LatticeXP2 Slave SPI Port User's Guide

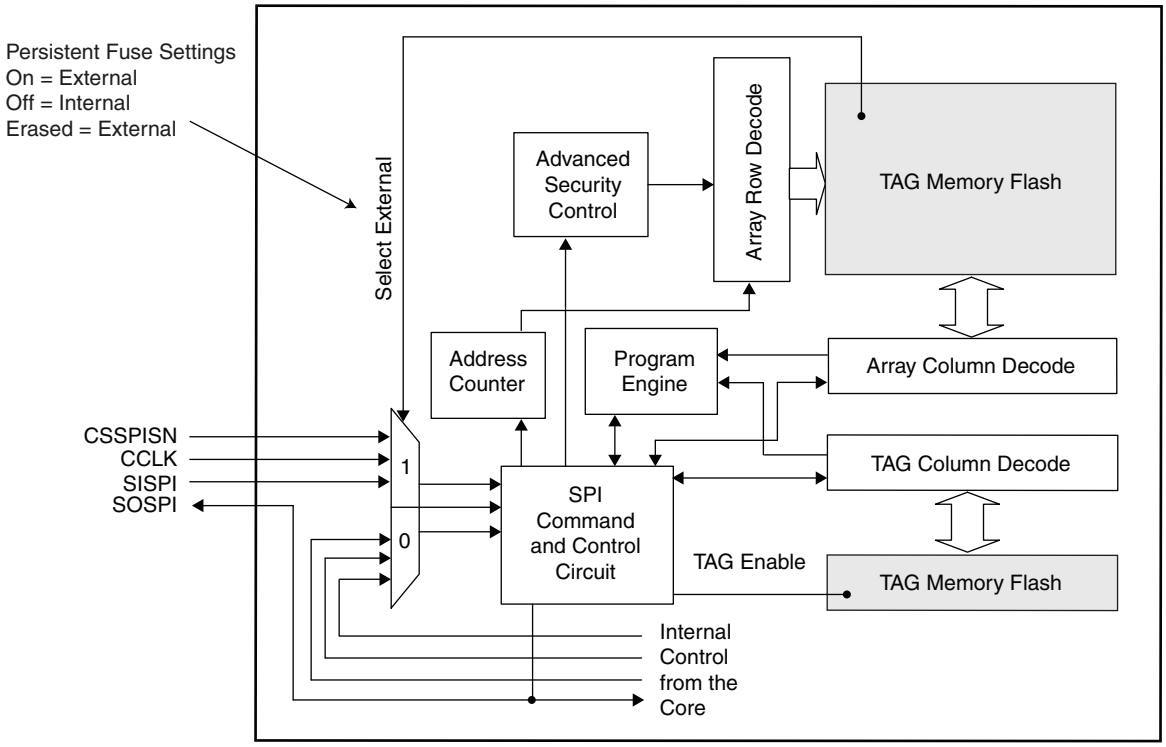
June 2012 Technical Note TN1213

Introduction

The Serial Peripheral Interface (SPI) is the industry standard interface that can be found on most CPU and serial Flash memory devices. The drivers for reading and writing from memory devices are readily available on modern digital systems.

This application note describes the programming detail using the built-in SPI port in the LatticeXP2 devices to program the configuration Flash module. A basic block diagram is shown in Figure 1.

Figure 1. LatticeXP2 Slave SPI Port, TAG Memory, and Configuration Flash Block Diagram



Note: The Slave SPI port can only access the embedded Flash cells in background programming mode.

Definitions

Erase

Clear all the Flash cells state to a logical one (1) (a.k.a. open fuse).

Program

Write into the selected Flash cells state a logical zero (0) (a.k.a. close fuse).

Configure

Write the pattern into the SRAM fuses.

Direct Mode

The device is in programming mode with all the I/O pins kept at tri-state.

Background Mode

The device is in programming mode with all the I/O pins remain operational.

Built-In Feature

A feature comes with the silicon without pre-instantiation.

Dedicated Pin

The pin has only one function.

Dual Purpose Pin

The pin has more than one function.

Secure

Protect the Flash and SRAM fuses from reading.

Advanced Security

The advanced features that provide additional security to the device, for example, Encryption, Flash Protect, or One Time Programmable (OTP) features.

Master SPI

Industry standard Serial Peripheral Interface (SPI) used to configure the SRAM fuses from the bitstream stored in an external SPI Flash device.

Slave SPI (SSPI)

Access the SPI port through the Dual Purpose I/O pins.

Internal SPI

Access the SPI port through the FPGA fabric.

Refresh

Initiates the device to reconfigure the SRAM fuses from the embedded configuration Flash fuses or external SPI Flash device, depending on the configuration mode selected.

Self Download Mode (SDM)

Upon power up or a Refresh, the device is configured by a massive parallel transfer of the data programmed in the embedded configuration Flash fuses to the SRAM fuses.

JEDEC File (.JED File)

The programming data file as defined by JEDEC 42.1C standard. The programming file is expressed in the ASCII 1 and 0 format. The file is printable. 3rd party programmers use it to support large volume production programming.

Binary Data File (.BIN File)

The programming data file converted from the JEDEC file. The programming file is expressed in binary hex format. The file is not printable.

The conversion utility, in C source code form or compiled program form, is available from Lattice.

Bitstream Data File (.BIT File)

The configuration data file in the format that can be loaded directly into the FPGA devices to configure the SRAM cells. The file is expressed in binary hex format. The file is not printable.

Dual Boot

The device can boot from two sources: the embedded Flash in the device or an external SPI Flash. Depending on configuration, the device may use one as primary source and the other as golden source. (Note: The only way to select this feature is by holding the CFG0 pin statically to low before, during and after powering up the LatticeXP2 device.)

Row

A row is one address location selected by the address pointer inside the LatticeXP2 device.

Column

A column is one data bit location in the row (or address) currently selected by the address pointier.

Hardware Interfaces

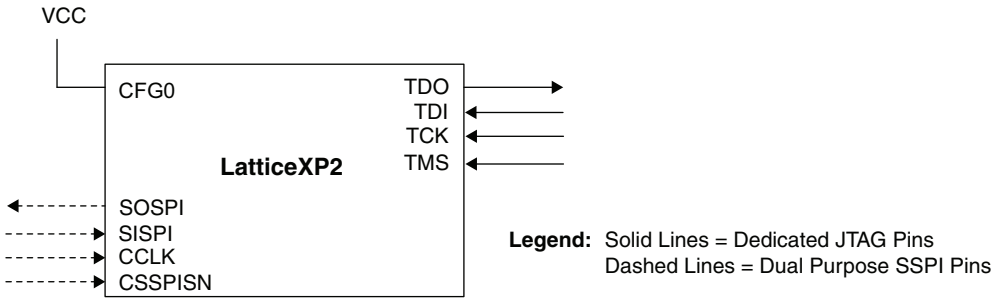
The following is a list of built-in interfaces that are available on the LatticeXP2 family.

- Slave SPI
  - Dedicated and always available except when:
    - The device is set into JTAG programming mode by the JTAG port.
    - The Master SPI interface is active.
  - Supports the industry standard SPI protocol.
    - OPCODE = 8 bits.
    - Command = 32 bits (8-bit OPCODE + 24-bit operand).
    - Supports intelligent Flash programming.
    - Programming action triggered by a low to high edge on the CSN pin.
  - External/Internal Slave SPI interface.
    - The external SPI interface is automatically selected when the device is erased.
    - Persistent Fuse Settings:
      - On: Selects the External Interface (SISPI, SOSPI, CSSPISN, and CCLK).
      - Off: Selects the Internal Interface (four CIB signals of the SSPIA core).
  - Supports JEDEC files.
    - Standard
    - Encrypted
- Master SPI (Power up with CFG0 = 0)
  - Supports industry standard SPI Flash devices.
  - Supports standard FPGA bitstream file (Encrypted bit-streams are not supported).
- ispJTAG™ (1149.1 Interface)
  - IEEE 1149.1 and IEEE 1532 compliant.
  - Dedicated and always available.

General Descriptions

If the CFG0 pin is tied to VCC, the LatticeXP2 device is in Self Download Mode (SDM). In this mode, the LatticeXP2 family of devices is designed with two standard programming ports, JTAG and Slave SPI. A basic block diagram is shown in Figure 2.

Figure 2. Self Download Mode with Persistent Enabled



ispJTAG™ (1149.1 Interface)

The JTAG port is the primary programming port. The JTAG pins are hardwired as required by the IEEE 1532 standard. The four JTAG pins can never be recovered as user I/O pins. As the primary programming port, all programming activities can be carried out on this port. Particularly, the Advanced Security features, such as enabling the Flash Protect, enabling the encryption, and programming the password and keys into the device can only be done on this port.

Slave SPI

The Slave SPI port is the secondary programming port. The four SPI pins are dual-purpose pins, not dedicated. All four SPI pins can be recovered as user I/O pins when the Persistent option is disabled. The Persistent option will be disabled by the software unless the user sets the SLAVE_SPI_PORT to Enable using the Design Planner in ispLEVER®. This option is shown in the Global tab of the Design Planner Spreadsheet view. As a secondary port, only a subset of the programming activities is available from this port.

- Program and verify an encrypted or standard JEDEC file into embedded configuration Flash in background mode only. Direct programming mode is not supported.
- Read the SRAM fuse cells in background mode only. The SRAM fuse configuration is not supported.
- Trigger a Refresh operation to download the embedded configuration Flash to the SRAM cells. The TransFR feature is not available on SPI port. The I/O pins will all be tri-stated during the Refresh.

The LatticeXP2 devices are shipped from Lattice with an erased embedded configuration Flash. The persistent fuse is On in the erased state, which is equivalent to setting the SLAVE_SPI_PORT to Enable in ispLEVER. Therefore, the Slave SPI interface is enabled when the LatticeXP2 devices are shipped from Lattice. It is up to the user to select the desired setting in ispLEVER before generating the JEDEC file.

This application note assumes that the SLAVE_SPI_PORT is set to Enable in ispLEVER when the JEDEC file is generated.

This application note also assumes that the Flash Protect password and the encryption key are already programmed into the device. The Flash Protect password and the encryption key can only be programmed into the device using the JTAG port.

Pin Descriptions

Serial Data Input (SISPI)

The SPI Serial Data Input pin is for commands and data to be serially written to (shifted into) the device. Data is latched on the rising edge of serial clock (CCLK) input pin.

Serial Data Output (SOSPI)

The SPI Serial Data Output pin is for status and data to be serially read out (shifted out of) the device. Data is shifted out on the falling edge of serial clock (CCLK) input pin.

Serial Clock (CCLK)

The SPI Serial Clock Input pin provides the timing for serial input and output operations.

Chip Select (CSSPISN)

The SPI Chip Select pin enables and disables (reset) SPI interface operations. When the Chip Select transits from low to high the SPI interface is reset into a ready for command state and the Serial Data Output (SOSPI) pin is at high impedance. When it is brought from high to low, the SPI interface is selected, commands can be written into and data read from the device. After power up, this pin must transit from high to low before a new command can be accepted.

SPI Operations

SPI Modes

The SPI interface is accessible through the SPI compatible bus consisting of four signals: Serial Clock (CCLK), Chip Select (CSSPISN), Serial Data Input (SISPI), and Serial Data Output (SOSPI). Both SPI bus operation Modes 0 (0,0) and 3 (1,1) are supported. The primary difference between Mode 0 and Mode 3 concerns the normal state of the CCLK pin when the SPI master is in standby and data is not being transferred to the device's SPI interface. For Mode 0, CCLK is normally low. For Mode 3, the CCLK signal is normally high. In either case, data input on the SISPI pin is sampled only during the rising edge. Data output on the SOSPI pin is clocked out only on the falling edge of CCLK.

