a 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 MachXO2 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.

Storing configuration data in the MachXO2's internal Flash memory has special considerations. The Flash memory in the MachXO2 provides three independent sectors. The first sector is dedicated for use in holding compressed configuration data, and is called Configuration Flash. The second sector, called the User Flash Memory, provides three different functions. It provides additional Configuration Flash storage for large configuration data images, it can store EBR contents, or it is available for use as general purpose Flash memory. The third sector is the Feature Row.

Figure 14-3. Flash Memory Space of a MachXO2 Device



The Configuration Flash is, for most designs, large enough to store the compressed configuration data that is loaded into the SRAM configuration memory. However, as the amount of logic in the design increases, and the amount of pre-initialized EBR increases, the size of the configuration data also increases. The increase in size can cause the configuration data to overflow into the UFM sector. It is also possible, but unlikely, that the configuration data can get too large for the internal Flash memory altogether. In the event configuration data grows too large to fit in the combined Configuration Flash/UFM memory space the design needs to be modified so that it is smaller, or an external configuration memory must be used. You can provide input to the software generating the configuration data to prevent the overflow into the UFM.

In the event the configuration data is too large for the combined Configuration Flash and UFM memory you can store the device bitstream in an external SPI Flash. Table 14-3 shows the maximum uncompressed bitstream sizes allowing you to select a SPI Flash.

Table 14-2. Maximum Configuration Bits

Device	Uncompressed Bitstream Size Without EBR	Uncompressed Bitstream Size With EBR	Maximum Internal Flash	Units
LCMXO2-256	0.09	N/A	0.071	Mb
LCMXO2-640	0.19	0.20	0.17	Mb
LCMXO2-640U			0.33	Mb
LCMXO2-1200	0.35	0.41	0.33	Mb
LCMXO2-1200U			0.47	Mb
LCMXO2-2000	0.51	0.58	0.47	Mb
LCMXO2-2000U			0.80	Mb
LCMXO2-4000	0.93	1.02	0.80	Mb
LCMXO2-7000	1.47	1.70	1.38	Mb

Feature Row

The MachXO2 includes a Feature Row that is used to control FPGA resources. For example, the Feature Row is used to determine how the MachXO2 SRAM configuration memory is loaded. In other FPGAs this operation is controlled using external I/O pins. The Feature Row permits more flexibility in selecting the functions available for configuration, increases the number of available I/O on the device, and eliminates the need to make changes to your hardware.

The resources controlled by the Feature Row become active based upon the configuration state of the MachXO2. There are three states used to determine which Feature Row resources are active:

- User Mode Feature Row
- Flash Mode Feature Row
- Default Mode Feature Row

When the MachXO2 is in User Mode the User Mode Feature Row state is active. The User Mode Feature Row state overrides the settings currently stored in the Feature Row Flash Memory. The MachXO2 User Mode Feature Row state can differ from the Feature Row settings stored in the Flash memory.

- The Feature Row Flash memory can be erased using Transparent Mode
- The configuration SRAM can be configured using the JTAG port with a configuration data image that differs from the one in CF/UFM.

When the MachXO2 configuration SRAM is erased or empty the Flash Mode Feature Row state is active. The values programmed into the Flash Feature Row determine which MachXO2 resources are active. Erasing the SRAM and entering Flash Mode Feature Row state is hazardous It is possible for configuration ports that were active due to being in User Mode Feature Row state to become inaccessible when Flash Mode Feature Row state is entered.

When the MachXO2 configuration SRAM is erased, and the Configuration Flash, UFM, and Feature Row are erased the MachXO2 is in Default Mode Feature Row state. This state differs from Flash Mode Feature Row state in that the Feature Row state is known The Default Mode Feature Row state is described in Table 14-3.

A full list of the functions controlled by the Feature Row and their default values are shown in Table 14-3.

Table 14-3. MachXO2 Feature Row Elements

Feature	Default State (Programmed)
PROGRAMN Persistence	Enabled
INITn Persistence	Disabled
DONE Persistence	Disabled
Custom IDCODE	0x00000000
TraceID™	00
Security[1]	OFF
JTAG Port Persistence	Enabled
SSPI Port Persistence	Disabled[2]
I^2C Port Persistence	Disabled[2]
MSPI Port Persistence	Disabled
I^2C Slave Address[3]	0x00
UFM OTP	OFF
SRAM OTP	OFF
Config Flash OTP	OFF
my_ASSP Enable	0
Boot Select	00b

1. Enabled/disabled using the CONFIG_SECURE preference.
2. Enabled when in Default Mode Feature Row state.
3. Address is assigned in IPexpress™.

The Feature Row is a unique sector in the Flash memory array and can be erased independently from the Configuration Flash and the User Flash Memory. 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 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 MachXO2 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. Entry into Default Mode 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 MachXO2 will never be allowed to complete its configuration process.

Entry into Default Mode Feature Row state occurs in two instances. The first is using Diamond Programmer to explicitly erase the Feature Row. The second, and less obvious time, is when an offline programming operation is performed. One of the first actions taken in an offline Flash erase, program, and verify sequence is the erasure of the Feature Row. Lattice recommends you always use the transparent Flash erase, program, and verify process to avoid entry into Default Mode Feature Row state.

The Feature Row settings are altered indirectly using the Diamond Spreadsheet View. Spreadsheet View allows you to edit the configuration settings for the MachXO2, and then saves your settings in the Lattice Preference File (LPF). These settings are applied to the MachXO2 configuration data during the Map, Place, and Route build phases.

Configuration Modes

The MachXO2 configuration SRAM memory must be loaded with valid configuration data before the FPGA will operate. The MachXO2 provides only four methods of getting the configuration data into the SRAM memory. Each of these methods has its own set of advantages. The four methods available are shown in Table 14-4.

Table 14-4. Configuration Modes

Mode	Number of Pins	Max. Frequency
1149.1 JTAG	4 (5)	25 MHz
Self-Download Mode	0	N/A
External Download	4	50 MHz
Dual Boot Download	0/4	N/A / 50 MHz

The primary configuration mode, for a majority of MachXO2 designs, is Self-Download Mode. It has an advantage in configuration speed because the internal configuration clock runs at frequencies higher than can be applied to an external memory. It does not require an extra PROM, which increases the cost of your product. It does not rely on an external programmer to load the SRAM using the JTAG port.

The External Download mode's advantage is that it makes all of the User Flash Memory available for your use. You do not have to be concerned about the Configuration Flash image overflowing into the UFM, or overflowing the available internal Flash memory.

The Dual Boot mode's advantage is the MachXO2 configures more reliably. The MachXO2 loads the internal Flash memory image first, and should that fail, a fail-safe configuration data image can be downloaded into the MachXO2's SRAM, allowing the FPGA to continue to operate. A failed reprogramming of the internal memory, usually as a result of a loss of power, is the primary reason MachXO2 would fail to configure.

The JTAG port's advantage is that it provides the widest set of functions and features for programming, configuring, and testing the MachXO2 system.

sysCONFIG™ Ports

Table 14-5. MachXO2 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)
	MSPI	Master Serial Peripheral Interface (SPI)
	I^2C	Inter-integrated Circuit (I^2C) Interface
Internal	WISHBONE	Internal WISHBONE bus interface

sysCONFIG Pins

The MachXO2 provides a set of sysCONFIG I/O pins that you use to program and configure the FPGA. The sysCONFIG pins are grouped together to create ports (i.e. JTAG, SSPI, I^2C, MSPI) that are used to interact with the FPGA for programming, configuration, and access of resources inside the FPGA. The sysCONFIG pins in a configuration port group may be active, and used for programming the FPGA, or they can be reconfigured to act as general purpose I/O.

Recovering the configuration port pins for use as general purpose I/O requires 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 MachXO2 is in Default Mode Feature Row state. One example is driving PROGRAMN with an active low signal after the MachXO2 is in Default Mode Feature Row 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 14-6 lists the default state of the shared sysCONFIG pins. As you can see, a Default Mode Feature Row device has the JTAG, SPI Slave and I^2C ports enabled. Upon entry to User Mode the MachXO2, 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 MachXO2 using SSPI, or I^2C when using the default sysCONFIG port settings. To retain the SSPI, or 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.

Table 14-6. Default State of the sysCONFIG Pins

Pin Name	Associated sysCONFIG Port	Default Pin Function in Default Mode Feature Row (Configuration Mode)	Pin Direction (Configuration Mode)	Default Function in User Mode Feature Row
PROGRAMN[1]	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
MCLK/CCLK	SSPI/MSPI	SSPI	Input with weak pull up	User-defined I/O
SN	SSPI/MSPI	SSPI	Input with weak pull up	User-defined I/O
SI/SISPI	SSPI/MSPI	SSPI	Input	User-defined I/O
SO/SPISO	SSPI/MSPI	SSPI	Output	User-defined I/O
CSSPIN	MSPI	I/O	I/O with weak pull up	User-defined I/O
SCL	I^2C	I^2C	Bi-Directional	User-defined I/O
SDA	I^2C	I^2C	Bi-Directional	User-defined I/O

1. The default for PROGRAMN in user mode was PROGRAMN in ispLEVER® 8.1 SP1 and Lattice Diamond® 1.1.

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 MachXO2 from leaving the Initialization phase. The PROGRAMN has a minimum pulse width assertion period in order for it to be recognized by the FPGA. You can find this minimum time in the MachXO2 Family Data Sheet in the AC timing section.

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

- If the device is currently being programmed via JTAG then PROGRAMN will be ignored until the JTAG mode programming sequence is complete.
- Toggling the PROGRAMN pin during device configuration will interrupt the process and restart the configuration cycle.
- Asserting PROGRAMN on a device in Default Mode Feature Row 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 PROGRAMN has been 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 MachXO2 from configuring, causing the FPGA to remain in a continuous RESET condition. See Figure 14-4.

Figure 14-4. Period PROGRAMN is Always Observed

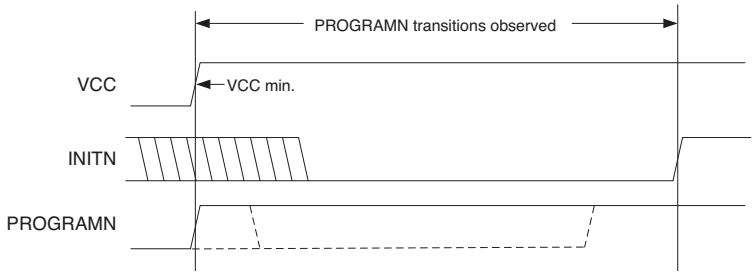
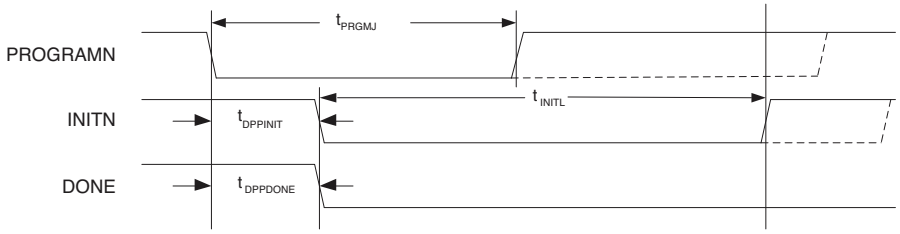


Figure 14-5. Configuration from PROGRAMN Timing



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

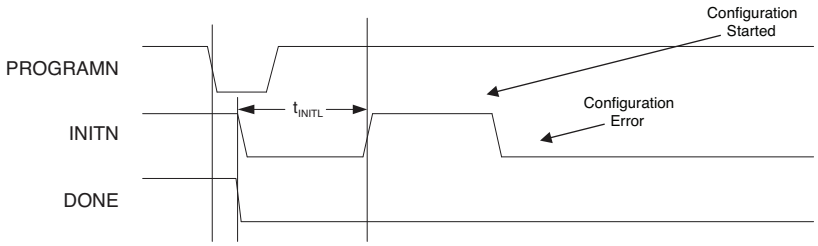
- After power is applied, after a PROGRAMN assertion, or a REFRESH command it goes low to indicate the SRAM configuration memory is being erased. The low time assertion is specified with the t_{INTIL} parameter.
- After the t_{INTIL} time period has elapsed the INITn pin is deasserted (i.e. is active high) to indicate the MachXO2 is ready for its configuration bits. The MachXO2 begins loading configuration data from either the internal Flash memory or an external SPI Flash.
- INITn can be asserted low by an external agent before the t_{INTIL} time period has elapsed in order to prevent the FPGA from reading configuration bits. This is useful when there are multiple programmable devices chained together. The programmable device with the longest t_{INTIL} 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_{INTIL} has elapsed and the INITn pin has gone high, any subsequent INITn assertion signals the MachXO2 has detected an error during configuration.

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

- Power has just been applied
- PROGRAMN falling edge occurred
- The IEEE 1532 REFRESH command has been sent using a slave configuration port (JTAG, SSPI, I^2C or WISHBONE).