Device Status Register

The LatticeXP2 has an 8-bit device status register, which indicates the status of the device, as defined in Table 1.

Table 1. 8-Bit Device Status Register

Status Bit	Description	Bit State	
		0	1
B0	The status of the erase operation just completed (see Note 1).	Pass	Fail
B1	The state of the Done fuse.	Erase	Programmed
B2	The test result of the Flash Protect Key entered.	Match	No Match
B3	Reserved (see Note 2).	Default	
B4	The OTP (One Time Programmable) fuse setting.	Off	On
B5	The Flash Protect setting.	Disabled	Enabled
B6	The Encryption setting.	Disabled	Enabled
B7	The standard security fuse setting.	Erase	Programmed

Notes:

1. This covers the ERASE and ERASE_TAG command. The one bit status bit accessed by the PROGRAM_STATUS command only indicates if the erase action has completed. However, it does not indicate if it passed or failed. Thus, this bit is defined to indicate whether the erase operation passed or failed. A Pass means all the cells were erased with the proper margin per the data retention requirement. A Fail means some cells fail to be erased to the proper margin.
2. If the SED (Soft Error Detection) feature is selected, this bit shows the comparison result of the SED CRC. If the SED feature is selected, the SED must not be running when using the SPI port to program the device. The SED engine and programming engine use the same data shift register. Only one task can use the data shift register at one time. Otherwise, the result will be unpredictable.

Program Status Register

The LatticeXP2 has a 1-bit program status register, which indicates when the erase or program action is completed, as defined in Table 2.

Table 2. 1-Bit Program Status Register

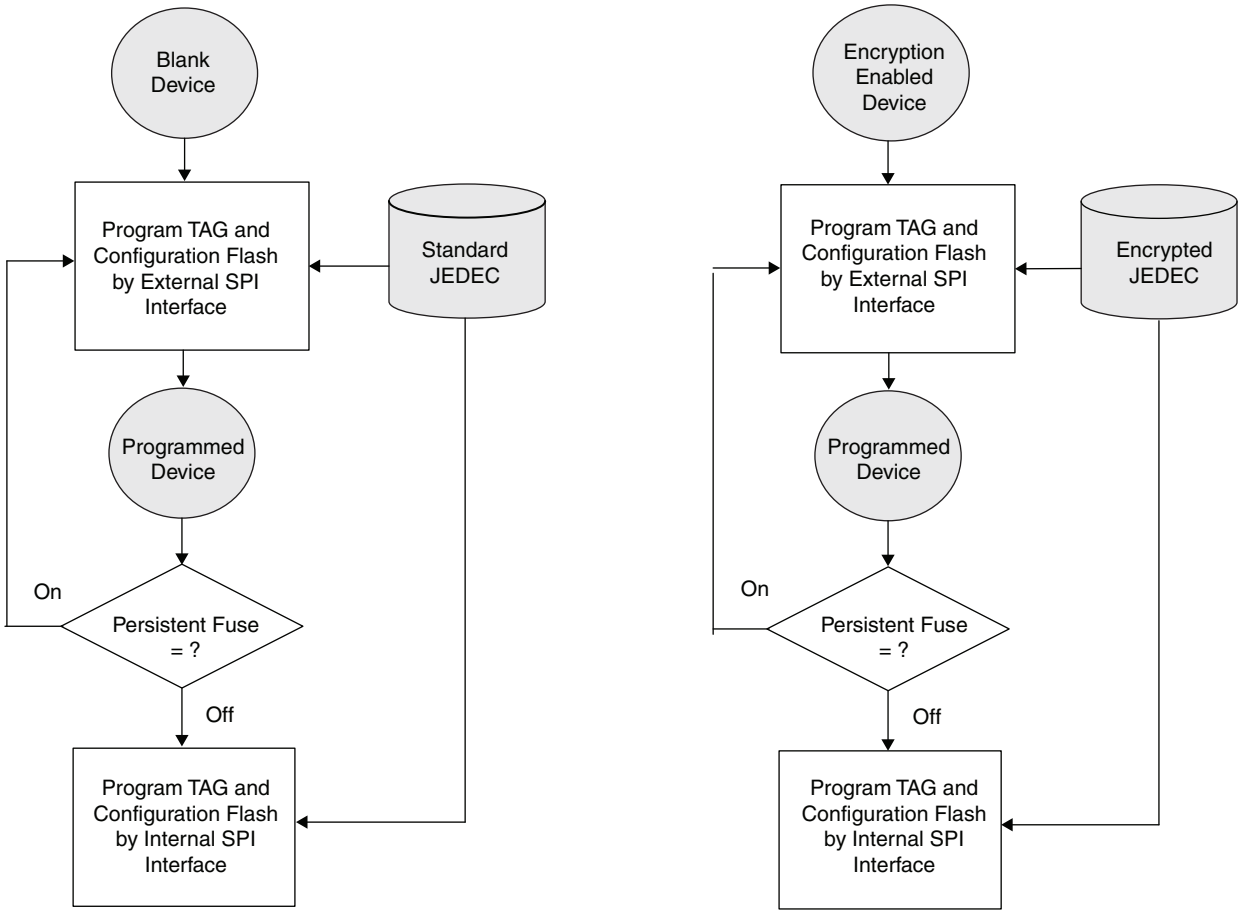
Program Status Register Value	Status
0	Not Complete
1	Completed

Slave SPI Programming Flow Diagrams

The LatticeXP2 can be programmed with a standard or encrypted JEDEC file, as shown in Figure 3.

Note: The JEDEC files must first be converted from an ASCII binary format to a binary HEX format using the JED2HEX utility or ispVM® System Software.

Figure 3. Slave SPI Programming Flow for Standard and Encrypted JEDEC Files



The LatticeXP2 Slave SPI programming algorithm for the embedded configuration Flash resembles the popular standard SPI Flash memory devices available in the market place. The erase, program, and verify flow diagrams are shown in Figure 4 through Figure 10.

Note: Please refer to Table 6 for the erase and program loop delays and maximun number of loops. Please refer to Table 8 for the number of rows and columns for each member of the LatticeXP2 family.

Figure 4. Check ID and Erase Flow Diagram

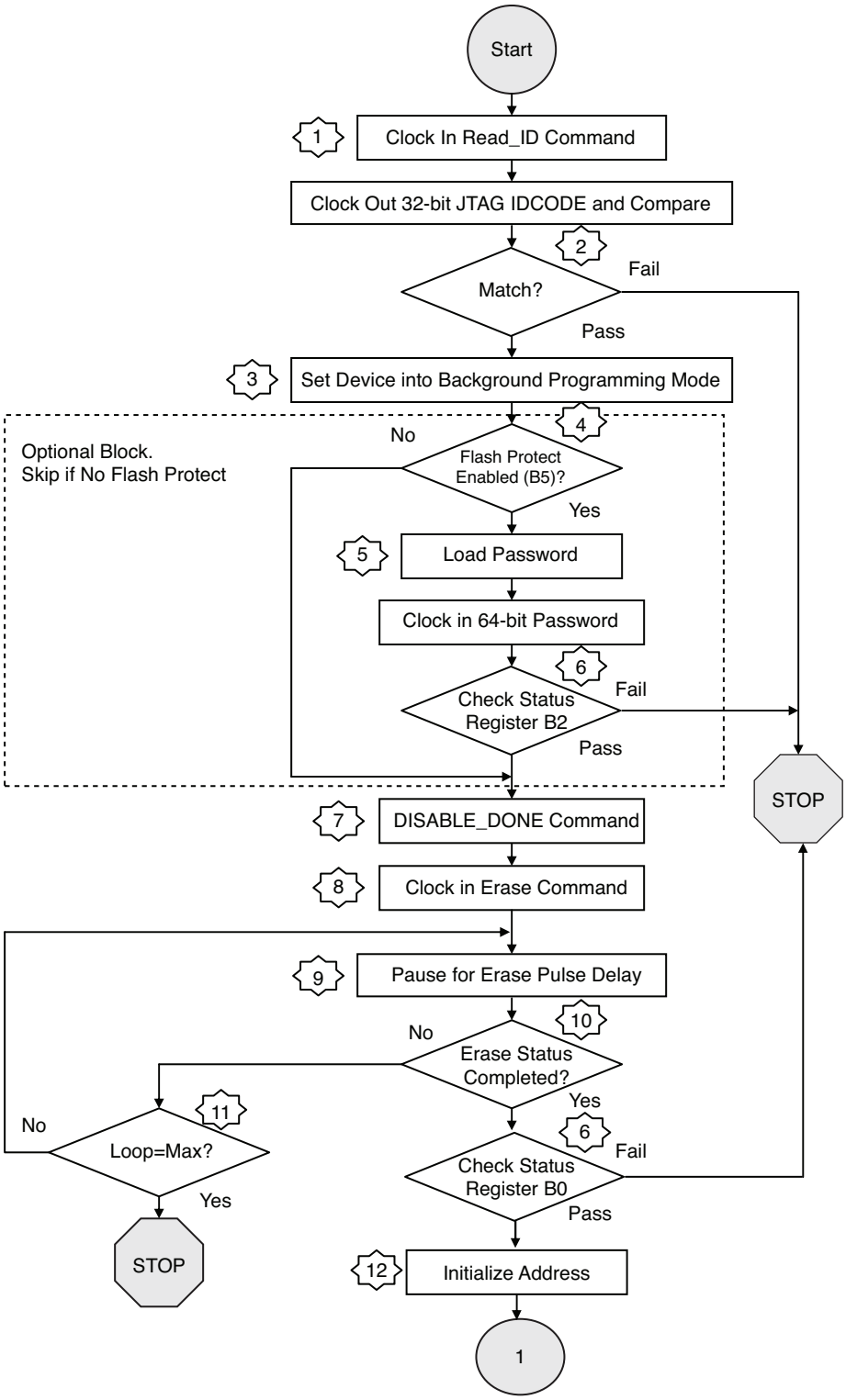


Figure 5. Standard and Encrypted Programming Flow Diagram

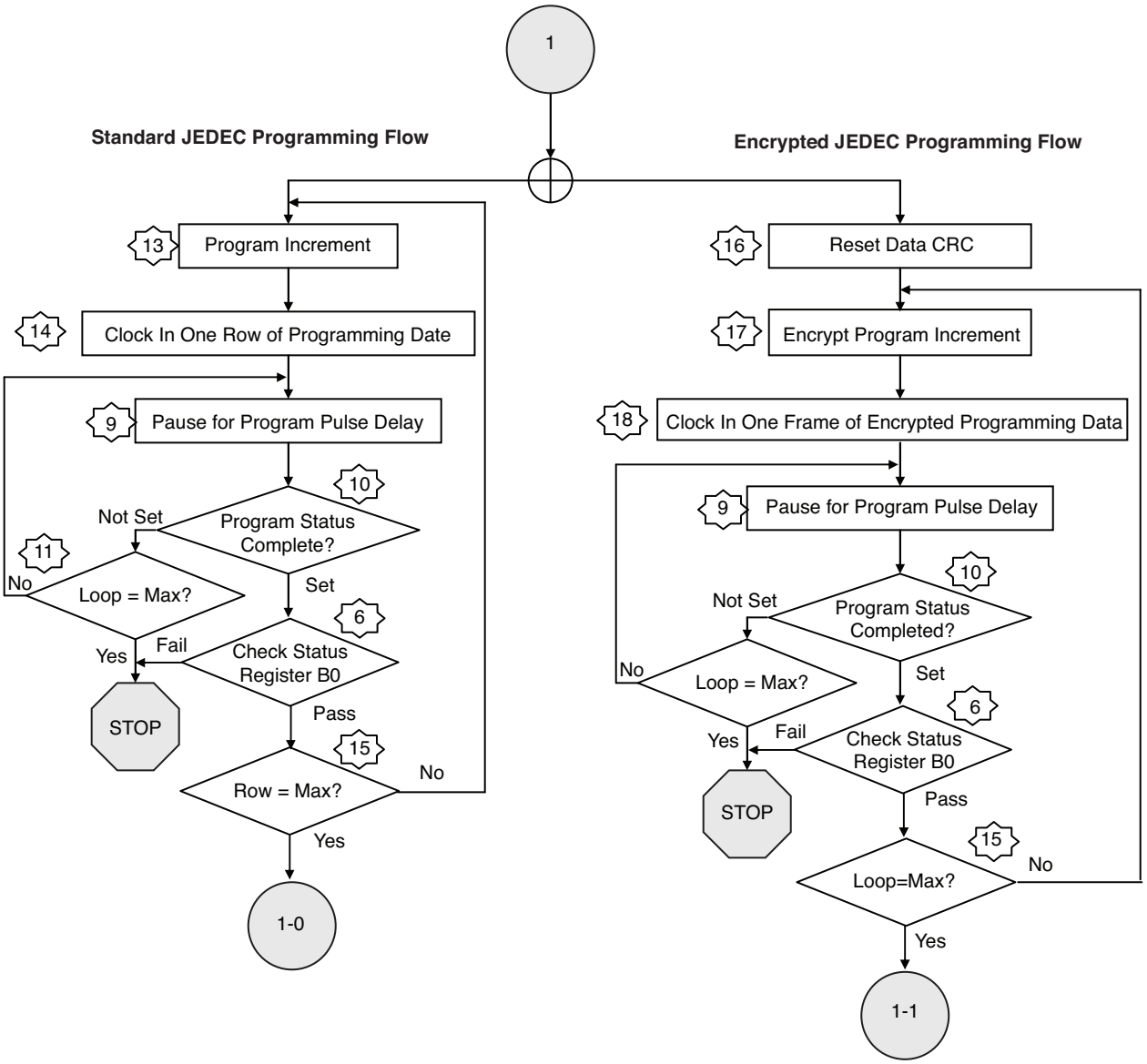


Figure 6. Program SED CRC and Verify Flow Diagram

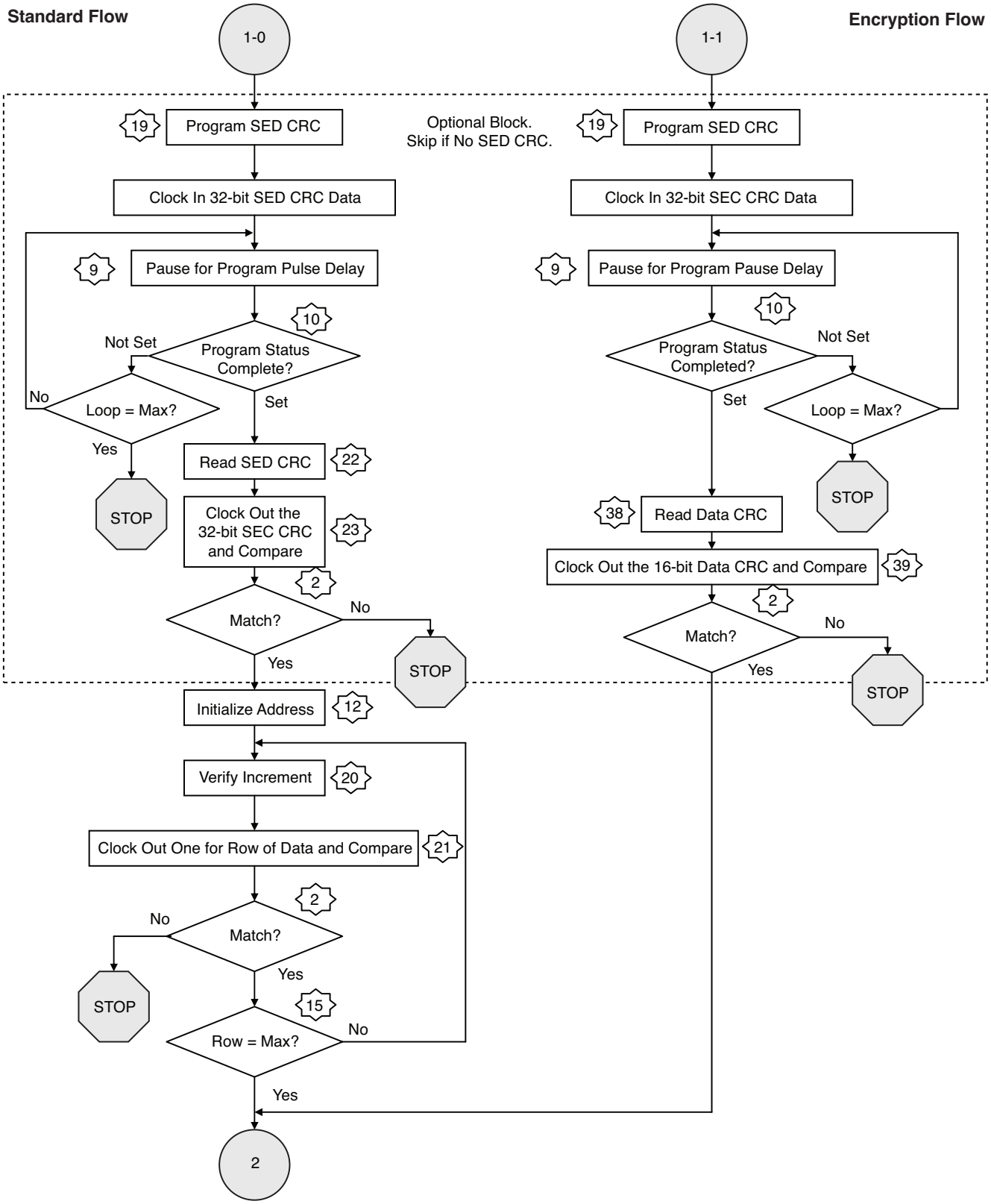


Figure 7. Program and Verify USERCODE Flow Diagram

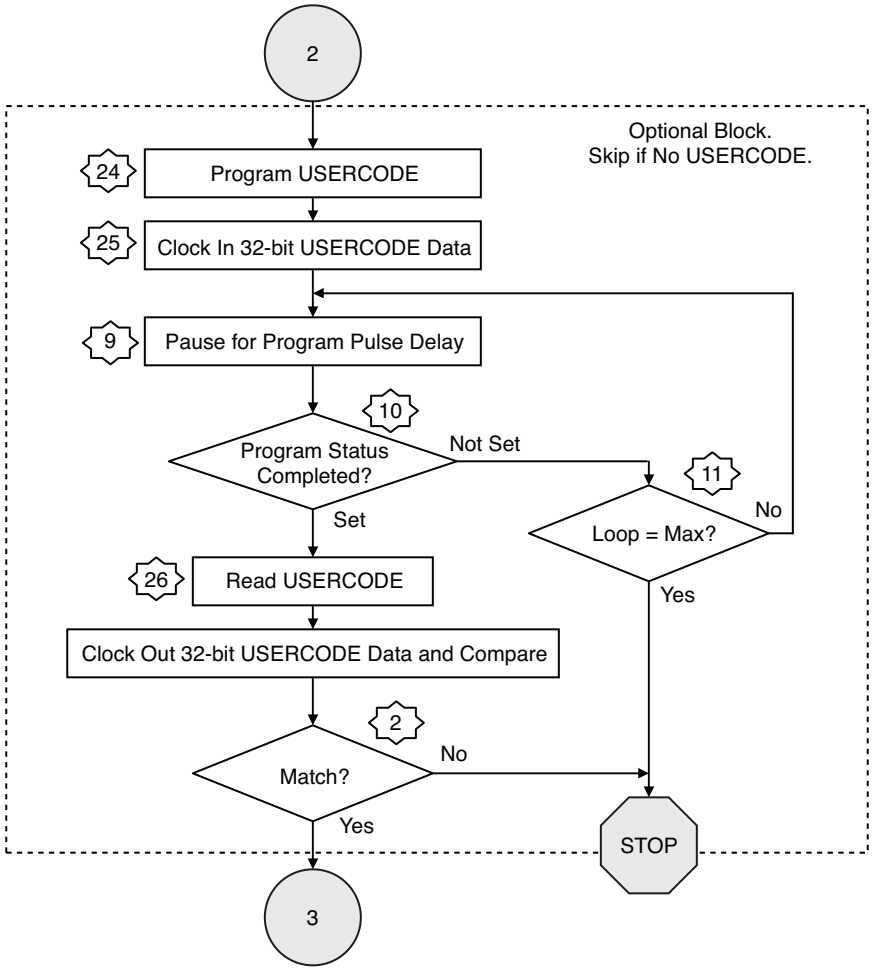


Figure 8. Program Security and Done Fuse Flow Diagram

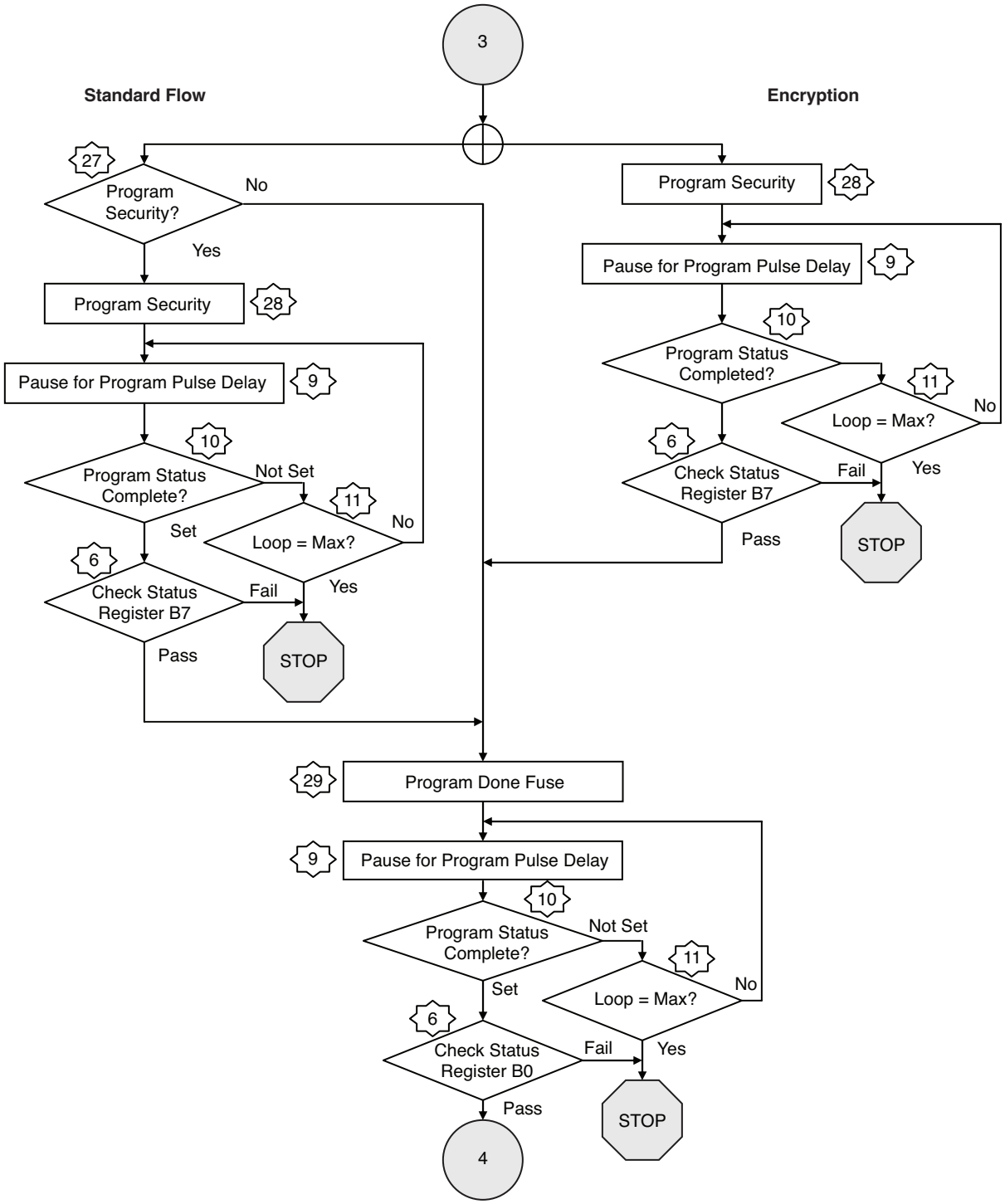


Figure 9. Program TAG Flow Diagram

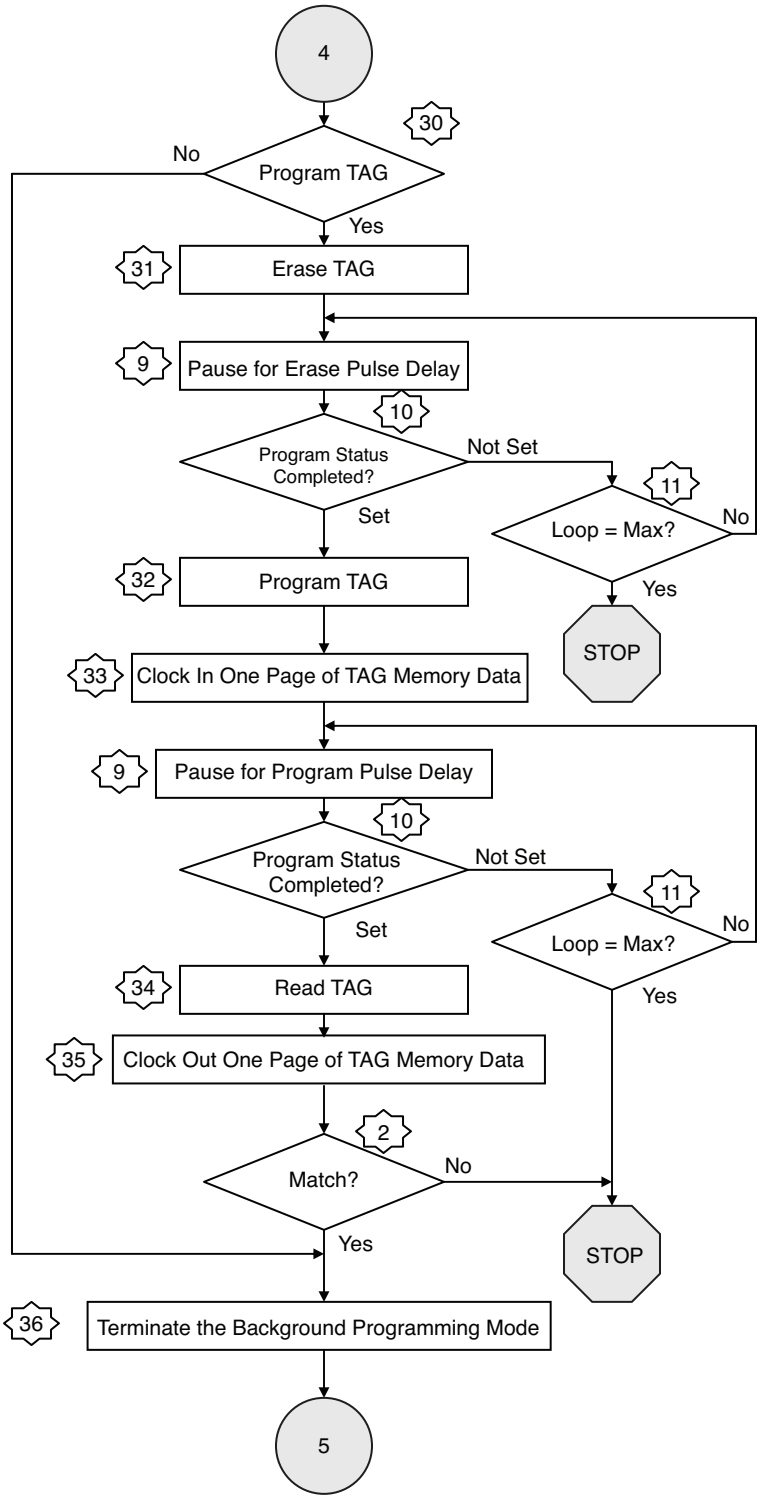
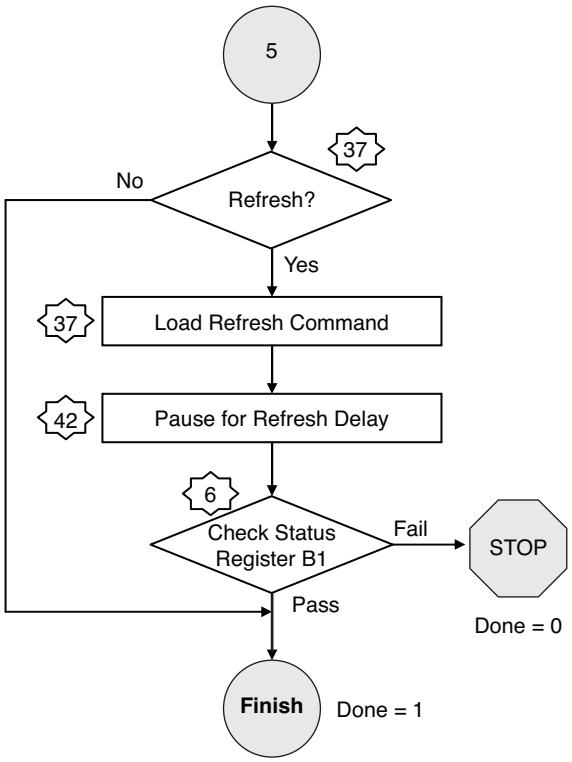


Figure 10. Refresh Flow Diagram



Commands Table

Table 3. Slave SPI Command Table

Command Name	Byte 0	Bytes 1-3	Description
READ_ID	0x98	Dummy	Read 32-bit JTAG IDCODE.
X_PROGRAM_EN	0xAC	Dummy	Enable Background Flash Mode to activate the Flash erase, programming, and read back command.
X_SRAM_READ_EN	0xAE	Dummy	Enable Background SRAM Mode to activate the read (VERIFY_INC) command to target at the SRAM fuses instead of the Flash fuses.
READ_TAG	0x4E	Dummy	Read TAG Memory. Use the command sequence as shown to read TAG memory. X_PROGRAM_EN, READ_TAG, PROGRAM_DIS. The sequence is necessary to minimize the stress to the Flash cells or to maximize the read cycle endurance.
PROGRAM_TAG	0x8E	Dummy	Program TAG Memory.
ERASE_TAG	0x0E	Dummy	Erase TAG Memory.
READ_PROGRAM_STATUS	0x4A	Dummy	Read out the one bit programming status register. 1 = erase or programming action complete. 0 = action not yet started or already in progress.
PROGRAM_DIS	0x78	Dummy	Terminate the effect of the previously loaded enable command.
DISABLE_DONE	0x24	Dummy	Erase the bit indicating the internal Flash has a valid pattern.
ERASE	0xC0	Dummy	Erase Configuration Flash.
READ_USERCODE	0xE8	Dummy	Read 32-bit Flash or SRAM USERCODE.
PROGRAM_USERCODE	0x58	Dummy	Program 32-bit USERCODE.
INIT_ADDRESS	0x84	Dummy	Sets the Configuration Flash address to the first row.
PROGRAM_INC	0xE6	Dummy	Programs one row of configuration Flash, then increments the row address.
VERIFY_INC	0x56	Dummy	Reads one row of configuration Flash or SRAM fuses, then increment the row address. Note: This command will not work if the encryption feature already enabled.
PROGRAM_SED_CRC	0xA2	Dummy	Program the 32-bit SED CRC fuses.
READ_SED_CRC	0x22	Dummy	Read the 32-bit SED CRC fuses.
PROGRAM_SECURITY	0x90	Dummy	Program Security Fuse
PROGRAM_DONE	0xF4	Dummy	Program the DONE fuse.
REFRESH	0xC4	Dummy	Tri-state I/O and clear all SRAM fuses. If the done fuse is programmed, trigger a Configuration Flash to SRAM transfer. If the done fuse is not programmed, the Flash to SRAM transfer will not happen. Note: This command will not work if PROGRAM_DIS is not issued. Please refer to the flow.