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 14-6. Configuration Error Notification



The INITN pin of a MachXO2 device is not visible external to the device when in the Default Mode Feature Row state. The INITN pin, when in this mode, is pulled high by default. The INITN behavior described in Figure 14-6 is only visible outside the MachXO2 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 will go low, the internal DONE bit will not be set, the DONE pin will stay low, and the device will not wake up. The device will fail 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 bi-directional open drain with a weak pull-up that signals the FPGA is in User mode. DONE is first able to indicate entry into User mode only after an internal DONE bit is asserted. The internal DONE bit defines the beginning of the FPGA Wake-Up state.

The 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 MachXO2 is in the Default Mode Feature Row state. The default mode causes the MachXO2 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. 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 has finished configuration. 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, DONE 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 Default Mode Feature Row state. This behavior can make a part appear to be successfully configured to other logic monitoring the DONE pin.

Master and Slave SPI Configuration Port Pins

Table 14-7. Master SPI Configuration Port Pins

Pin Name	Function	Direction	Description
MCLK/CCLK	MCLK	Output with weak pullup	Master clock used to time data transmission/reception from the MachXO2 Configuration Logic to a slave SPI PROM. A 1K pull-up resistor is recommended on MCLK for External and Dual Boot configuration modes.
CSSPIN	CSSPIN	Output	Chip select used to enable an external SPI PROM containing configuration data
SI/SISPI	SISPI	Output	SISPI carries output data from the MachXO2 Configuration Logic to the slave SPI PROM
SO/SPISO	SPISO	Input	SPISO carries output data from the slave SPI PROM to the MachXO2 Configuration Logic
SN	SN/IO	Input	MachXO2 Configuration Logic slave SPI chip select input. Pull high externally whenever the MSPI port is active.

Table 14-8. Slave SPI Configuration Port Pins

Pin name	Function	Direction	Description
MCLK/CCLK	CCLK	Input with weak pullup	Clock used to time data transmission/reception from an external SPI master device to the MachXO2 Configuration Logic.
SI/SISPI	SI	Input	SI carries output data from the external SPI master to the MachXO2 Configuration Logic
SO/SPISO	SO	Input	SO carries output data from the MachXO2 Configuration Logic to the external SPI master
SN	SN	Input with weak pullup	MachXO2 Configuration Logic slave SPI chip select input. SN is an active low input.

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

The MCLK/CCLK pin's default state for a MachXO2 in the Default Mode Feature Row state is to act as the configuration clock (i.e., CCLK). This allows an external Slave SPI master controller to program the MachXO2. The maximum CCLK frequency and the data setup/hold parameters can be found in the AC timing section of the MachXO2 Family Data Sheet. The Feature Row must be configured to ENABLE the Slave SPI Port if you want to use the port to reprogram the MachXO2 after it enters user mode.

The MCLK/CCLK pin functions as a Master Clock (MCLK) when the MachXO2 is configured in Dual Boot or External Boot modes. A 1K pull-up resistor is recommended when using these modes. The MCLK becomes an output and provides a reference clock for a SPI Flash attached to the MachXO2's Master SPI Configuration port. MCLK actively drives until all of the configuration data has been received. When the MachXO2 enters user mode the MCLK output tri-states. This allows the MCLK to become a general purpose I/O. The MCLK is reserved for use, in most post-configuration applications, as the reference clock for performing memory transactions with the external SPI PROM.

The MachXO2 generates MCLK from an internal oscillator. The initial frequency of the MCLK is nominally 2.08 MHz. The MCLK frequency can be altered using the MCCLK_FREQ parameter. You can select the MCCLK_FREQ using the Diamond Spreadsheet View. For a complete list of the supported MCLK frequencies, see Table 14-9.

Table 14-9. MachXO2 MCLK Valid Frequencies (MHz)

2.08	9.17	33.25
2.46	10.23	38.00
3.17	13.30	44.33
4.29	14.78	53.20
5.54	20.46	66.50
7.00	26.60	88.67
8.31	29.56	133.00

During the initial stages of device configuration the frequency value specified using MCCLK_FREQ is loaded into the FPGA. Once the MachXO2 accepts the new MCLK_FREQ value the MCLK output begins driving the selected frequency. Make certain when selecting the MCLK_FREQ that you do not exceed the frequency specification of your configuration memory, or of your PCB. Review the MachXO2 AC specifications in the MachXO2 Family Data Sheet when making MCLK_FREQ decisions.

SN: The SN pin is the Slave SPI ports chip select. An external SPI bus master asserts the SN pin active low in order to perform actions using the MachXO2's programming and configuration logic. The SN pin is available when the MachXO2 is in the Default Mode Feature Row 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 MachXO2 depends upon maintaining the SN pin in the correct state:

- SN must be deasserted (i.e. held high) when configuring using Master SPI mode
- SN must be deasserted when the MachXO2 is in user mode, and SPI memory transactions are initiated using the internal WISHBONE bus
- SN must be deasserted when accessing the Configuration Logic in the MachXO2 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 MachXO2 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.

CSSPIN: The CSSPIN pin is an active low chip select used by the Master SPI configuration mode to enable an external SPI Flash. When the MachXO2 is programmed to configure in either External or Dual Boot mode the CSSPIN pin is asserted to the attached SPI Flash. The MachXO2 asserts CSSPIN until all configuration data bytes have been loaded, at which time the CSSPIN enters a high impedance state.

When the MachXO2 is in the Default Mode Feature Row state the CSSPIN is a general purpose I/O with a weak pulldown. It must have an external pullup resistor when the External and Dual Boot configuration modes are used. CSSPIN must ramp in tandem with the SPI PROM VCC input. It remains a general purpose I/O when the FPGA enters user mode. You must ENABLE the Master SPI port to reserve CSSPIN for use by the internal SPI Master logic.

Some SPI PROM manufacturers require the chip select input of the PROM ramp in unison to the PROMs VCC rail. The CSSPIN pin, by default, has a weak pull-down resistor internally. Adding a 4.7K to 10K ohm pull-up resistor to the CSSPIN pin on the MachXO2 is recommended.

SI/SISPI: The SI/SISPI is a dual function bi-directional pin. The direction depends upon whether a Master or Slave mode is active. The SI/SISPI is an input data pin when using the Slave SPI mode and is an output data pin when using the Master SPI mode. In Master SPI mode, the MachXO2 drives SI/SISPI until all configuration data bytes have been loaded, at which time the SI/SISPI enters a high impedance state.

At least one of the sysCONFIG preferences, SLAVE_SPI_PORT or MASTER_SPI_PORT, must be set to ENABLE in order to preserve this pin as SI/SISPI and allow access to the SPI interface.

SO/SPISO: The SO/SPISO pin is a dual function bi-directional pin. The direction depends upon whether a Master or Slave mode is active. The SO/SPISO is an input data pin when using the Master SPI mode and is an output data pin when using the Slave SPI mode.

At least one of the sysCONFIG preferences, SLAVE_SPI_PORT or MASTER_SPI_PORT, must be set to ENABLE in order to preserve this pin as SO/SPISO and allow access to the SPI interface.

I^2C Configuration Port Pins

SCL: The MachXO2 provides an I^2C configuration port. The SCL is the I^2C Serial Clock pin, and is used to initiate and time transactions on the I^2C bus. It is a bi-directional, open-drain signal that is an output when the MachXO2 I^2C controller is mastering transactions on the bus, and is an input when an external I^2C master is accessing resources inside the MachXO2. SCL requires an external pull-up resistor in order to operate.

The SCL pin is available when the MachXO2 is in the Default Mode Feature Row state. You must ENABLE the I2C_PORT for the I^2C port to continue to be available in user mode. The SCL pin becomes a general purpose I/O if you do not ENABLE the I2C_PORT.

SDA: The SDA pin is the I^2C serial data input/output pin. It is bi-directional, open-drain, and requires an external pull-up resistor in order to operate. The pin changes direction dynamically during data transactions on the I^2C 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 MachXO2 is in the Default Mode Feature Row state. You must ENABLE the I2C_PORT if you want the I^2C port to continue to be available in user mode. The SDA pin becomes a general purpose I/O if you do not ENABLE the I2C_PORT.

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 MachXO2 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 MachXO2 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 MachXO2 is in the Default Mode Feature Row 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 (i.e. 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 in the 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 (i.e. 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 will drive the pin, per the IEEE JTAG standard, rather than normal internal logic.

Table 14-10. 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 will be 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.

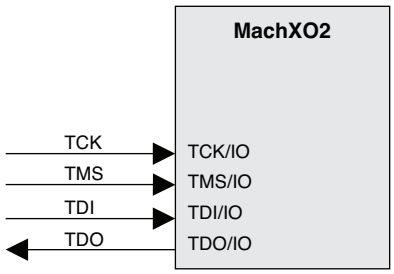
TMS: The Test Mode Select (TMS) pin is an input pin that controls the progression through the 1149.1 compliant state machine states. The TMS pin is sampled on the rising edge of TCK. The JTAG state machine remains in or transitions to a new TAP state depending on the current state of the TAP, and the 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 the MachXO2 Family Data Sheet. The TCK pin does not have a pull-up. An external pull-down resistor of 4.7 K ohms is recommended to avoid inadvertently clocking the TAP controller as power is applied to the MachXO2.

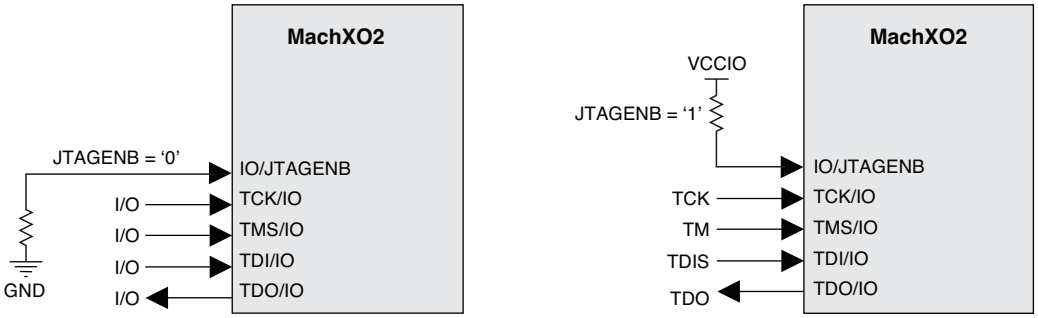
JTAGENB: The JTAG ENABLE pin, also known as the IEEE 1149.1 conformance pin, is an input pin that can be used to multiplex the JTAG port. The JTAGENB pin is only active in user mode. The JTAGENB pin is a user I/O while the JTAG port is in the ENABLE state. Figure 14-7 shows the default behavior of the JTAG port of a MachXO2 device.

Figure 14-7. 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/Os. Driving the JTAGENB input high enables the JTAG port. Figure shows JTAG port behavior under the control of the JTAGENB.

Figure 14-8. JTAG Port Behavior with JTAG_PORT = DISABLE



It is critical when using the JTAGENB feature that logic attached to the JTAG I/O pins not contend with a JTAG programming system. 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.

Configuration Modes

The MachXO2 provides multiple options for loading the configuration SRAM from a non-volatile memory. The previous section described the physical interface necessary to interact with the MachXO2 configuration logic. This sec-

tion 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.

SDM Mode

Self Download Mode is the primary configuration method for the MachXO2. The advantages of Self Download Configuration Mode include:

- **Speed**: The MachXO2 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 MachXO2.
- **Reduced board space**: Elimination of an external PROM allows your board to be smaller.
- **Improved reliability**: The MachXO2 can boot from an external PROM if the internal Flash memory gets corrupted during a system update.

The MachXO2 retrieves the configuration data from the internal Flash memory when it is using Self Download Mode. SDM is triggered when power is applied, a REFRESH command is received, or by asserting the PROGRAMN pin. As shown in Figure 14-5, the internal Flash memory has three sectors. The first sector, in most cases, is large enough to store the MachXO2 device's configuration data. The size of the configuration data changes based on how well it can be compressed and how many pre-initialized EBR components are in the design. As the size of the configuration data increases, the Configuration Flash sector can overflow. The overflow can be handled by allowing the configuration data to overflow into the User Flash Memory sector. It is, in rare cases, possible for the configuration data to overflow the Configuration Flash (CFM), and the User Flash Memory (UFM). Self Download Mode cannot be used when the Configuration Flash and User Flash Memory overflow occurs. Master SPI Configuration Mode must be used in the event of the CF/UFM overflow.

The normal situation for the configuration data is to fit completely within the Configuration Flash sector. Designs that do not use very much pre-initialized EBR will almost always meet this condition. The UFM is available for use as an internal Flash memory array. It is recommended that the CONFIGURATION option be set to CFG. This setting prevents the configuration data from overflowing into the UFM, assuring that data provisioned for the UFM is not overwritten during a device update.

The User Flash Memory, which is the second Flash sector, provides three different use models:

- Configuration data overflow from Configuration Memory
- Initialization of EBR and user-defined storage
- User-defined storage

Diamond, by default, builds the pre-initialized EBR data into the configuration data image. This may cause the configuration data to overflow into the UFM sector. In order to change the default state you need to use the Diamond Spreadsheet tool to modify the sysCONFIG's CONFIGURATION entry. The default state for the CONFIGURATION entry is to be set to CFG.

The configuration data can be logically split to place the pre-initialization data for the EBR into the UFM. Setting the CONFIGURATION option to CFG_EBRUFM causes the Diamond software to place the configuration data into the Configuration Flash, and the EBR initialization data into the UFM. This locates the EBR initialization data into the first pages of the UFM. The current Diamond development tools do not provide a way to map the EBR initialization data stored in the UFM to the associated EBR in the FPGA fabric.