ENCRYPT_PROGRAM_INC	0x02	Dummy	Program one row of encrypted data into the configuration Flash then increment the row address. Note: The command works only if the encryption feature already enabled.
PROTECT_SHIFT	0x82	Dummy	Program the 64-bit Flash Protect Password to unlock Configuration Flash for reprogramming.
READ_DEVICE_STATUS	0x4D	Dummy	Read the 8-bit device status register.
READ_16_CRC	0xA3	Dummy	Read the 16-bit CRC value from the CRC engine.
RESET_16_CRC	0x63	Dummy	Initialize the 16-bit CRC engine to 0.

Notes:

1. Commands are classified into classes for the convenience of waveform illustration.
2. Byte 1-3 are dummy clocks to provide extra timing for the device to execute the command. The data presented at the SI pin during these dummy clocks can be any value and do not have to be 0x00 as shown.
3. The READ_ID command reads out the 32-bit JTAG IDCODE of the device. The first bit shifted out on SOSPI pin is thus bit 0, which has the value = 1 as per IEEE 1149.1 standard of the JTAG IDCODE and the last bit is bit 31, which is the last bit of the revision field, of the IDCODE.

4. The PROGRAM_STATUS command read from the single bit status register. When read from the register, only the 1st bit is valid, the other bits are dummies and should be ignored.
5. Shift into the device an invalid OPCODE will cause the 1 bit bypass register connected to the SOSPI pin.
6. If there is under clocking, the CSSPISN pin is driven from low to high before enough clocks were applied, the device will behave as follows:
   a. If under clocking on the 8-bit IDOCDE, it has the same effect as an invalid OPCODE.
   b. If under clocking on the 24-bit dummy, the action of the command won't take place.
   c. If under clocking on shifting out data, the remaining un-read data will be lost.
   d. If under clocking on shifting in programming data, the partial programming data will be programmed into the device.
   e. If there is over clocking, the CSSPISN pin is driven from low to high after more than enough clocks were applied, the device will behave as follows:
7. If over clocking on the 8-bit IDOCDE, the extra clocks will be treated as dummy.
   a. If over clocking on the 24-bit dummy, the extra clocks will be treated as data if it is a Class C or D command, otherwise they are ignored.
   b. If over clocking on shifting out data, the extra data being shifted out will be unknown.
   c. If over clocking on shifting in programming data, the extra clock will overflow the targeted data register.