In addition to the automatic assignment of the initialized EBR data, you have the ability to add a data block for your own purposes. Using IPexpress, you can associate a memory initialization file to the UFM. This data is stored in the last memory locations of the UFM in order to prevent collisions with the EBR initialization data.

The user-defined storage mode of operation permits the sector to behave like a general purpose Flash memory. You can choose to use IPexpress to pre-initialize data, or you can use it as if it were a discrete Flash memory device with a single erasable sector.

In all three cases, the UFM can only be erased by erasing the whole sector. It is your responsibility to restore configuration data, EBR initialization data, and your implementation specific data. In other words, you need to read all data in the UFM, merge your changes, erase the UFM, and write the new data back into the UFM.

Master SPI Configuration Mode (MSPI)

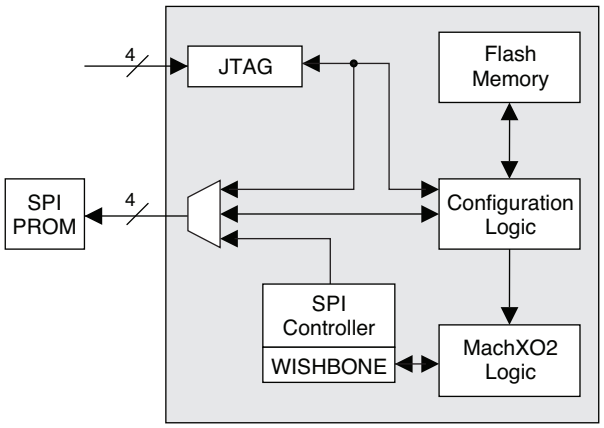
Master SPI Configuration Mode is the only other self-controlled configuration mode available to the MachXO2. When the MachXO2 has the Master SPI Configuration mode (MSPI) enabled it is able to automatically retrieve the configuration data from an externally attached SPI Flash. The MSPI configuration port is not available when the MachXO2 is in the Default Mode Feature Row state. When configuring using the MSPI mode be sure to enable the MSPI port in the Feature Row. Lattice recommends having a secondary configuration port available, one that is active when the MachXO2 is in Default Mode Feature Row state, that allows you to recover the MachXO2 in the event of a programming error.

Table 14-11. Master SPI Port Pins

Pin Name	I^2C Function
MCLK	Clock output from the MachXO2 Configuration Logic and Master SPI controller. Connect MCLK to the SCLK input of the Slave SPI device.
SISPI	Serial Data output from the MachXO2 to the slave SPI SI input.
SPISO	Serial Data input to the MachXO2 configuration logic from the slave SPI SO output.
CSSPIN	Chip select output from the MachXO2 configuration logic to the slave SPI Flash holding configuration data for the MachXO2.

Table 14-2 provides information about the amount of memory needed for MachXO2 configuration data by device density. Select a SPI Flash that accepts 03 hex Read Opcodes. The MachXO2 is only able to use the 03 hex Read Opcode.

Figure 14-9. Master SPI Configuration Mode



The MachXO2 begins retrieving configuration data from the SPI Flash when power is applied, a REFRESH command is received, or the PROGRAMN pin is asserted and released. The MCLK/CCLK I/O takes on the Master Clock (MCLK) function, and begins driving a nominal 2.08 MHz clock to the SPI Flash's SCLK input. CSSPIN is asserted low, commands are transmitted to the PROM over the SI/SISPI output, and data is read from the PROM on the SO/SPISO input pin. When all of the configuration data is retrieved from the PROM the CSSPIN pin is deasserted, and the MSPI output pins are tri-stated.

The MCLK frequency always starts downloading the configuration data at the nominal 2.08 MHz frequency. The MCCLK_FREQ parameter, accessed using Spreadsheet View, can be used to increase the configuration frequency. The configuration data in the PROM has some padding bits, and then the data altering the MCLK base frequency is read. The MachXO2 reads the remaining configuration data bytes using the new MCLK frequency.

After the MachXO2 enters user mode the Master SPI configuration port pins tri-state. This allows data transfers across the SPI. There are two primary methods available for transferring data across the SPI bus. The first method available to you is to enable the Embedded Function Block (EFB) in the MachXO2. Using IPexpress™ you instantiate the EFB, and you choose the features you want active. One of the features available in the EFB is a SPI Master Controller. The SPI Master Controller in the EFB attaches directly Master SPI configuration port pins. The controller provides a set of status, control, and data registers for initiating SPI bus transactions. The registers are accessed using the internal WISHBONE data bus. Logic residing in the programmable section of the the MachXO2 can be created to perform transactions across the WISHBONE bridge to the EFB, which in turn generate SPI bus transactions.

The second way to perform Master SPI configuration port transactions is to master them from the JTAG port. The MachXO2 includes a JTAG to MSPI passthru circuit that allows the slave SPI Flash to be erased, programmed, and read. The primary method for programming the attached SPI Flash is to use Diamond Programmer to transfer a configuration data file from your personal computer. This is useful during board development and debug. *Note: To support JTAG to MSPI passthru programming mode a 1Kohm pull-up resister is required on MCLK*.

Another way to program a SPI Flash using the JTAG port is to use the Lattice ispVME solution. ispVME is C code written for an embedded microprocessor. The microprocessor reads a data file crafted by the Diamond Deployment Tool, and runs the ispVME code. The firmware uses port I/O to drive the JTAG port of the MachXO2, which in turn passes the data to the Master SPI port. Refer to the ispVME tool suite for information about updating an attached SPI Flash using a microprocessor.

The advantage of using the JTAG port for programming the SPI Flash attached to the MachXO2 is that the MachXO2 is permitted to be in the Default Mode Feature Row state. JTAG is able to program a device in Flash Mode Feature Row or User Mode Feature Row state. In order to do so, the Master SPI port pins must be enabled. The passthru is an integral part of the JTAG TAP system. Obviously, the JTAG port must be available in order for this method to succeed.

To set the MachXO2 for operation using the MSPI configuration mode you must:

- Store the entire configuration data in an external SPI Flash
- The data must start at offset 0x000000 within the PROM
- Set the preferences as shown in Table 14-12
- Enable JEDEC File creation in the Diamond Process Pane
- Run the Export Files process to build your design

Table 14-12. Master SPI Configuration Software Settings

Preference	Setting
MASTER_SPI_PORT	ENABLE
CONFIGURATION	EXTERNAL
GENERATE_BITSTREAM	ENABLE

The Export Files process generates both a JEDEC file and BIT file. It is important that both files be used. The JEDEC file must be programmed into the MachXO2 in order to write the Feature Row. The JEDEC file enables the MSPI configuration port.

The BIT file must be programmed into the external SPI Flash. There are several ways to get the data into the SPI Flash:

- Diamond Programmer can transmit the SPI Flash data using a JTAG download cable
- A microprocessor running ispVME
- Automatic Test Equipment can program the SPI Flash using JTAG
- Pre-programmed SPI Flash memories can be pre-assembled onto your printed-circuit board

Once the MachXO2 Feature Row is programmed, and the SPI Flash contains your configuration data, you can test the configuration. Assert the PROGRAMN, transmit a REFRESH command, or cycle power to the board, and the MachXO2 will configure from the external SPI Flash.

Dual Boot Configuration Mode

Dual Boot Configuration Mode is a combination of Self Download Mode and Master SPI Configuration Mode. The MachXO2, when set up in Dual Boot Mode, tries to configure first from the internal Flash memory using SDM. If the SDM configuration fails, the MachXO2 attempts to configure itself using MSPI mode. The load order can't be reversed.

The internal data is the primary configuration data, and the external data is the golden configuration data. The primary image can fail in one of two ways:

- A bitstream CRC error is detected from on-chip Flash memory
- A time-out error is encountered when loading from on-chip Flash

A CRC error is caused by incorrect data being written into the internal Flash memory. Data is read from the Flash memory 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 external PROM.

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 will never become active. The MachXO2 counts the number of master clock pulses it has provided after the Power On Reset signal was released. When the count expires without DONE becoming active the FPGA attempts to get it's configuration data from the external PROM.

Dual boot configuration mode typically requires two configuration data files. One of the two configuration data files is a fail-safe image that is rarely, if ever, updated. The second configuration data file is a working image that is routinely updated. The working image is stored in the Configuration Flash, the fail-safe image is stored in the external SPI memory. 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.

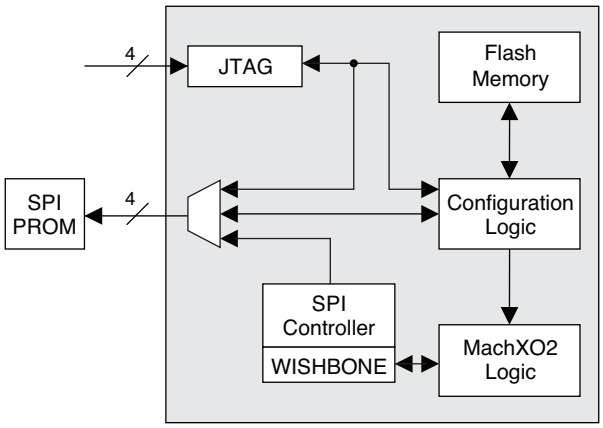
Use the following preferences to build a dual-boot design:

Preference	Dual-Boot Setting
CONFIGURATION	CFG | CFG_EBRUFM | CFGUFM
MASTER_SPI_PORT	ENABLE
GENERATE_BITSTREAM	ENABLE
COMPRESS_CONFIG	ON | OFF

Diamond creates a JEDEC file for the primary configuration data that is stored in the internal Flash memory. A BIT file is created for the golden configuration data that is stored in the external SPI Flash. The golden configuration data must be located in the external SPI Flash starting at address 0x010000. This differs from a single image Master SPI Configuration Mode, which requires the configuration data be stored at offset 0x000000.

To prevent the MachXO2 from using dual boot mode when using the User Master SPI controller set the MASTER_SPI_PORT preference to EFB_USER. This reserves the Master SPI configuration port pins and prevents dual-boot.

Figure 14-10. MSPI Mode



Slave SPI Mode (SSPI)

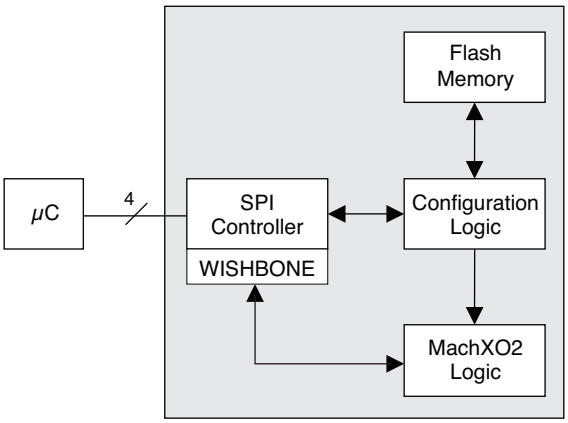
The MachXO2 provides a Slave SPI configuration port that allows you to access features provided by the Configuration Logic. You can reprogram the Configuration Flash, UFM and Feature Row, and access status/control registers within the Configuration Logic block. The Flash memories is done using either offline or transparent operations. The SRAM is not directly accessible using the Slave SPI port. It is necessary to send a REFRESH command to load a new Flash image into the SRAM.

Table 14-13. Slave SPI Port Pins

Pin Name	Description
CCLK	Configuration clock input that is driven by a SPI master controller.
SI	Serial Data Input to the MachXO2 Configuration Logic for command and data.
SO	Serial Data Output from the MachXO2 configuration logic.
SN	Chip select to enable the MachXO2 configuration logic.

In the Slave SPI mode, the MCLK/CCLK pin becomes CCLK (i.e. Configuration clock). Input data is read into the MachXO2 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 MachXO2 when SN is asserted. The MachXO2 SSPI port only accepts Mode 0 bus transactions to the Configuration Logic.

Figure 14-11. Slave SPI Configuration Mode



The SSPI port is active when the MachXO2 is in Default Mode Feature Row state. Diamond's default preference for the SLAVE_SPI_PORT is to DISABLE the port. Use the Spreadsheet View to ENABLE the SLAVE_SPI_PORT preference in your design to keep the SSPI port active 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 MachXO2 is using External or Dual Boot configuration mode because the REFRESH operation will not begin until SN is deasserted.

Programming the MachXO2 using the SSPI port is complex. Lattice provides 'C' source code called SSPIEmbedded to insulate you from the complexity of programming the MachXO2. It is recommended that SSPIEmbedded be used when you want to reprogram the MachXO2 Flash memory.

In addition to reprogramming the Flash memory the SSPI port can be used to access several status and control registers in the MachXO2. A list of the available commands and information about the registers is described in TN1205, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices. Accessing the status registers is less complex and does not require the use of the SSPIEmbedded code.