Use of Commands

Table 4. Slave SPI Command Usage Table

Command Name	OPCODE MSB LSB	Class	Data	Flow Operation Number	Delay Time
READ_ID	0x98	A	Out	1, 2	None (see Note 1)
X_PROGRAM_EN	0xAC	E		3	None (see Note 1)
X_SRAM_READ_EN	0xAE	E		See Note 3	None (see Note 1)
READ_TAG	0x4E	D	Out	34, 35	5µs min. (see Note 1 and 2)
PROGRAM_TAG	0x8E	C	In	32, 33, 34	See Table 6
ERASE_TAG	0x0E	B		31	SeeTable 7
READ_PROGRAM_STATUS	0x4A	D	Out	9, 10, 11	= Delay of the previous command
PROGRAM_DIS	0x78	E		36	1ms min.
DISABLE_DONE	0x24	B		7	1ms min.
ERASE	0xC0	B		8	See Table 7
READ_USERCODE	0xE8	D	Out	26	None (see Note 1)
PROGRAM_USERCODE	0x58	C	In	24, 25	1ms min., 25ms max.
INIT_ADDRESS	0x84	E		12	None (see Note 1)
PROGRAM_INC	0xE6	C	In	13, 14	See Table 6
VERIFY_INC	0x56	G	Out	20, 21	5µs min.
PROGRAM_SED_CRC	0xA2	C	In	19	1ms min., 25ms max.
READ_SED_CRC	0x22	D	Out	22, 23	None (see Note 1)
PROGRAM_SECURITY	0x90	B		27, 28	20ms min.
PROGRAM_DONE	0xF4	B		29	1ms min., 25ms max.
REFRESH	0xC4	E		37, 42	2ms min.
ENCRYPT_PROGRAM_INC	0x02	C	In	17, 18	1ms min., 25ms max.
PROTECT_SHIFT	0x82	F	In	5	None (see Note 1)
READ_DEVICE_STATUS	0x4D	D	Out	4, 6	None (see Note 1)
READ_16_CRC	0xA3	D	Out	38, 39	None (see Note 1)
RESET_16_CRC	0x63	E		16	None (see Note 1)

Notes:
1. The intrinsic delay provided by shifting in the twenty four (24) dummy bits, even if at 25 MHz, is sufficient and hence additional delay is not required.
2. The READ_TAG command requires the X_PROGRAM_EN command is shifted into the device first to qualify the READ_TAG command.
3. If this command, X_SRAM_READ_EN, is loaded instead of the X_PROGRAM_EN, the following commands will read out the SRAM fuses instead of from the Flash.
   VERIFY_INC

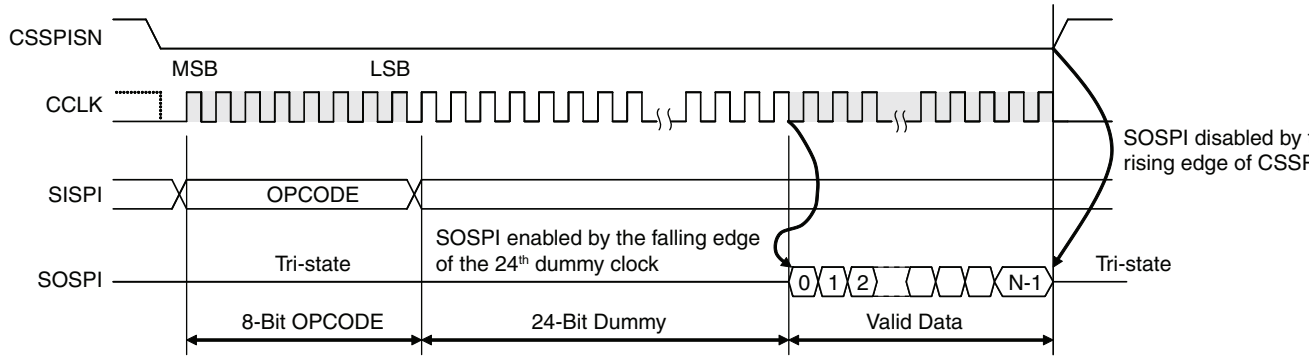
READ_SED_CRC
READ_USERCODE
This command has no effect on the READ_TAG command.

Command Waveforms

Class A Command Waveforms

The Class A commands is one that reads data out without an additional verification delay. The IDCODE command for reading out the 32-bit JTAG IDCODE is the only command in this class.

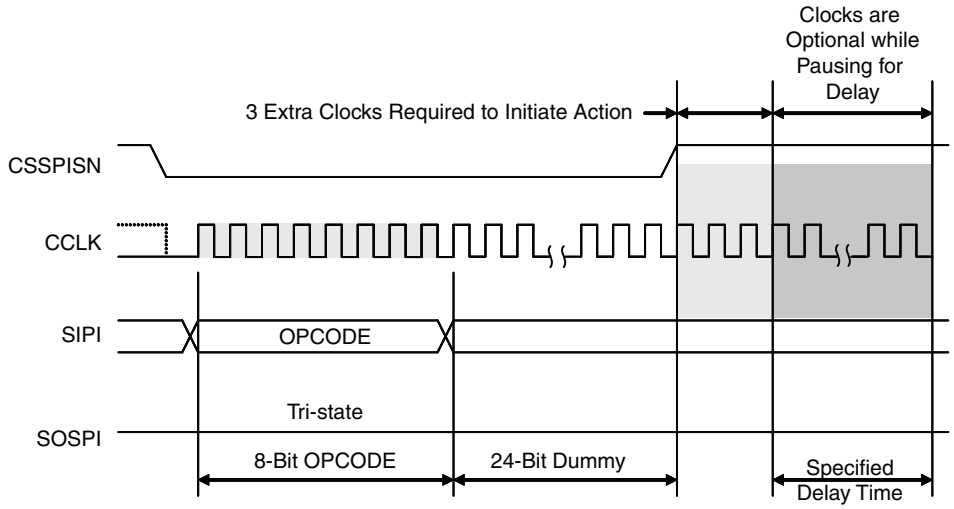
Figure 11. Class A Command Waveforms



Class B Commands Waveform

The Class B commands require a delay to execute the action associated with the command. These types of commands can only initiate it's own action. It cannot terminate the action of any commands, even it's own action. The device requires three (3) extra clocks after the rising edge of CSSPISN to initiate the action. After the specified delay time, the PROGRAM_STATUS command should be used to check if the action is completed. This will eliminate the possibility of having more than one action in progress at a giving time, which could cause the actions to clash.

Figure 12. Class B Command Waveforms



Class C Commands Waveforms

The Class C commands are used for shifting data into the devices. All of the programming data registers are constructed as FIFO's. The exact number of programming data bits must be provided to the selected programming data register. If over-shift or under-shift happens, the resulting data programmed into the device will be invalid.

Operations in this class are PROGRAM_TAG, PROGRAM_USERCODE, PROGRAM_INC, PROGRAM_SED_CRC, and ENCRYPT_PROGRAM_INC.

The actual configuration data may not be byte-bounded. Leading padding bits of zeros (0) can be used to pad the programming data into a byte bound hex file format. When shifting the programming data into the device, the overflowing characteristic of the FIFO behavior of the shift register is utilized to discard the leading padding bits. As a result, only the actual programming data will reside in the programming data register. The device requires three (3) extra clocks after the rising edge of CSSPISN to initiate the action. After the specified delay time, the PROGRAM_STATUS command should be used to check if the action is completed. This will eliminate the possibility of having more than one action in progress at a giving time, which could cause the actions to clash.

Figure 13. Type Programming Data Shift Register

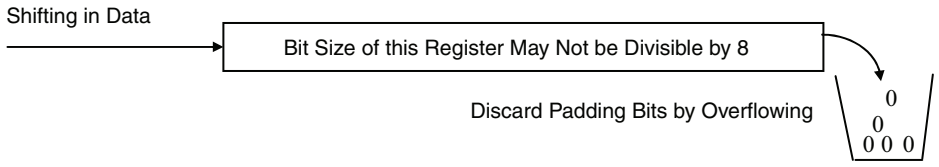
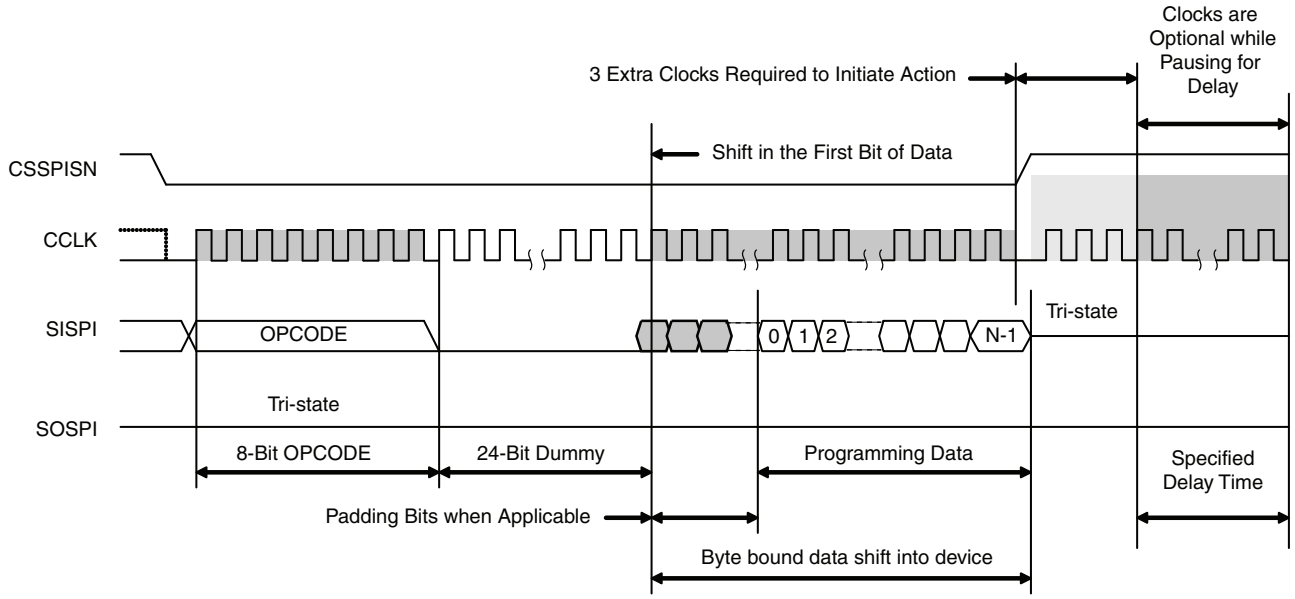


Figure 14. Class C Command Waveforms



Class D Commands Waveforms

The Class D commands read data out from the device. The data can be device status, programming status, USERCODE, or TAG Memory.

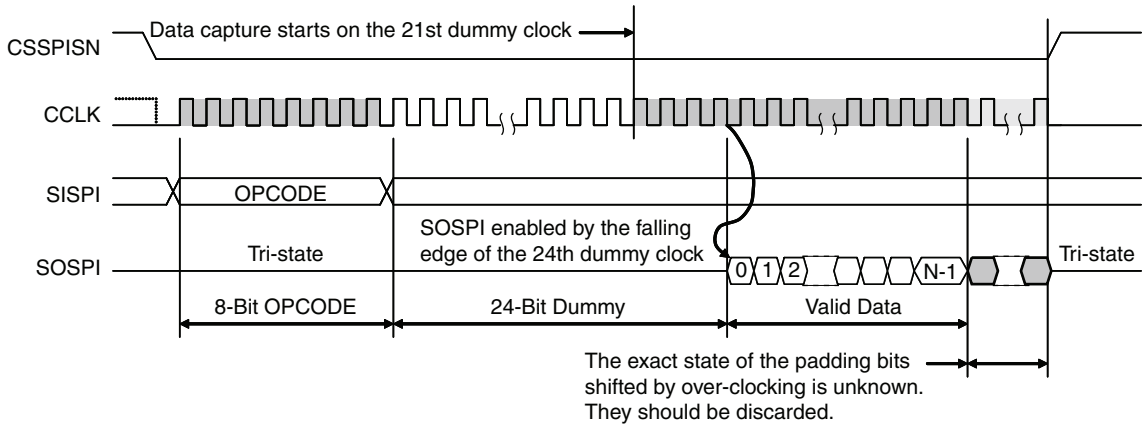
When reading the TAG Memory data, there is a 5µs verify delay time required to provide the device the time it needs to physically transferring the data storing in Flash cells into the FIFO data shift register. The twenty four (24) clocks used to shift in the dummy bits are considered part of the delay by the device. If the twenty four (24) clocks are applied faster than 5µs, then extra delay is required by holding the CCLK low before sampling SOSPI.

All commands in this class must be qualified by shifting in first the X_PROGRAM_EN command, which is equivalent to the WRITE_EN command in standard SPI Flash devices. If not, the transfer action will not happen, resulting in unknown data being shifted out.

If the programming data size is not divisible by eight (8), trailing padding clocks can be used to shift out byte bound data and then discard the residual bits clocked out by the trailing padding clocks.

Note: Please refer to Table 8 for the number of data bits *N* (columns) for each member of the LatticeXP2 family.

Figure 15. Class D Command Waveforms



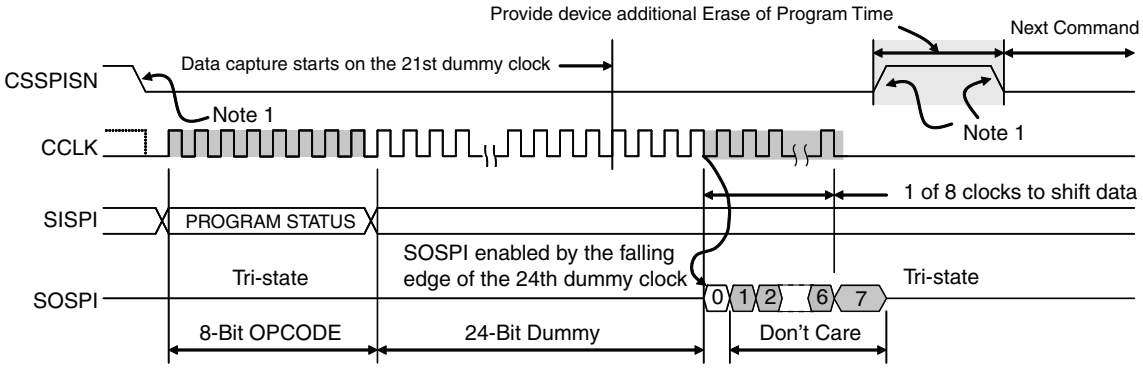
PROGRAM_STATUS (4Ah) Waveforms

The PROGRAM_STATUS command is a Class D command. Instead of reading out programming data, a single bit programming status is read. This command determines if the erase action or programming action complete successfully.

This command does not initiate, interrupt, or terminate the erase or programming action. It is used to interrogate the device to determine if the action initiated by the erase or programming command is done. If repetitive status bit polling is necessary, then repetitively issue the command and the twenty four (24) bits dummy.

The actual bit size of the status bit register is one (1) bit. After the thirty two (32) clocks of command OPCODE and dummy, if eight (8) read clocks are applied for the convenience of the driver program, then only care about the first bit shifted out and ignore the rest.

Figure 16. PROGRAM_STATUS Waveforms



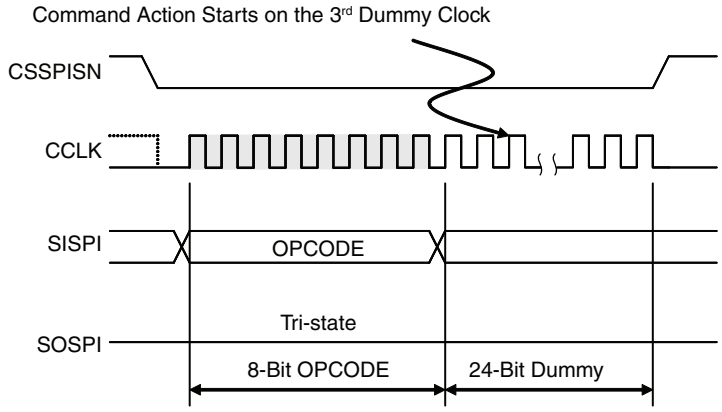
Note

1. These edges on the CSSPISN won't interrupt or terminate the erase or programming action in progress. The CCLK pulses are ignored while the CSSPISN is high.

Class E Commands Waveforms

The Class E commands do not require any data to be shifted in or out, and do not require any delay. The twenty four (24) dummy clocks provide the device the necessary delay for the proper execution of the command. Even if extra dummy clocks are presented, the device ignores them.

Figure 17. Class E Command Waveforms



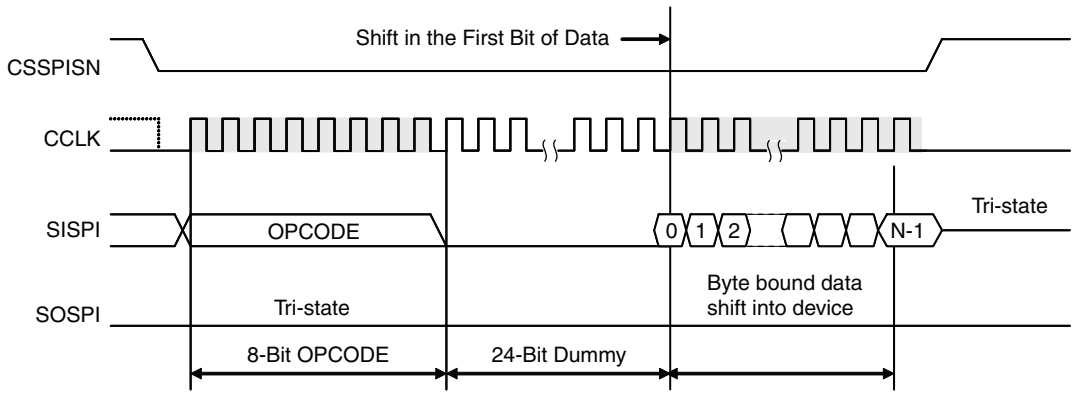
Note

1: For REFRESH command, minimum of two continuous clocks after the 8-bit opcode is required to initiate the refresh action. Only one clock or too much delay between the first and the second clock may result unsuccessful refresh.

Class F Commands Waveforms

The Class F commands shifts data into the device. It does not need to observe any delay. The twenty four (24) dummy clocks provide the device the necessary delay to properly execute the command.

Figure 18. Class F Command Waveforms



Class G Commands Waveforms

The Class G commands read data out from the device. The data can be Flash or SRAM programming data.

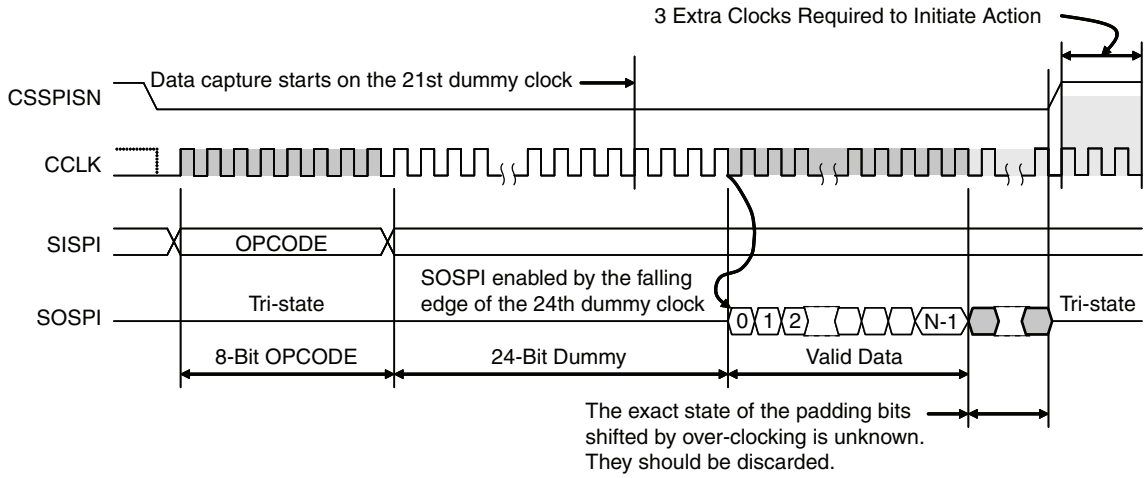
When reading Flash programming data, there is a 5µs verify delay time required to provide the device the time it needs to physically transferring the data storing in Flash cells into the FIFO data shift register. The twenty four (24) clocks used to shift in the dummy bits are considered part of the delay by the device. If the twenty four (24) clocks are applied faster than 5µs, then extra delay is required by holding the CCLK low before sampling SOSPI.

All commands in this class must be qualified by shifting in first the X_PROGRAM_EN command, which is equivalent to the WRITE_EN command in standard SPI Flash devices. If not, the transfer action will not happen, resulting in unknown data being shifted out.

If the programming data size is not divisible by eight (8), trailing padding clocks can be used to shift out byte bound data and then discard the residual bits clocked out by the trailing padding clocks. Data is not shifted out after the rising edge of CSSPISN. However, the device requires three (3) extra clocks in order to increment the address pointer to the next row of data.

Note: Please refer to Table 8 for the number of data bits *N* (columns) for each member of the LatticeXP2 family.

Figure 19. F Class G Command Waveforms



Specifications and Timing Diagrams

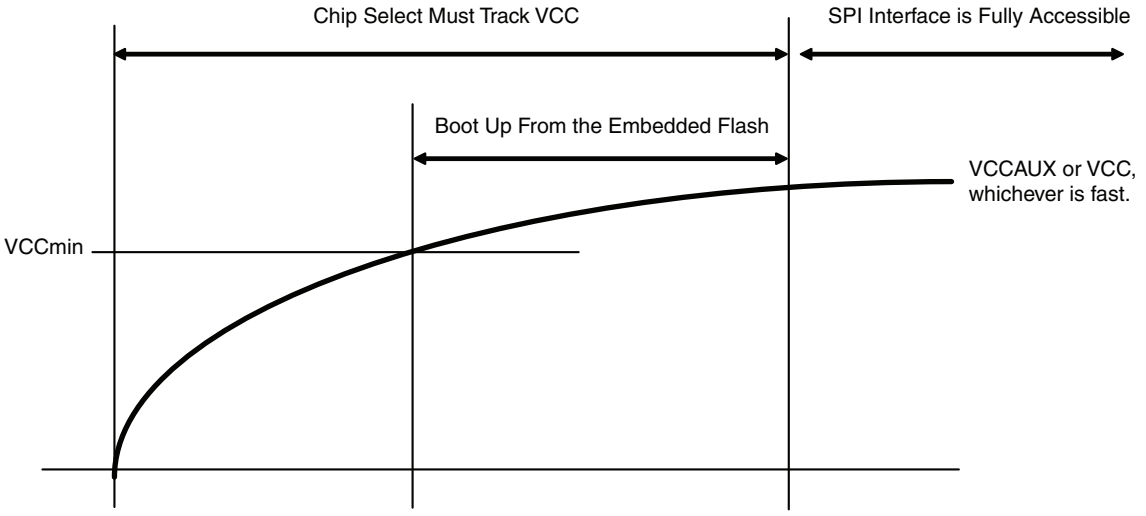
Power Up

If the device is configured to Self Download Mode (SDM), the Slave SPI port is available when the device completes the self-download. If the Flash Done fuse is not programmed, the self-download will not happen, and the Slave SPI port will be available immediately. However, it is recommended to wait the same amount of time as if the self-download was occuring before accessing the SPI port.

If the device is configured to Dual Boot Mode, and if the boot up is from an external SPI Flash boot PROM, then the DONE pin must be high before accessing the Slave SPI port. If the DONE pin is not high, the Slave SPI port is not available.