I²C Configuration Mode

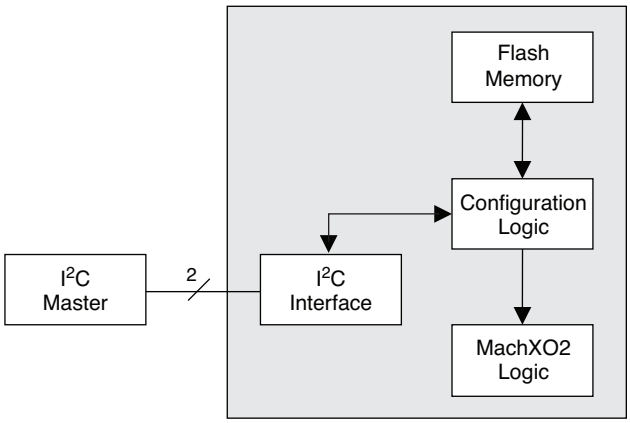
The MachXO2 has an I²C Configuration port for use in accessing the configuration logic. An I²C master can communicate to the configuration logic using 10-bit or 7-bit addressing modes. The I²C SCL input can accept a clock frequency up to 400KHz. You can reprogram the Configuration Flash, UFM and Feature Row, and access status/control registers within the configuration logic block. Reprogramming the Flash memories can be done either offline or in transparent operations. You cannot directly update the configuration SRAM. It is necessary to send a REFRESH command after reprogramming the Flash memory in order for the configuration SRAM to be updated.

Table 14-14. I²C Port Pins

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

The I²C Configuration port is available when the MachXO2 is in Default Mode Feature Row 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 in order to recover from erroneously disabling the I²C port.

Figure 14-12. I²C Configuration Logic



There are two hardened I²C controllers in a MachXO2 device, a primary and a secondary. The primary controller provides an interface to the MachXO2 Configuration Logic, and access to Wishbone registers. Access to the Wishbone registers is referred to as User Mode I²C. The primary I²C controller is the only one that permits access to the Configuration Logic. The Secondary I²C controller is always a User Mode I²C controller.

When the MachXO2 is in Default Mode Feature Row state the I²C port is enabled, and you may interact with the primary I²C controller. Whenever the I²C port is enabled access to the Configuration Logic is possible. It is not necessary to instantiate the EFB to gain access to the Configuration Logic.

The Primary I²C controller provides access to the Configuration Logic when:

- The MachXO2 is in Default Mode Feature Row state
- The EFB is not instantiated, and the I²C port pins are in the ENABLE state
- The EFB is instantiated, and the I²C port pins are in the ENABLE state

An external I²C master accesses the Configuration Logic using address 1000000 (7-bit mode) or 1111000000 (10-bit mode) unless the EFB I²C base address has been 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 in order to change the address. The address is shared by the Primary and Secondary I²C controllers.

Table Table 14-15 shows the address decoding used to access the I²C resources in the MachXO2.

Table 14-15. Slave Addresses for I²C Ports

Slave Address	I²C Function
yyyxxxxx**00**	Primary I²C Controller Configuration Logic address. Always responds to 7-bit or 10-bit addresses.
yyyxxxxx**01**	User Mode Primary I²C Controller address.
yyyxxxxx**10**	User Mode Secondary I²C Controller address.
yyyxxxxx**11**	Primary I²C Configuration Logic Reset. Always responds to 7-bit or 10-bit addresses.

The Primary I²C core can be used for accessing the User Flash Memory (UFM) and for programming the Configuration Flash. However, the Primary I²C port cannot be used for both UFM/Configuration access and User functions in the same design. The operation of the User Mode Primary and User Mode Secondary I²C controllers is described in TN1205, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices. Interacting with these I²C slave devices is not covered in this document.

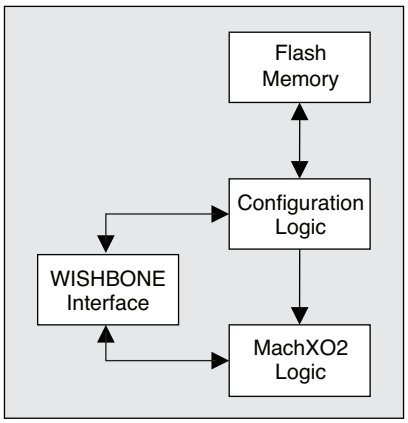
The fourth I²C resource in the MachXO2 is located at offset 3. In some instances an I²C memory transaction to the configuration logic may be interrupted or abandoned. It is possible for a command to be accepted by the configura-

tion 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 I^2C 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 will reset, any queued data will be flushed, and subsequent I^2C memory transactions to the Configuration Logic will operate correctly.

WISHBONE Configuration Mode

The MachXO2 can access the Configuration Flash, User Flash Memory, and the Feature Row from an internal WISHBONE bus. To use the WISHBONE bus the Embedded Function Block must be inserted into your design. You design logic to interface to the EFB and then perform WISHBONE bus transactions to access resources attached to the configuration logic.

Figure 14-13. WISHBONE Configuration Mode



The MachXO2 must be in user mode in order to access the WISHBONE interface. Accessing and updating the resources made available by the configuration logic must be done in Transparent mode. Attempting accesses to the configuration logic in offline mode causes a deadlock because the MachXO2 leaves user mode.

Appendix C describes how you can perform erase, program, and verify functions using the WISHBONE interface. You can get more detailed information about the MachXO2 WISHBONE interface by reading TN1205, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices.

JTAG Mode

The JTAG port is the most flexible configuration and programming port available on the MachXO2. The JTAG provides:

- Offline Flash memory programming
- Transparent Flash memory programming
- Offline SRAM configuration
- Full access to the MachXO2 Configuration Logic
- Device chaining
- IEEE 1149.1 testability
- IEEE 1532 Compliant programming

The JTAG port is available when the MachXO2 is in Default Mode Feature Row state. The port is enabled by default by Diamond 1.4. The MachXO2 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. See "sysCONFIG Pins" on page 14-10 for details on recovering the JTAG port pins for use as general purpose I/O.

The MachXO2 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 MachXO2 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 MachXO2 to be combined in a chain of other programmable logic.
- **Reveal Debug**: The Lattice Reveal debug tool is an embed-able logic analyzer tool. It allows you to analyze the logic inside the MachXO2 in the same fashion as an external logic analyzer permits analysis of board level logic. Reveal access is only available via the MachXO2 JTAG port.
- **SRAM Readback**: The JTAG port is the only sysCONFIG port able to directly access the MachXO2's configuration SRAM. It is occasionally necessary to perform failure analysis for SRAM based FPGAs. A key component to failure analysis can involve reading the configuration SRAM. This kind of failure analysis is lost when the JTAG port is not enabled.
- **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 MachXO2 JTAG port is vital for boundary scan testability. Lattice provides Boundary Scan Description Language files for the MachXO2 on the Lattice website.

TransFR Operation

The MachXO2, like other Lattice FPGAs, provides for the TransFR™ capability. TransFR is described in TN1087 Minimizing System Interruption During Configuration Using TransFR Technology. The MachXO2 operates differently than earlier generations of Lattice FPGAs. Earlier generations depend solely on the JTAG port for TransFR operation. The JTAG port uses the BSCAN cells to either capture the current state on the I/O, or to force the I/O to a known state.

The MachXO2 allows TransFR programming to occur on two other ports. The I^2C, and SSPI configuration ports can perform programming operations, but do not have access to the JTAG BSCAN cells. In order to make TransFR available for the I^2C and SSPI programming ports a new TransFR process was created.

The MachXO2 enables the TransFR feature using configuration SRAM bits. In order for TransFR to operate correctly the configuration active in the SRAM must have the ENABLE_TRANSFR set to the ENABLE state. The new configuration data being programmed into the MachXO2 must also have the ENABLE_TRANSFR set to the ENABLE state. When both of these conditions are met the MachXO2 preserves the I/O personality and voltage levels when a Refresh command is issued. The MachXO2 latches and holds the state of the output buffer at the INITN falling edge and maintains the state on the outputs until the GOE strobe is asserted during the Wake Up process. Input data is used as soon as the internal DONE strobe is asserted during the Wake Up process.

In the event the MachXO2 fails to configure, the outputs remain held pending the GOE assertion in the Wake Up process. The outputs, as a result, remain in a known state should the MachXO2 perform a Dual Boot fail safe load. This behavior is superior to the Boundary Scan method used in prior device families.

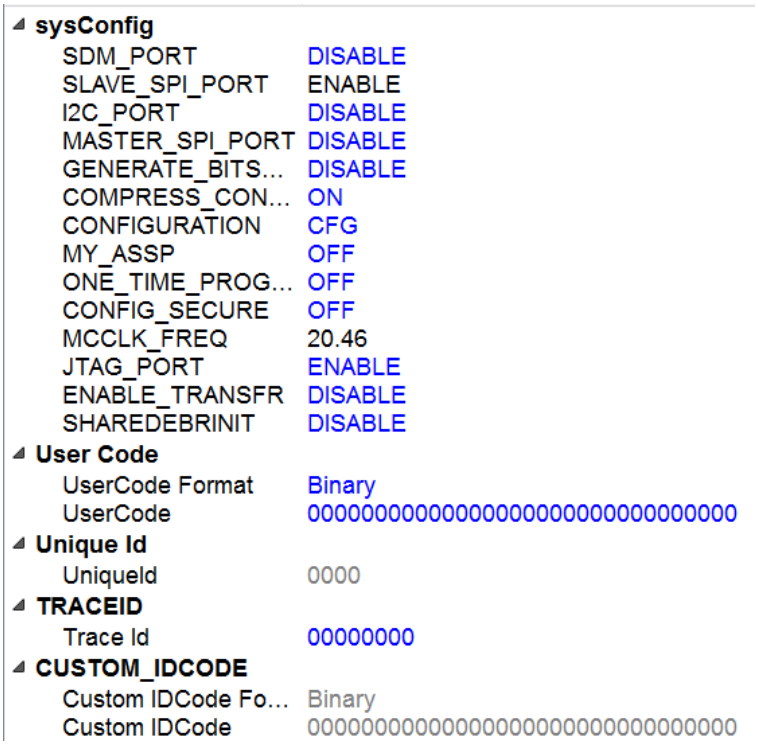
Software Selectable Options

The operation of the MachXO2 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 MachXO2 uses the non-volatile Feature Row to select how it will configure. The Feature Row's 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 MachXO2 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 14-14. The sysCONFIG preferences are divided into three categories:

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

Figure 14-14. sysCONFIG Preferences in Global Preferences Tab, Diamond Spreadsheet View



Configuration Mode and Port Options

The configuration and port options allow you to decide which configuration ports continue to operate after the MachXO2 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 14-16. Configuration Mode/Port Options

Option Name	Default Setting	All Settings
JTAG_PORT	ENABLE	DISABLE, ENABLE
SLAVE_SPI_PORT	DISABLE	DISABLE, ENABLE
MASTER_SPI_PORT	DISABLE	DISABLE, ENABLE, EFB_USR
I2C_PORT	DISABLE	DISABLE, ENABLE
SDM_PORT[1, 2]	DISABLE	DISABLE, PROGRAMN, PROGRAMN_DONE, PROGRAMN_DONE_INITN
MCCLK_FREQ	2.08	See description below
ENABLE_TRANSFR	DISABLE	DISABLE, ENABLE

1. The default for SDM_PORT was PROGRAMN in ispLEVER 8.1 SP1 and Diamond 1.1.
2. The 32 QFN package does not have an INITN pin. Because of this, the option SDM_PORT = PROGRAMN_DONE_INITN is not available.

JTAG Port

The JTAG_PORT preference allows you to decide how the JTAG configuration port pins operate when the MachXO2 device is in user mode. There are two states the JTAG_PORT can be set to:

- **ENABLE** – In this mode the JTAG I/O are dedicated and provide an IEEE 1149.1 JTAG interface
- **DISABLE** – In this mode the JTAG I/O pins are controlled dynamically using the JTAGENB pin

The JTAGENB pin is only available when the JTAG_PORT is in the DISABLE state. JTAGENB, when asserted high, makes the four JTAG I/O act as an IEEE 1149.1 JTAG port. JTAGENB driven low causes the four I/O to be available for use as general purpose I/O.

Lattice recommends designing so that the JTAG port can be accessed in the event reprogramming the MachXO2 disables your primary configuration port.

Slave SPI Port

The SLAVE_SPI_PORT allows you to preserve the Slave SPI configuration port after the MachXO2 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 MachXO2 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.

Master SPI Port

The MASTER_SPI_PORT allows you to preserve the SPI configuration port after the MachXO2 device enters user mode. There are three states to which the MASTER_SPI_PORT preference can be set:

- **ENABLE** – This setting preserves the SPI port I/O when the MachXO2 is in user mode. This preference makes External or Dual Boot configuration modes active. Using this preference in combination with CONFIGURATION = EXTERNAL enables external boot mode. This preference in combination with CFG, CFG_EBRUFM, CFGUFM enables Dual Boot mode. After entering user mode, the SPI controller in the EFB has access to the SPI port for performing SPI bus transactions. The preference also prevents you from over-assigning I/O to the port pins.
- **EFB_USER** – This setting preserves the SPI port I/O when the MachXO2 is in user mode. After entering user mode, the SPI controller in the EFB has access to the SPI port for performing SPI bus transactions. 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 MASTER_SPI_PORT can be enabled at the same time as the SLAVE_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.

I^2C Port

The I2C_PORT allows you to preserve the I^2C configuration port after the MachXO2 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 MachXO2 is in user mode. When the pins are pre-

served, 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.