A low to high transition on the Chip Select pin (CSSPISN) is needed to reset the Slave SPI interface. During power up, the low to high transition is assured by requiring the CCLK pin to track the VCC. The other method is to drive the Chip Select pin to high, to low, and then high to reset the Slave SPI interface before shifting the first command into the device.

Figure 20. Power Up Timing and Voltage Level



Note: Please refer to the LatticeXP2 Datasheet for the VCCmin specifications.

Availability of the Slave SPI Port after Power Up

The Slave SPI Port is not available during the following events:

- When the JTAG Port is not at the Test-Logic-Reset state. The JTAG port is powered up into the Test-Logic-Reset state. If the JTAG port is used, before accessing the Slave SPI port, return the device to Test-Logic-Reset state by either toggling the TMS pin or by power cycle the device.
- During Refresh.
- During Reboot.

AC Timing

Table 5. Slave SPI AC Timing

Parameter	Value
Maximum CCLK	25 MHz
Minimum delay from VCCmin to shifting in the first command	2 ms
Minimum Refresh time (embedded Flash to SRAM transfer)	2 ms

Erase and Program Timing

The Slave SPI Program and Read Back timinig requirements are listed in Table 6. The Erase timinig requirements are listed in Table 6.

Table 6. Slave SPI Program and Read Timing

Device	Fuses	Program			
		Delay per Loop	Maximum Number of Loops per Row	Maximum Delay per Row	Minimum Delay
All Members	Embedded Flash	1ms	10	10ms	5µs per row
	TAG Memory	1ms	10	10ms	5µs
	Embedded SRAM				1µs per row

Table 7. Slave SPI Erase Timing

Device	Fuses	Erase		
		Delay per Loop	Maximum Number of Loops	Maximum Delay
LFXP2-5E	Embedded Flash	100ms	1200	120 sec
	TAG Memory	100ms	20	2 sec
LFXP2-8E	Embedded Flash	100ms	1800	180 sec
	TAG Memory	100ms	20	2 sec
LFXP2-17E	Embedded Flash	100ms	2400	240 sec
	TAG Memory	100ms	20	2 sec
LFXP2-30E	Embedded Flash	100ms	2800	280 sec
	TAG Memory	100ms	40	4 sec
LFXP2-40E	Embedded Flash	100ms	3000	300 sec
	TAG Memory	100ms	40	4 sec

All Lattice non-volatile devices support JEDEC files. Utilities are available in the ispVM System 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 Table 8 for completeness. The DOS program and the C source code are available from Lattice for converting the JEDEC file into the binary file format.

Table 8. JEDEC File Details

JEDEC Field	Syntax	Description
Don't Care	My design	Characters appear before ^B are don't care. All character sets or internal language can be used here except ^B.
Start-of-text	^B	^B (CTLB) 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 (see Note 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 actually 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> (see Note 1).

Table 8. JEDEC File Details (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 (7) digits, thus L0000000 should have seven (7) zeros (0). 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. Following the last row of data is the 32-bit SED_CRC value. If the SED feature is not used, the value will be all ones (32 bits of 1's). If the JEDEC file is encrypted, all the data in the link field are encrypted except the SED_CRC value. 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 (see Note 2).
E Field	EH 012..ABCDEF	JEDEC standard defines this field to hold the architecture fuses. Lattice changed this field to store the TAG Memory instead.
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.

Notes:
1. Only the JTAG port can program the OTP Feature fuse.
2. 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.

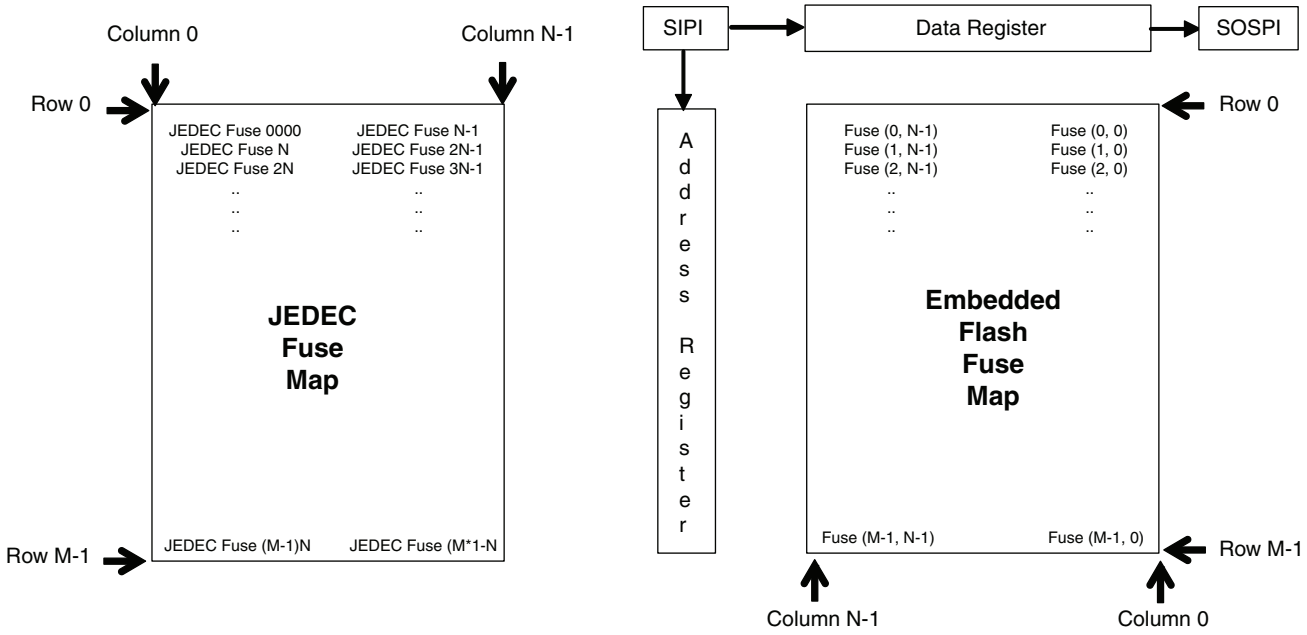
An example of a LatticeXP2 unencrypted JEDEC file is shown in Figure 21. The data highlighted in grey is the data for one row (or address) in the device. The LatticeXP2 contains M number of rows (or addresses). Each row has N number columns (or data locations) per row. The JEDEC mapping into the LatticeXP2 embedded Flash is shown in Figure 22. It is important to note that address bit 0 of the address shift register is connected to SISPI whereas bit 0 of the data register is connected to SOSPI. This assignment is due to the convention that addressing is in ascending order. Address 0 is the first address to be selected (by the INIT_ADDRESS command), whereas for data bit 0 is the first bit to be shifted out from the device. The PROGRAM_INC command programs one row of configuration Flash, then increments the row address.

Refer to Table 8 for the row and column lengths for each density.

Figure 21. Unencrypted JEDEC File Example

```
*
NOTE ispLever_v71_PROD_Build (58) JEDEC Compatible Fuse File.*
NOTE Copyright (C), 1992-2007, Lattice Semiconductor Corporation.*
NOTE All Rights Reserved.*
NOTE DATE CREATED:Fri Oct 03 13:45:53 2008*
NOTE DESIGN NAME:lfxp2_05et144.ncd*
NOTE DEVICE NAME:LFXP2-5E-5TQFP144*
NOTE PIN ASSIGNMENTS*
NOTE PINS seven_seg_3 : 143 : out*
NOTE PINS clk_in : 144 : in*
NOTE PINS seven_seg_0 : 130 : out*
NOTE PINS seven_seg_1 : 133 : out*
NOTE PINS seven_seg_2 : 142 : out*
QP144*
QF1236476*
G0*
F0*
L0000000
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
.
.
.
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111010111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111010111111010111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111111*
NOTE SED_CRC = 0xFE0F558F*
L1236444
11110001101010101111000001111111*
C3222*
NOTE User Electronic Signature Data*
UH01234567*
18D5
```

Figure 22. JEDEC Fuse Map vs. Embedded Flash Fuse MapDee Specifics



Device Specifics

The device specifics for the LatticeXP2 family are listed in Table 9

Table 9. LatticeXP2 Device Specifics

Device	32-bit JTAG IDCODE	Standard JEDEC				Encrypted JEDEC			TAG Memory Page Size
		Rows (M)	Column (N)	Padding Bits[3]	Density	Rows (M)	Column[1] (N)	Density[2]	
LFXP2-5E	0x01299043	1938	638	2	1,236,476	1938	640	1,240,352	632
LFXP2-8E	0x0129A043	2532	772	4	1,954,736	2532	896	2,268,704	768
LFXP2-17E	0x0129B043	1658	2188	4	3,627,704	1658	2304	3,820,032	2184
LFXP2-30E	0x0129D043	2252	2644	4	5,954,320	2252	2688	6,053,408	2640
LFXP2-40E	0x0129E043	2454	3384	0	8,304,368	2454	3456	8,481,056	3384

Notes:
1. The column size increases because the encrypted file must be packet size bounded (16 bytes per packet).
2. The fuse density increases due to the apparent increase in the column size to comply with the packet bound requirement. However, the real fuse density remains the same as shown under the Standard JEDEC column. The decryption engine will automatically discard the packet bounded filler bits during the programming process.
3. Byte-bounding padding bits if required by target system.

Advanced Applications – The Internal Slave SPI Interface

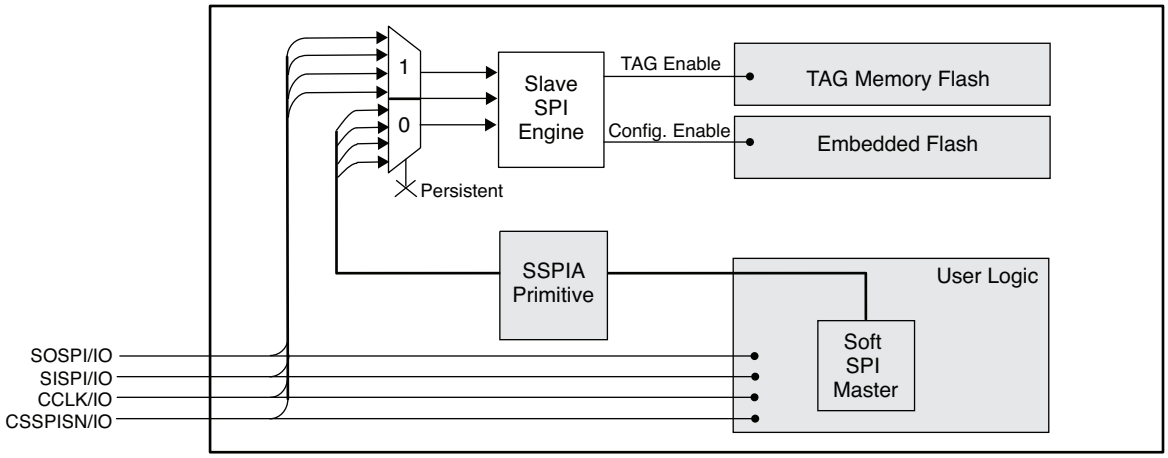
Once the device has been programmed either by using the JTAG port or the external Slave SPI port, the internal SPI port can then be selected to carry out the re-programming or field upgrade. One of the main concerns of field upgrade a device is due to un-controllable disruption like loss of power during field upgrade. The system would not be able to recover. This concern is magnified when using the internal SPI interface to support field upgrade.

In the event the power disruption happens before re-programming finished, it is obvious that the internal SPI interface will not be available. Only the external SPI interface will be available.

The best solution is to use the dual boot feature of the LatticeXP2 family of devices. The feature requires the Golden pattern be stored in an external standard SPI Flash device that will reboot the LatticeXP2 devices when field upgrade failure happens.