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

SDM Port

The SDM_PORT allows you to select the programming status pins after the MachXO2 device enters user mode. There are four 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 MachXO2 device is in user mode. Asserting this pin active low causes the MachXO2 device to reconfigure. The DONE and INITN pins are general purpose I/O.
- **PROGRAM_DONE** – This setting preserves the PROGRAMN and DONE pins when the MachXO2 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 modes. The PROGRAMN pin is the only way to perform a “warm” reconfiguration of the MachXO2 device, unless another configuration port is available to transmit a REFRESH command.

MCCLK Frequency

The MCLK_FREQ preference allows you to alter the MCLK frequency used to retrieve data from an external SPI Flash when using EXTERNAL or Dual Boot configuration modes. The MachXO2 uses a nominal 2.08MHz (+/- 5.5%) clock frequency to begin retrieving data from the external SPI Flash. The MCLK_FREQ value is stored in the incoming configuration data. It is not stored in the Feature Row. The MachXO2 device reads a series of padding bits, a “start of data” word (0xBDB3) and a control register value. The control register contains the new MCLK_FREQ value. The MachXO2 switches to the new clock frequency shortly after receiving the MCLK_FREQ value. The MCLK_FREQ has a range of possible frequencies available from 2.08 MHz up to 133 MHz (see Table 14-9). Take care not to exceed the maximum clock rate of your SPI Flash, or of your printed circuit board.

Lattice recommends having a back-up configuration port available in the event you specify a clock frequency that is out of specification.

ENABLE_TRANSFR

The TransFR function used by the MachXO2 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 "TransFR Operation" on page 14-26, and TN1087 Minimizing System Interruption During Configuration Using TransFR Technology for more information about using TransFR with the MachXO2.

Bitstream Generation Options

The Bitstream Generation options allow you to decide how the Diamond development tools create the configuration data for the MachXO2 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. Table 14-17 gives a summary of these options.

Table 14-17. Bitstream Generation Options

Option Name	Default Setting	All Settings
GENERATE_BITSTREAM	DISABLE	DISABLE, ENABLE
COMPRESS_CONFIG	ON	OFF, ON
CONFIGURATION[1]	CFG	CFG, CFG_EBRUFM, CFGUFM, EXTERNAL
USERCODE_FORMAT	BINARY	HEX, BINARY, ASCII
USERCODE	<all zero>	32-bit arbitrary
CUSTOM_IDCODE_FORMAT	BINARY	HEX, BINARY
CUSTOM_IDCODE	<all zero>	32-bit arbitrary
SHAREDEBRINIT	DISABLE	DISABLE, ENABLE
MUX_CONFIGURATION_PORTS	DISABLE	DISABLE, ENABLE

1. The default was CFG_EBRUFM in ispLEVER 8.1 SP1 and Diamond 1.1.

GENERATE BITSTREAM

The GENERATE_BITSTREAM preference informs Diamond to create a BIT file in addition to a JEDEC file. The default setting for GENERATE_BITSTREAM, DISABLE, prevents Diamond from creating a BIT file. The GENERATE_BITSTREAM, in the ENABLE state, causes Diamond to build both a JEDEC file and a BIT file. The BIT file is a binary data file that can be programmed directly into an external SPI Flash.

Bitstream generation is affected by the COMPRESS_CONFIG preference. Read the COMPRESS_CONFIG section for additional information.

COMPRESS_CONFIG

The COMPRESS_CONFIG preference alters the way JEDEC and BIT files are generated. The COMPRESS_CONFIG default setting is to be ON. There are three legal combinations for the COMPRESS_CONFIG and GENERATE_BITSTREAM preferences, shown in Table 14-18.

Table 14-18. Valid GENERATE_BITSTREAM/COMPRESS_CONFIG Combinations

COMPRESS_CONFIG	GENERATE_BITSTREAM	JEDEC Generated	
ON	OFF	Valid[1]	Yes
ON	ON	Valid	Yes
OFF	OFF	Invalid	N/A
OFF	ON	Valid	No

1. Default setting for COMPRESS_CONFIG and GENERATE_BITSTREAM.

JEDEC files, when they are built, are always compressed. Lattice recommends using COMPRESS_CONFIG=ON and GENERATE_BITSTREAM=ON when using External or Dual Boot configuration modes. The configuration time will be slightly reduced when reading configuration data from the external PROM and the Diamond tool will create a JEDEC file you can program into the internal Flash memory.

CONFIGURATION

The CONFIGURATION preference allows you to control the Configuration Flash and UFM sectors. The CONFIGURATION preference has four possible settings:

- **CFG** – The CFG preference is the default mode for building configuration data. The configuration bitstream is stored in the Configuration Flash and is not permitted to overflow into the UFM sector. The configuration data includes EBR initialization data. The UFM sector is available for your use as general purpose Flash memory in user mode.
- **CFG_EBRUFM** – This preference creates configuration data that is stored in the Configuration Flash. EBR initialization data is stored in the lowest page addresses of the UFM sector. The UFM sector is available in user

mode. You must restore the EBR initialization data when making changes to the UFM to guarantee correct operation.

- **CFGUFM** – This preference creates configuration data that is stored in the Configuration Flash. This mode differs from CFG by allowing the configuration data to overflow into the UFM. The configuration data increases in size as EBR initialization data is added to the design. This mode prevents write access to UFM in user mode.
- **EXTERNAL** – This preference generates configuration data that is stored in an external memory. The UFM sector is available as general purpose Flash memory in user mode.

The CONFIGURATION preference defaults to the CFG state in the current release of the Diamond software. The Diamond design software only generates JEDEC files when your entire design fits within the Configuration Flash memory. The UFM is guaranteed to be available when the MachXO2 device enters user mode.

In the event the configuration data does not fit in the Configuration Flash memory try using the CFG_EBRUFM preference. This preference works well if your design has a significant amount of initialized EBR, and you still want access to UFM pages to store data. Depending on the amount of initialized EBR the UFM may still have sufficient space available for storing your data.

Use the CFGUFM option when the Configuration Flash is not large enough to store the configuration data, and you do not need to use UFM for storing your own data. It is possible, in rare instances, for the size of the configuration data to exceed the combined space of the Configuration Flash and UFM.

Use the EXTERNAL preference to build configuration data for use with Master SPI Configuration Mode. When the configuration data exceeds the combined space available in the Configuration Flash and UFM it is necessary to switch to EXTERNAL mode. EXTERNAL mode does not use any Configuration Flash or UFM resources. The UFM is available for your use in user mode. The GENERATE_BITSTREAM option must be in the ENABLE state when EXTERNAL is selected.

The MachXO2-256 device does not contain any UFM. The only configuration options available to this device are the CFG and EXTERNAL modes.

USERCODE

The MachXO2 Configuration Flash sector contains a 32-bit register for storing a user-defined value. 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.

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 (i.e., 0-9A-F)
- **ASCII** – USERCODE is set using up to four ASCII characters

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 "MY_ASSP" on page 14-32 for more information about how to assign a value to the CUSTOM_IDCODE preference.

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 (i.e., 0-9A-F)

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

MUX_CONFIGURATION_PORTS
The MUX_CONFIGURATION_PORTS is used in the event that all configuration ports are disabled. Disabling all of the available configuration ports turns the MachXO2 into a "write one time" device. MUX_CONFIGURATION_PORTS confirms the removal of all configuration ports. The control is only active when all of the other configuration ports are set to the DISABLE state. MUX_CONFIGURATION_PORTS set to the ENABLE state enables the JTAGENB input pin, permitting the JTAG port pins to be multiplexed. Setting MUX_CONFIGURATION_PORTS to the DISABLE state causes the Diamond build tools to honor the removal of all other configuration ports, allowing the MachXO2 to become a "write one time" device.

Security Options

The Security Options allow you to select from a range of options for tracking or securing the MachXO2 device. Table 14-19 provides a summary of these options.

Table 14-19. Security Options

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

TRACEID
The MachXO2 introduces a new feature called TraceID. TraceID stamps each MachXO2 with a unique 64-bit ID. No two MachXO2 devices will 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 TN1207, Using TraceID in MachXO2 Devices.

MY_ASSP
Every Lattice device has its own identification code identifying the device family, device density, and other parameters (e.g. voltage, device stepping, etc.). The code is accessible from any MachXO2 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 the MY_ASSP ON enables the CUSTOM_IDCODE preference.

CUSTOM_IDCODE
The CUSTOM_IDCODE is the value you assign to override the default IDCODE in the MachXO2 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 MachXO2 device, and as a result, prevents Programmer from being able to directly program the MachXO2 device. It is necessary to migrate to generating Serial Vector Format (SVF) files in order to program MY_ASSP enabled MachXO2 devices.

CONFIG_SECURE
When this preference set to ON, the read-back of the SRAM memory and the Configuration Flash memory are blocked. The read-back of the UFM will also be blocked if the bitstream overflows into the UFM block. The MachXO2 device cannot be read back, nor can it be programmed without erasing. The device must be erased in order to reset the security setting. The CONFIG_SECURE fuse and the Configuration Flash are erased in tandem. Once the security fuses are reset, the device can be programmed again.

ONE_TIME_PROGRAM
The MachXO2 has One Time Programmable (OTP) fuses that can be used to prevent the on-chip memory from being erased or programmed. The MachXO2 device has three OTP security fuses, one for each of the following memory sectors: SRAM, Configuration Flash, and UFM. This preference provides options to set the OTP security for each memory sector.

- **FLASH** – The Configuration Flash cannot be erased or programmed
- **FLASH_UFM** – The Configuration Flash and UFM cannot be erased or programmed
- **FLASH_UFM_SRAM** – The Configuration Flash, UFM, and SRAM cannot be erased or programmed

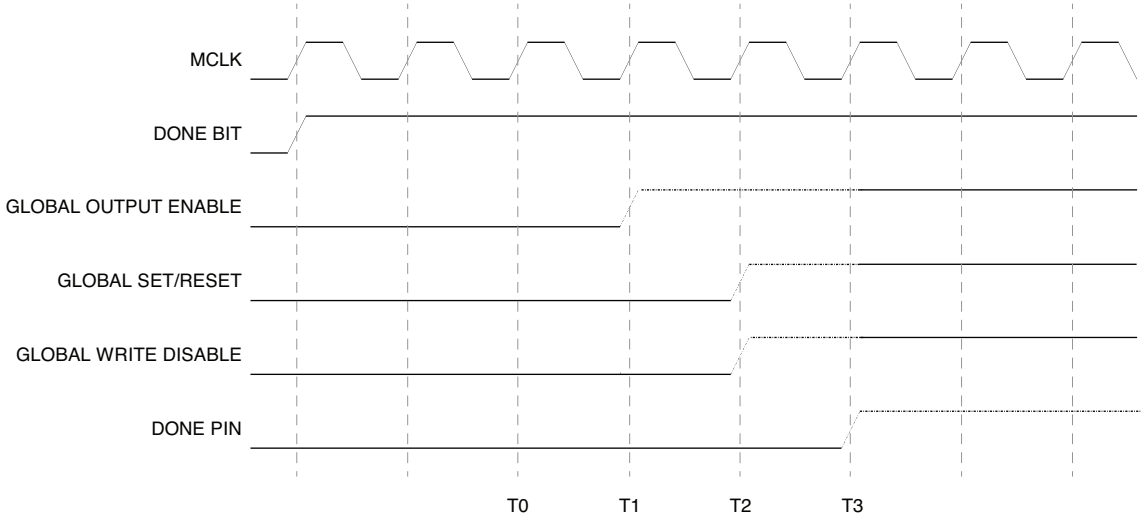
Once the ONE_TIME_PROGRAM preference is set for the Flash memory, the on-chip Flash memory cannot be erased or programmed. The configuration data is prevented from further modification, but the SDM mode can still be used to configure the device.

When the ONE_TIME_PROGRAM preference is set for the FLASH_UFM_SRAM memory, the device acts like an ASIC. You are no longer able to reprogram the internal Flash or UFM, and the SRAM cannot be changed from the JTAG port. Configuration of SRAM from on-chip Flash memory or external SPI Flash is still enabled.

Device Wake-up Sequence

When configuration is complete (the SRAM has been loaded), the device will wake up in a predictable fashion. If the MachXO2 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 complete, the internal DONE bit will be set and then the wake-up process will begin. Figure 14-15 shows the wake-up sequence using the internal clock.

Figure 14-15. Wake-up Sequence Using Internal Clock



Wake-up Signals

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

- 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.
- 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.
- When low, GOE prevents the device's I/O buffers from driving the pins. The GOE only controls **output** pins. Once the internal DONE is asserted the MachXO2 will respond to input data.
- When high, the DONE pin indicates that configuration is complete and that no errors were detected.

Wake-up Clock Selection

The clock source used to complete the four state transitions in the wake-up sequence is user-selectable. Once the MachXO2 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. The clock must be supplied on an external input pin, because the MachXO2 does not begin internal operations until the Wake-up sequence is complete. 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>);
```

Advanced Configuration Information

Flash Programming

The MachXO2's internal Flash memory is the heart of the FPGA's configuration system. It is flexible, allowing you to store the FPGA's configuration data, as well as storing design specific data in the User Flash Memory. It is also a resource that uses a precise erase and programming sequence. Lattice provides several methods for programming the MachXO2 Flash memory:

- **Diamond Programmer**: 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 TN1246, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices Reference Guide, permits creation of a custom solution.

The Flash memory space can be accessed by the JTAG port, I^2C port, SPI port, or through the WISHBONE bus. These configuration ports may use offline or transparent programming modes to erase, program, and verify the MachXO2 Flash memory resources. The WISHBONE 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 MachXO2 contains a certain quantity of Configuration Flash memory and User Flash Memory. The amount of memory depends on the device density of the MachXO2. Figure 14-20 shows the number of Flash memory pages available for each MachXO2 device density. Each page represents 128 bits of data.

Table 14-20. Number of Pages of Flash Memory for the MachXO2 Family

MachXO2 Device Density	Configuration Flash (Pages)	UFM (Pages)	CFG + UFM Bridged[1] (Usable Pages)
7000	9,212	2,048	11,257
4000, 2000U	5,758	768	6,524
2000, 1200U	3,198	640	3,836
1200, 640U	2,175	512	2,686
640	1,151	192	1,342
256	575	0	575

1. CFG+UFM (CONFIGURATION = CFGUFM) page count may be less than the sum of its parts due to device limitations.

MachXO2 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 (hex or binary). The relevant detail about the JEDEC file is provided in the table below for completeness.

Table 14-21. MachXO2 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 will be 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's JEDEC files add "OTE" to the N key word to make it a more meaningful word NOTE.
Fuse Count	QF3627736	The key word QF identifies the total real fuse count of the device[1].
Default Fuse State	F0 or F1	The key word F identifies the fuse state of those fuses not included in the link field. F0 = fill them with zeros (0), F1 = fill them with ones (1). It is defined for the purpose of reducing JEDEC file size. It has no meaning in Lattice's 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>
OTP and Security Setting	G0, G1, G2, or G3	Lattice enhances the G field to cover OTP fuse programming as well. G<0=both no, 1=only security yes, 2=only OTP yes, 3=both yes>.

Table 14-21. MachXO2 JEDEC File Format (Continued)

JEDEC Field	Syntax	Description
Link Field	L0000000 101011…100011 …….. 111111…101100 110 101011…100011 …….. 111111…101100 110 …….. …….. 101011…100011 …….. 111111…101100 110* NOTE SED_CRC* L3627704 111111….111111* CC1B9	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 zeros. 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 Flash 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 TAG DATA*” line. If the JEDEC file is encrypted, all the data in the link field are encrypted. The column size will increase accordingly to include filler bits to make the column size packet (128-bit, or 16 bytes, per packet) bounded.
Fuse Checksum	CC1B9	The checksum of all the fuses = Fuse count. The fuse state of all the fuses can be 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 can be found on 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[2].
E Field	EH 012..ABCDEF	JEDEC standard defines this field to hold the architecture fuses. Lattice uses this field to store the Feature Row and FEABITS. The Feature Row data is on the first line. The FEABITS values are on line 2.
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.

1. For encrypted JEDEC file, the first sixteen (16) bits of USERCODE is the CRC value calculated from of the row 0 only; the second sixteen (16) bits is the CRC value calculated from row 0 to the last row.

An example of a MachXO2 unencrypted JEDEC file is shown in Figure 14-16.

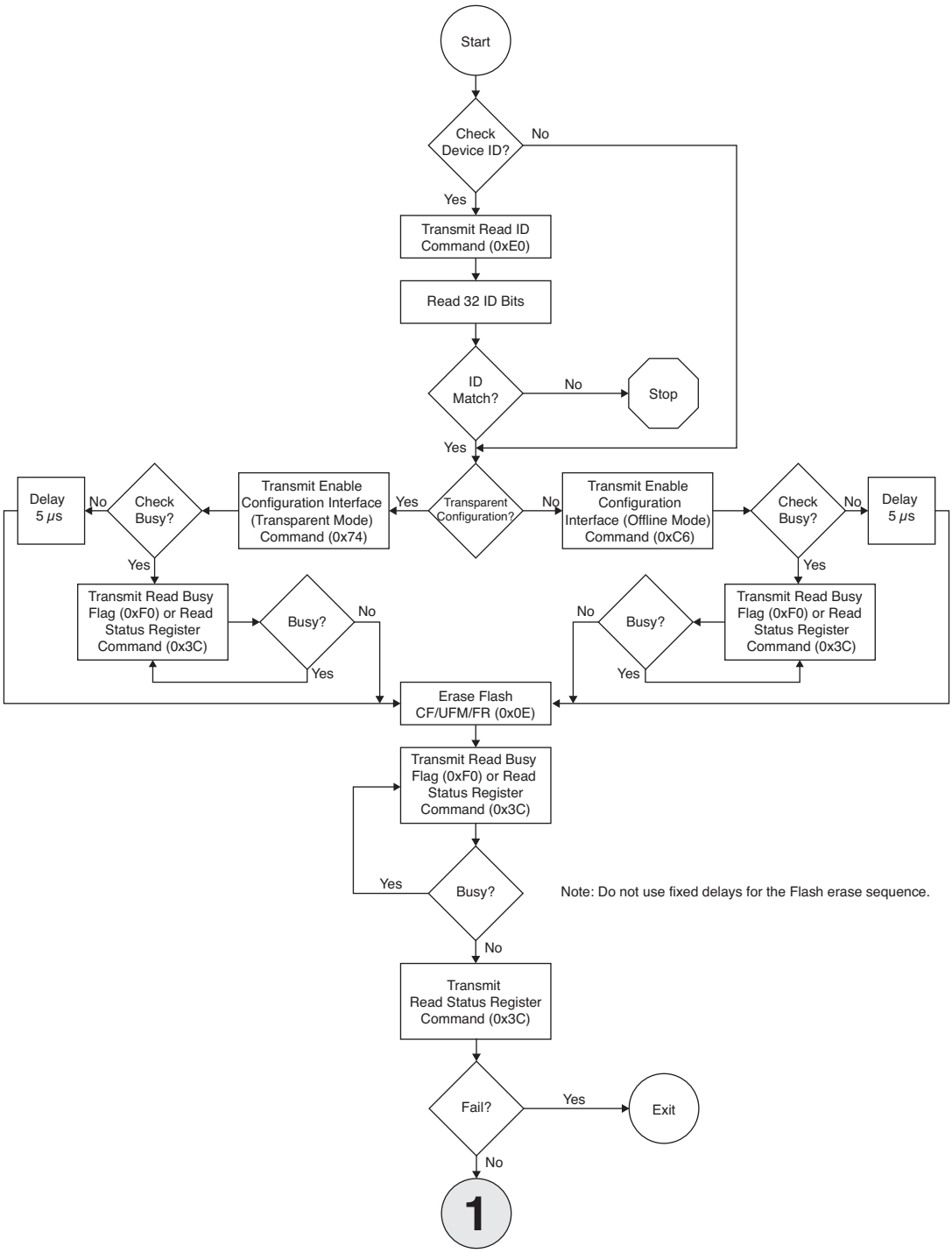
Figure 14-16. Unencrypted JEDEC File Example

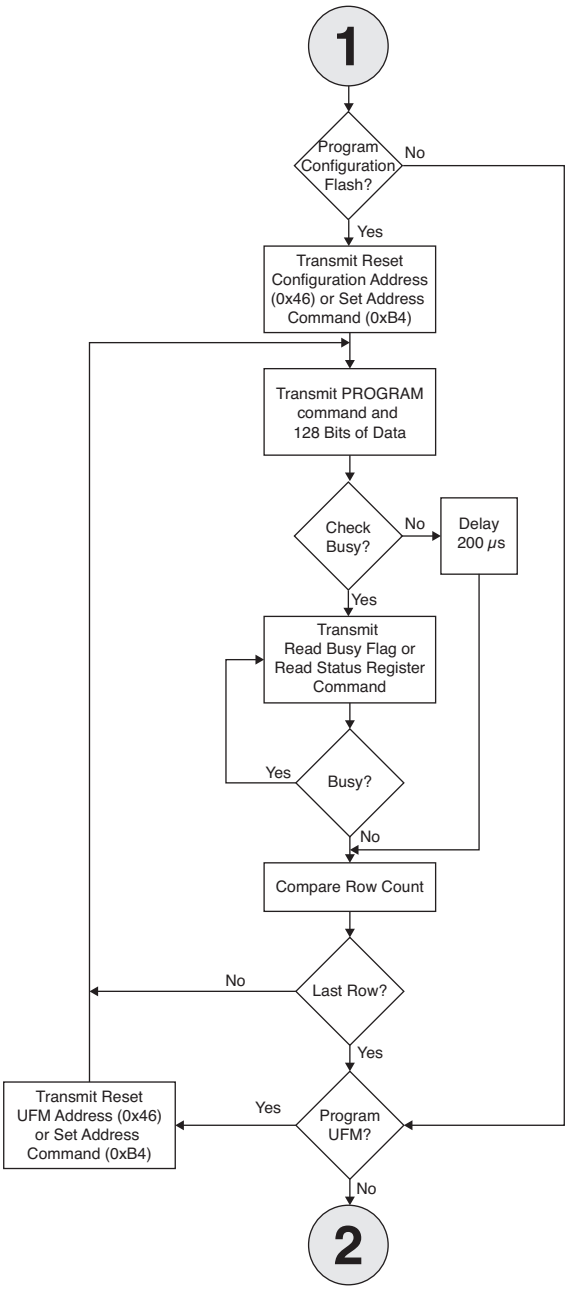
```
*
NOTE Diamond_1.2_Production (92) JEDEC Compatible Fuse File.*
NOTE Copyright (C), 1992-2010, Lattice Semiconductor Corporation.*
NOTE All Rights Reserved.*
NOTE DATE CREATED:Fri Dec 02 14:50:08 2011*
NOTE DESIGN NAME:control_SoC_demo_impl1.ncd*
NOTE DEVICE NAME:LCMXO2-1200ZE-1CSBGA132ES*
NOTE TRACEID 0x84*
NOTE LVDS_72P M9*
NOTE LVDS_72N N10*
NOTE LVDS_68P P8*
NOTE LVDS_68N M8*
NOTE PIN ASSIGNMENTS*
NOTE PINS cap_btn1 : M10 : inout*
NOTE PINS LCD_COM0 : B14 : out*
NOTE PINS xout : N7 : out*
...
NOTE PINS xin : M5 : in*
NOTE PINS Icc_analog_cmp_p : P8 : in*
QP132*
QF343936*
G0*
F0*
L000000
11111111111111111011110110110011111111111111111110011101100000000000000000000000000000001000000000000000000000000000001100000000110
01000010010100001001000001001000000010011111111101000110000000000000000000000000010111000111000000000000101001101000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000010001000100010000000000001000100000000000001000100000000000010001000000000001000100010001000000010001000000010001000000000
01000100010001000000100010000000000000000001000100000000000000000000000000000000000000000000100010001001000000000000001001000000000
00000010010000000000000100100000000001000100010010000000100100000001001000000000010001000100100000000100100000000000000000000010010
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
...
00000000000000000000000000000000000000000101111010110101011111111111111111111111111111111111111111111111111111111111111111101110010
00010000010000000000010111000000000000000000110000000000000000000111001000010000010101000000010100000000010000000000000000100001001
11111110000000000000000000000000000000000000000110000101000000000000000000000001100101011111110101110101011111011110001010001000
00100010000000000000000000000000011000000000000000000000000000000000001111111111111111111111111111111111111111111111111111111111111
*
NOTE EBR_INIT DATA*
L137984
11111111111111111111111111111111111110110000000000000000000000000000000000000000000000011000000000001011001000010000000000000010000000
00011110100011110100011110100011110100011110100011110100011110100011110100011110100011110100011110100011110100011110100011110100
01111010001111010001111010001111010001111010001111010001111010001111010001111010001111010001111010001111010001111010001111010001
...
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00100010000000000000000000000000000000000000000000000000000000010111100000000000000000000000000011111111111111111111111111111111
*
NOTE END CONFIG DATA*
L184832
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
...
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
*
NOTE TAG DATA*
L343808
01000100010001000000100010000000000000000001000100000000000000000000000000000000000000000000100010001001000000000000001001000000000
00000010010000000000000100100000000001000100010010000000100100000001001000000000010001000100100000000100100000000000000000000010010
00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
*
C5CC8*
NOTE FEATURE_ROW*
E0000000000000000000000000000000000100001000010000000000000000000
0000010000000000*
NOTE User Electronic Signature Data*
UHCAFEBABE*
 6243
```

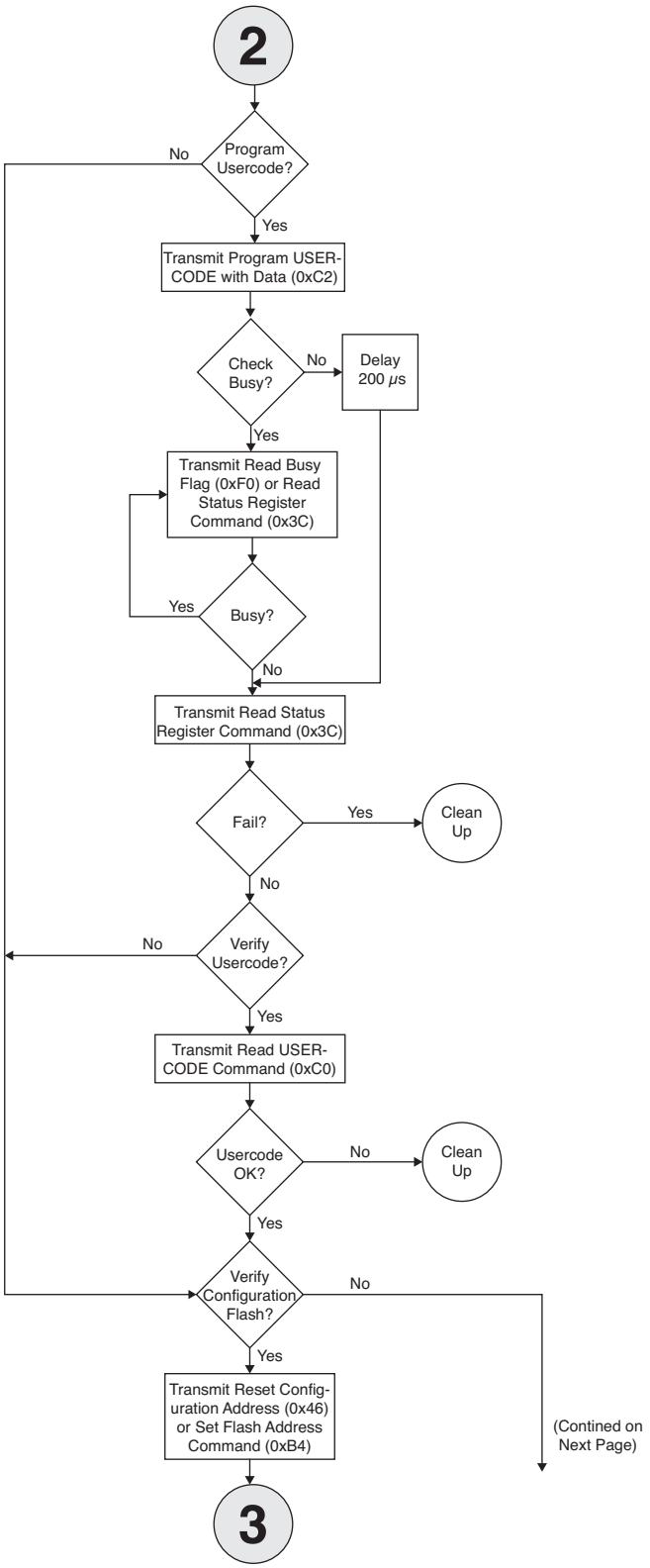
MachXO2 Flash Memory Programming Flow

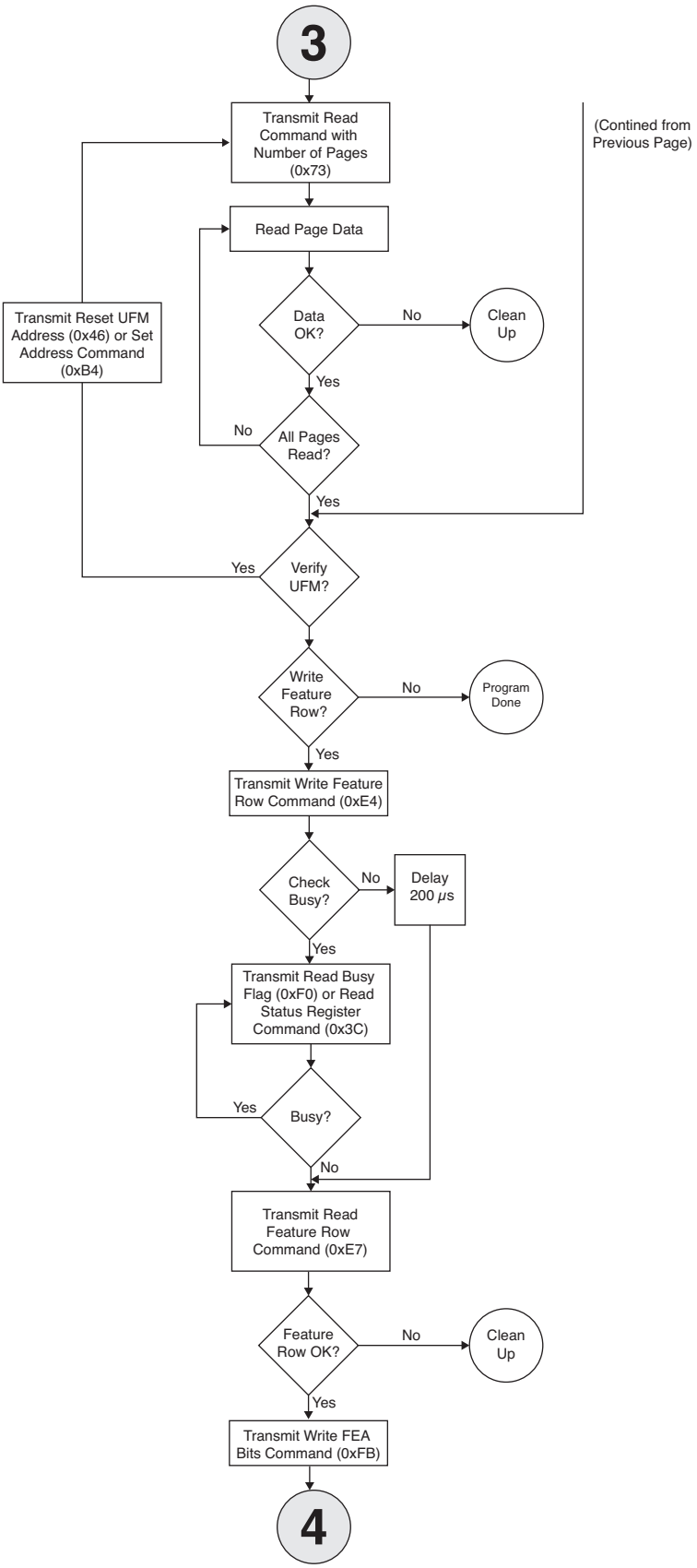
The MachXO2 Flash memory erasure, and programming requires a specific set of steps and timing. The flow chart in this section describes the command sequences and the timing required for successful Flash programming. The commands and timing are common between all of the configuration ports. There are some minor variations in the protocol, but not the timing, based on the configuration port used. Exceptions are described in the configuration port specific sections.

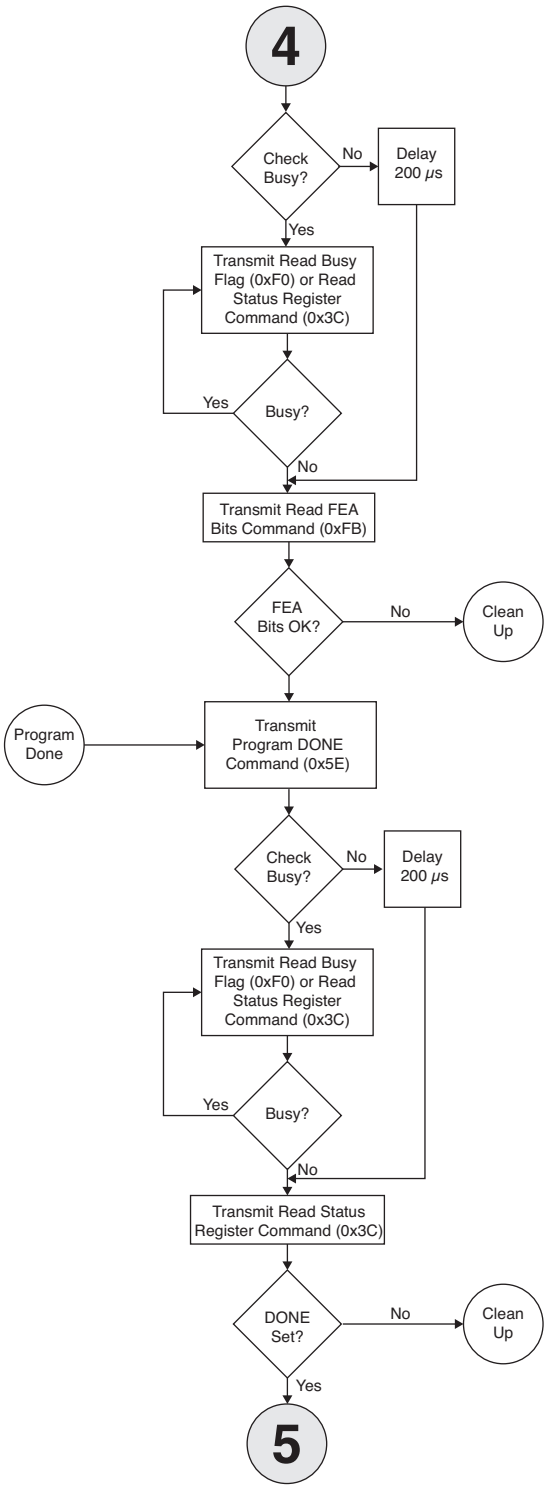
Figure 14-17. MachXO2 Flash Memory Programming Flow

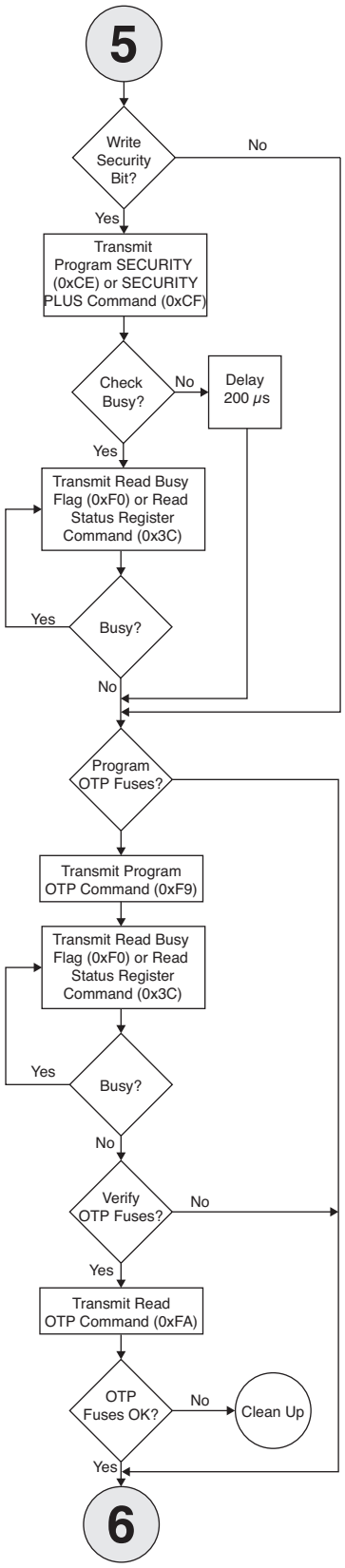


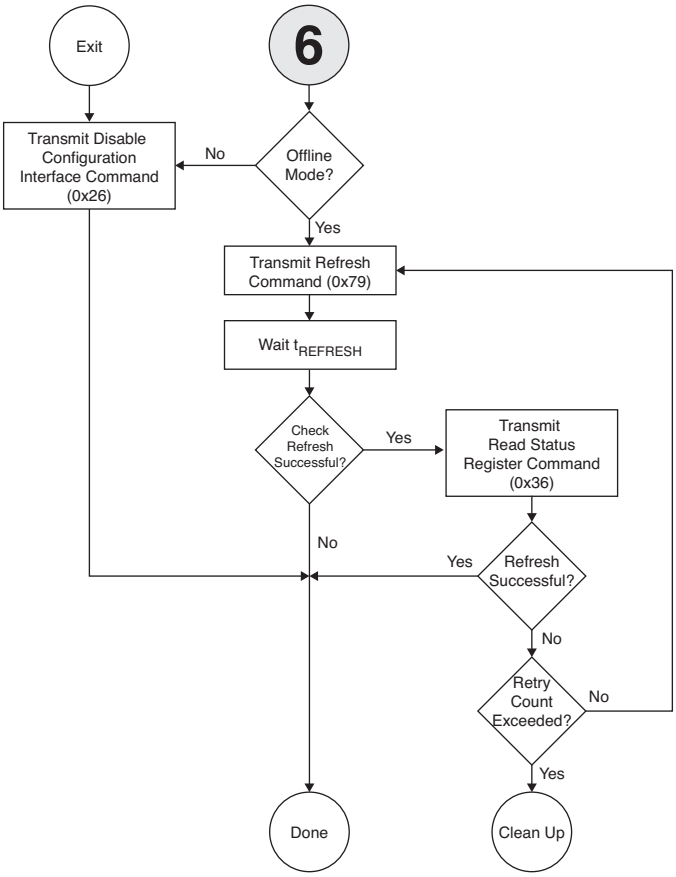

1
Program Configuration Flash?
No
Yes
Transmit Reset Configuration Address (0x46) or Set Address Command (0xB4)
Transmit PROGRAM command and 128 Bits of Data
Check Busy?
No
Delay 200 µs
Yes
Transmit Read Busy Flag or Read Status Register Command
Yes
Busy?
No
Compare Row Count
No
Last Row?
Yes
Transmit Reset UFM Address (0x46) or Set Address Command (0xB4)
Yes
Program UFM?
No
2

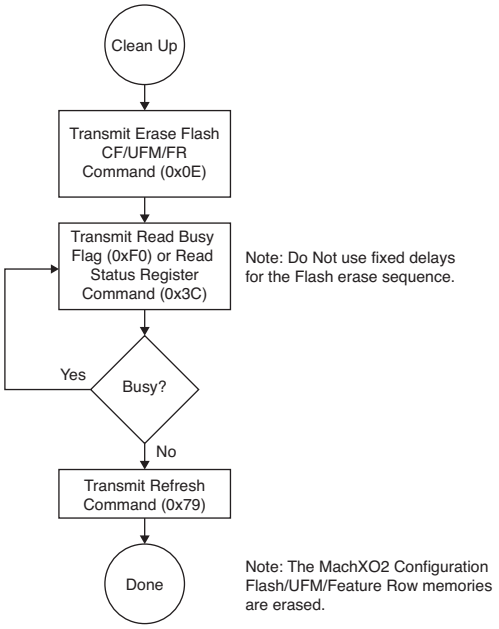

2
Program Usercode?
No
Yes
Transmit Program USER-CODE with Data (0xC2)
Check Busy?
No
Delay 200 µs
Yes
Transmit Read Busy Flag (0xF0) or Read Status Register Command (0x3C)
Busy?
Yes
No
Transmit Read Status Register Command (0x3C)
Fail?
Yes
Clean Up
No
Verify Usercode?
No
Yes
Transmit Read USER-CODE Command (0xC0)
Usercode OK?
No
Clean Up
Yes
Verify Configuration Flash?
No
Yes
Transmit Reset Config-uration Address (0x46) or Set Flash Address Command (0xB4)
(Contined on Next Page)
3


3
Transmit Read Command with Number of Pages (0x73)
(Contined from Previous Page)
Read Page Data
Data OK?
No
Clean Up
Yes
Transmit Reset UFM Address (0x46) or Set Address Command (0xB4)
No
All Pages Read?
Yes
Yes
Verify UFM?
Write Feature Row?
No
Program Done
Yes
Transmit Write Feature Row Command (0xE4)
Check Busy?
No
Delay 200 µs
Yes
Transmit Read Busy Flag (0xF0) or Read Status Register Command (0x3C)
Yes
Busy?
No
Transmit Read Feature Row Command (0xE7)
Feature Row OK?
No
Clean Up
Yes
Transmit Write FEA Bits Command (0xFB)
4


4
Check Busy?
No
Delay 200 µs
Yes
Transmit Read Busy Flag (0xF0) or Read Status Register Command (0x3C)
Busy?
Yes
No
Transmit Read FEA Bits Command (0xFB)
FEA Bits OK?
No
Clean Up
Program Done
Transmit Program DONE Command (0x5E)
Check Busy?
No
Delay 200 µs
Yes
Transmit Read Busy Flag (0xF0) or Read Status Register Command (0x3C)
Busy?
Yes
No
Transmit Read Status Register Command (0x3C)
DONE Set?
No
Clean Up
Yes
5


5
Write Security Bit?
No
Yes
Transmit Program SECURITY (0xCE) or SECURITY PLUS Command (0xCF)
Check Busy?
No
Delay 200 µs
Yes
Transmit Read Busy Flag (0xF0) or Read Status Register Command (0x3C)
Busy?
Yes
No
Program OTP Fuses?
Transmit Program OTP Command (0xF9)
Transmit Read Busy Flag (0xF0) or Read Status Register Command (0x3C)
Busy?
Yes
No
Verify OTP Fuses?
No
Yes
Transmit Read OTP Command (0xFA)
OTP Fuses OK?
No
Clean Up
Yes
6


Exit
6
Transmit Disable Configuration Interface Command (0x26)
No
Offline Mode?
Yes
Transmit Refresh Command (0x79)
Wait $t_{REFRESH}$
Check Refresh Successful?
Yes
Transmit Read Status Register Command (0x36)
No
Yes
Refresh Successful?
No
Retry Count Exceeded?
No
Yes
Done
Clean Up


Clean Up
Transmit Erase Flash CF/UFM/FR Command (0x0E)
Transmit Read Busy Flag (0xF0) or Read Status Register Command (0x3C)
Note: Do Not use fixed delays for the Flash erase sequence.
Yes
Busy?
No
Transmit Refresh Command (0x79)
Done
Note: The MachXO2 Configuration Flash/UFM/Feature Row memories are erased.

MachXO2 Programming Commands

Table 14-22. MachXO2 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 00[1]	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 00[1]	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 Status Register [LSC_READ_STATUS]	0x3C	00 00 00	N/A	YY YY YY YY	Bit 1 0 12 Busy Ready 13 Fail OK
Erase [ISC_ERASE]	0x0E	0Y 00 00	N/A	N/A	Y = Memory space to erase Y is a bitwise OR Bit 1=Enable 17 Erase Feature Row 18 Erase Configuration Flash 19 Erase UFM
Erase UFM [LSC_ERASE_TAG]	0xCB	00 00 00	N/A	N/A	Erase the UFM sector only.
Reset Configuration Flash Address [LSC_INIT_ADDRESS]	0x46	00 00 00	N/A	N/A	Set Page Address pointer to the beginning of the Configuration Flash sector
Set Address [LSC_WRITE_ADDRESS]	0xB4	00 00 00	M0 00 0P PP	N/A	Set the Page Address pointer to the Flash page specified by the least significant 14 bits of the PP PP field. The 'M' field defines the Flash memory space to access. Field 0x0 0x4 M Configuration Flash UFM
Program Page [LSC_PROG_INCR_NV]	0x70	00 00 01	YY * 16	N/A	Program one Flash page. Can be used to program the Configuration Flash, or UFM.
Reset UFM Address [LSC_INIT_ADDR_UFM]	0x47	00 00 00	N/A	N/A	Set the Page Address Pointer to the beginning of the UFM sector
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 FEA bits
Read FEABITS [LSC_READ_FEABITS]	0xFB	00 00 00	N/A	YY * 2	Retrieves the FEA bits
Read Flash [LSC_READ_INCR_NV]	0x73	M0 PP PP	N/A	See "Reading Flash Pages" on page 14-48	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 memory. 0x0 I^2C 0x1 JTAG/SSPI/WB

Table 14-22. MachXO2 sysCONFIG Programming Commands (Continued)

Command Name [SVF Synonym]	Command	Operands	Write Data	Read Data	Notes
Read UFM Flash [LSC_READ_UFM]	0xCA	M0 PP PP	N/A	See "Reading Flash Pages" on page 14-48	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/WB
Program DONE [ISC_PROGRAM_DONE]	0x5E	00 00 00	N/A	N/A	Program the DONE status bit enabling SDM
Program OTP Fuses [LSC_PROG_OTP]	0xF9	00 00 00	UCFSUCFS	N/A	Makes the selected memory space One Time Programmable. Matching bits must be set in unison to activate the OTP feature. Bit 1 0 0, 4 SRAM OTP SRAM Writable 1, 5 Feature Row OTP Feature Row Writable 2, 6 CF OTP CF Writable 3, 7 UFM OTP UFM Writable
Read OTP Fuses [LSC_READ_OTP]	0xFA	00 00 00	N/A	UCFSUCFS	Read the state of the One Time Programmable fuses. Bit 1 0 0, 4 SRAM OTP SRAM Writable 1, 5 Feature Row OTP Feature Row Writable 2, 6 CF OTP CF Writable 3, 7 UFM OTP UFM Writable
Disable Configuration Interface [ISC_DISABLE]	0x26	00 00	N/A	N/A	Exit Offline or Transparent programming mode. The command causes the MachXO2 to reconfigure when leaving Offline mode. The Configuration SRAM must be cleared prior to transmitting the Disable Configuration Interface command.
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 MachXO2 to reconfigure. Transmitting a REFRESH command reconfigures the MachXO2 in the same fashion as asserting PROGRAMN.
Program SECURITY [ISC_PROGRAM_SECURITY]	0xCE	00 00 00	N/A	N/A	Program the Security bit (Secures CFG Flash sector).[2]
Program SECURITY PLUS [ISC_PROGRAM_SECPLUS]	0xCF	00 00 00	N/A	N/A	Program the Security Plus bit (Secures CFG and UFM Sectors).[2]
Read TraceID code [UIDCODE_PUB]	0x19	00 00 00	N/A	YY*8	Read 64-bit TraceID.

1. Transmit the command opcode and first two operand bytes when using the I^2C port. The final operand byte must not be transmitted.
2. SECURITY and SECURITY PLUS commands are mutually exclusive.

Reading Flash Pages

Reading the Configuration Flash and the User Flash Memory pages requires a specific procedure. The Configuration Flash and UFM pages are accessible from any of the MachXO2's configuration ports. The JTAG, Slave SPI, and the WISHBONE configuration ports all behave identically when performing read operations. The I^2C port requires a modified access protocol. A high-level representation of the data flow, by port, is shown in Figure 14-19.

All ports start the read process in the same way, by sending a Read Flash/Read UFM Flash command. The MachXO2 begins the read process once the command byte has been accepted by the configuration logic. The Page Address Pointer determines the first page returned from the MachXO2. For the first returned page to be valid (e.g. for single-page read operations), a Retrieval delay of 240ns 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 See Figure 14-18. Note that for slower interface clock rates, 240ns may be consumed entirely by the normal transmission of the first Operand and no additional delay may be necessary.

Figure 14-18. Retrieval Delay Timing Requirement for Single-Page Reads

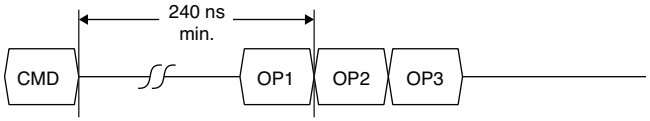
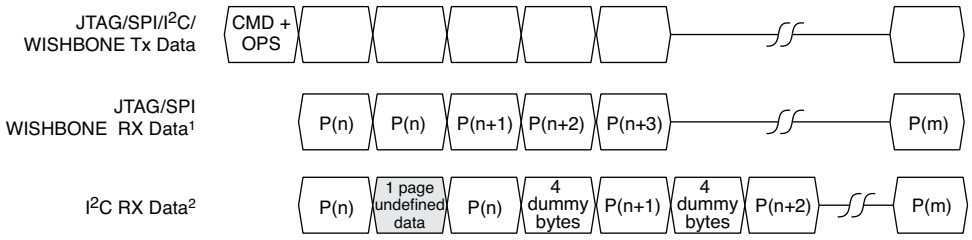


Figure 14-19. Flash Page Command and Data Sequence



Notes:

1. JTAG/SSPI must transmit data in order to read data back. The data sent by the JTAG/SSPI master is not specified (i.e. don't care).

2. The I^2C must use RESTART between sending the CMD and reading the data. (Issuing a STOP terminates a CMD and resets the I^2C state machine.)

CMD + OPS = Read Flash or Read UFM Flash command byte + 3 operand bytes.

Figure 14-19 shows a multiple page read sequence. The Read Page, or Read UFM Page command is transmitted to the MachXO2. As can be seen in Figure 14-19, 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 I^2C interface differs only in the Read Flash/Read UFM Flash 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 Flash/Read UFM Flash command to be assigned a value of 3. If the Page Address Pointer is 0000, the MachXO2 will return three pages, Page 0, Page 0, and Page 1.

The I^2C interface has additional overhead when reading Flash memory pages. Reviewing Figure 14-19 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 will return Page 0, 16 undefined bytes, Page 0, 4 dummy bytes, and Page 1. Reading the final four dummy bytes is optional.

References

- MachXO2 Family Data Sheet
- TN1205, Using User Flash Memory and Hardened Control Functions in MachXO2 Devices
- TN1207, Using TraceID in MachXO2 Devices

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
November 2010	01.0	Initial release.
May 2011	01.1	Added Appendix A and Appendix B.
May 2011	01.2	Corrected I^2C Function for the yyyxxxxx11 slave address in the Slave Addresses for I^2C Ports table.
August 2011	01.3	Added User SPI during transparent programming caution.
		Added external SPI address for dual boot option.
February 2012	01.4	Document status changed from Advance to Final.
		Updated document with new corporate logo.
June 2012	02.0	Major update, including: • Updated Programming algorithm • Added Feature Row discussion • Improved coverage of configuration port management
July 2012	02.1	Clarified SECURITY/SECURITY PLUS in Figure 14-16, MachXO2 Flash Memory Programming Flow, and Table 14-21, MachXO2 sysCONFIG Programming Commands.
		Added Figure 14-17, Retrieval Delay Timing Requirement for Single-Page Reads.
		Clarified Retrieval delay in Figure 14-18, Flash Page Command and Data Sequence.
		Added missing 'Program DONE' step in Flash Memory Programming Flow.
September 2012	02.2	Updated TransFR operation.
		Enhanced programming flow chart.
		Updated PROGRAMN configuration pin information.
		Added details about MUX_CONFIGURATION_PORTS.
October 2012	02.3	Added restriction: Primary port can be used as Configuration/UFM port or as User port, but not both.