The second best solution is to use the same I/O pins to access the external and the internal SPI interface. This solution works due to the programming flow is exactly the same using internal external SPI Interface. The diagram of such an application is shown in Figure 23.

Figure 23. Internal and External Slave SPI Block Diagram



Description

The diagram shows the SSPIA primitive instantiated connects the four (4) user I/O pins to the Slave SPI Command Engine and the other user logical functions. It is strongly recommended to keep the CSSPISN pin dedicated as the select pin to multiplex the SOSPI, SISPI, and CCLK pin between the SSPIA primitive and the other logical functions. This is to take advantage of the fact that the Slave SPI Engine will not recognize the CCLK signal unless the CSSPISN pin is low. Regardless whether the persistent fuse is On or Off, the four (4) pins can always access the SPI command engine reliably to re-program the device.

The persistent fuse controls the multiplexer in front of the SPI command engine. The external SPI pins are only connected to the SPI command engine when the persistent fuse is set to On. The persistent fuse is On in the erased state (1), which is equivalent to setting the SLAVE_SPI_PORT to Enable in the in the Lattice Diamond™ Spreadsheet view, or the ispLEVER® Design Planner 'Global' tab.

The persistent fuse is On under the following circumstances.

1. The device is a blank part.
2. The done fuse is not programmed due to re-programming failure.
3. The device is programmed with the SLAVE_SPI_PORT set to Enable.

The persistent fuse is Off under the following circumstances.

1. The device is programmed with the SLAVE_SPI_PORT set to Disable.
2. The SSPIA primitive is instantiated.

Since the SSPIA module is an internal module, I/Os can be treated as I/Os of any other soft module. Therefore, they can be routed to other internal modules, or they can go out to I/O pads. The recommended routing is to the sysCONFIG port pins.

Figure 24. SSPIA Primitive

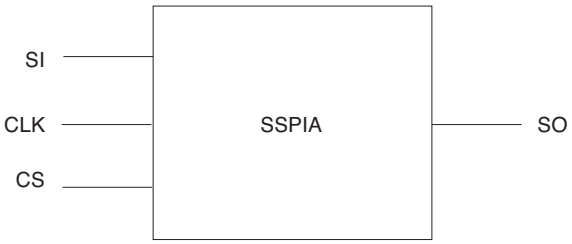


Table 10. SSPIA Primitive Signal Description

Primitive Port Name	Description
SI	Data input
SO	Data output
CLK	Clock
CS	Chip select

Verilog Example Code

```
module test (CLK, SI, CS, SO);
input wire CLK;
input wire SI;
input wire CS;
output wire SO;

SSPIA TAGInst_0 (.SI(SI), .CLK(CLK), .CS(CS), .SO(SO))
/* synthesis TAG_INITIALIZATION="DISABLED" */;

endmodule
```

VHDL Example Code

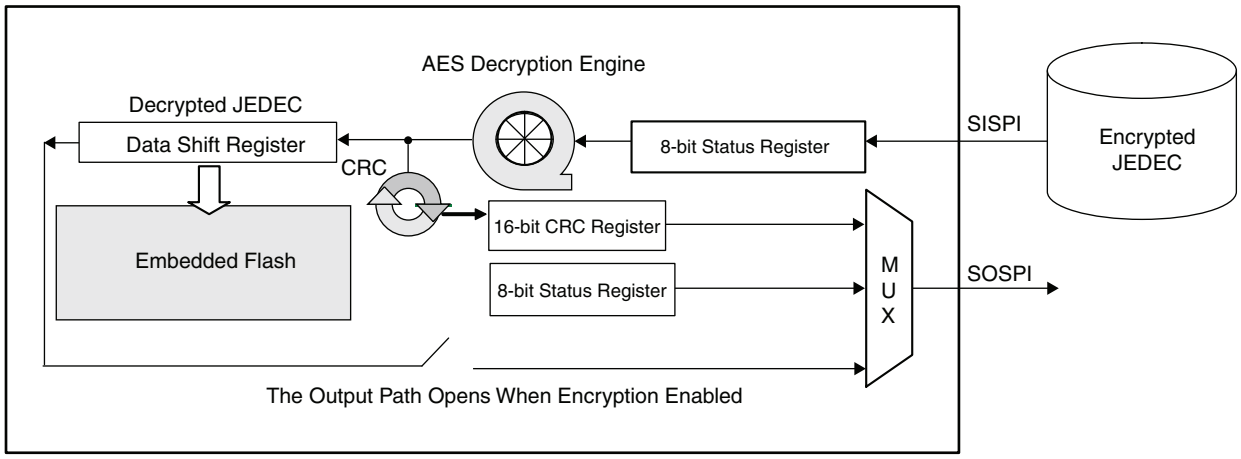
```
-- local component declarations
component SSPIA
port (SI: in  std_logic; CLK: in  std_logic; CS: in  std_logic;
  SO: out  std_logic);
end component;
attribute TAG_INITIALIZATION : string;
attribute TAG_INITIALIZATION of TAGInst_0 : label is "DISABLED";

begin
-- component instantiation statements
TAGInst_0: SSPIA
port map (SI=>SI, CLK=>CLK, CS=>CS, SO=>SO);
```

There are three special considerations when implement the encryption programming flow.

1. Provide the device the password to unlock the device for re-programming.
2. Authentication of the encrypted JEDEC file. The purpose is to make sure the Encryption Key programmed in the device and encrypting the JEDEC file matches. If there is a mismatch, the erroneous JEDEC programmed into the device could cause internal contention.
3. Check the CRC instead of read back and verify. The purpose is to confirm the entire JEDEC file is decrypted and programmed into the embedded Flash correctly.

The conceptual encryption programming circuitry in the LatticeXP2 family of devices is shown in Figure 25.

Figure 25. Decryption Block Diagram



Description

The 128-bit register can store only one packet of encrypted data.

The Data Shift Register size can store one column of un-encrypted data.

The decryption engine converts the incoming encrypted data into decrypted data. The decrypted data is shifted into the Data Shift Register as well as the CRC calculator.

The VERIFY_INC command would return unknown data due to the command is disabled when enabling the Encryption Feature. Verification through authentication is thus recommended.

The encrypt JEDEC can be authenticated by reading out the CRC value after shifting in the first row or all the rows and compare with the value written to the upper (left hand) or lower (right hand) sixteen (16) bits of the U field, respectively.

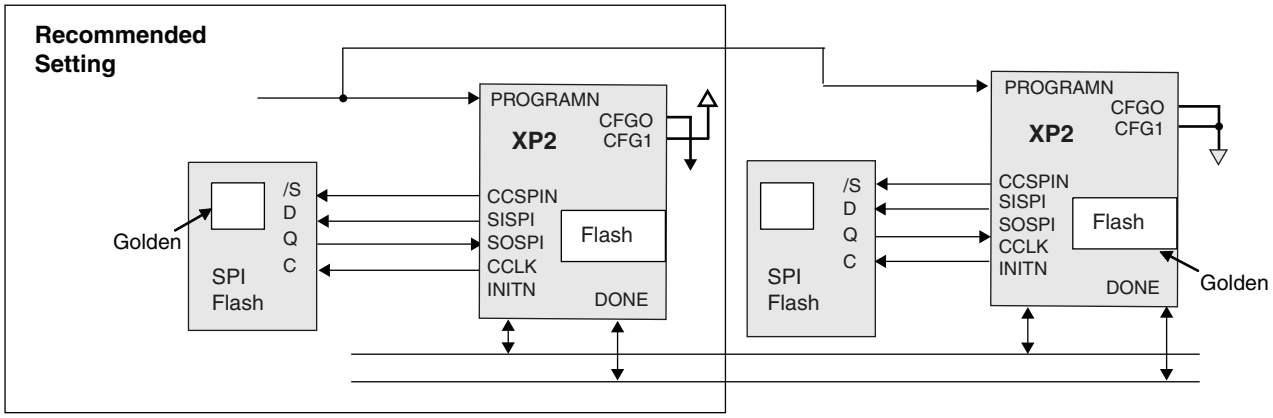
The programming flow is listed below.

1. Check bit 6 of the device status register for a one (1) to confirm the encryption feature is enabled. Refer to Table 1 for the definition of the device status register.
2. Check bit 5 of the device status register for a one (1) to confirm if Password is needed to unlock the device for re-programming.
3. Shift in the Flash Protect password and then check bit 2 of the device status register for a zero (0) for match.
4. Flow the rest of the encryption programming flow.

Advanced Applications – Dual Boot Configuration

The above documentation provides Slave SPI port programming information when the LatticeXP2 device is set to Self Download Mode only (CFG0 = 1). When the CFG0 pin is tied to low (logical 0), the device may load pattern from two sources: embedded Flash memory in the latticeXP2 device or external SPI Flash. This configuration mode is called Dual Boot mode. When the CFG0 is set to 0, several User IO pins will become dedicated pins to support the Dual Boot feature. One of the pin which becomes dedicated is the CFG1 pin. Figure 26 shows how the CFG1 configures the priority of pattern sources.

Figure 26. Master SPI Dual Boot Block Diagram

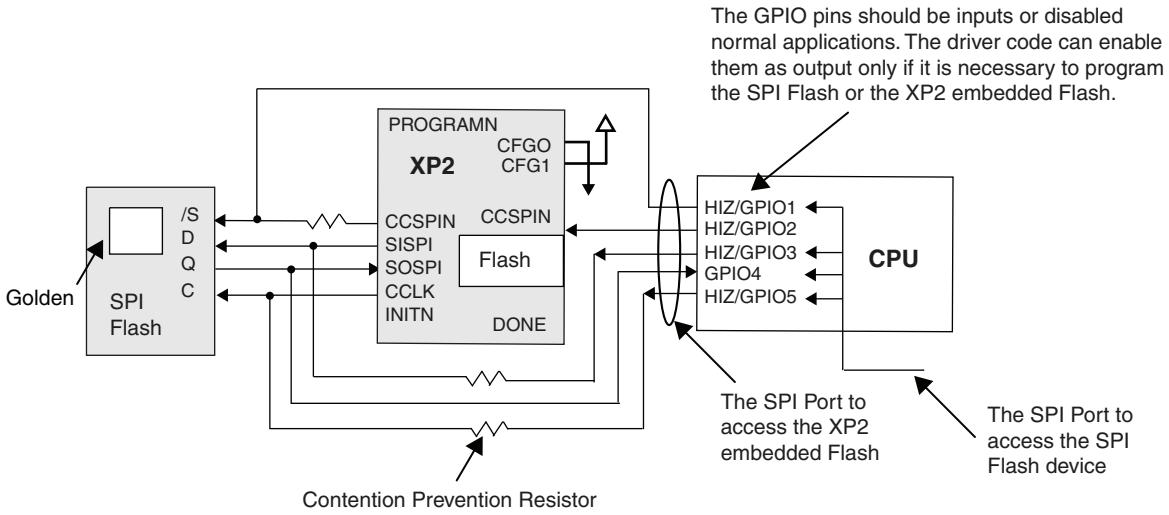


The advantage of Dual Boot is that it allows users to perform field upgrade more safely. When performing upgrade, the new pattern could be written to the Primary source, and the device will be able to wake up with new pattern. If the upgrade fails, the device can retrieve the pattern from the Golden source, which may contain logic to perform upgrade again.

The Slave SPI interface is very well suited for field-upgrade applications. It is quite obvious that the LatticeXP2 embedded Flash memory should contain the Primary boot pattern due to the security and fast boot up time it offers. Booting from the SPI Flash would be much slower and the bitstream in the SPI Flash does not support encryption.

Field upgrades will be done by a CPU. Board manufacturing programming also can use the same CPU. Figure 27 illustrates the necessary board connections.

Figure 27. Example Embedded Dual Boot Block Diagram

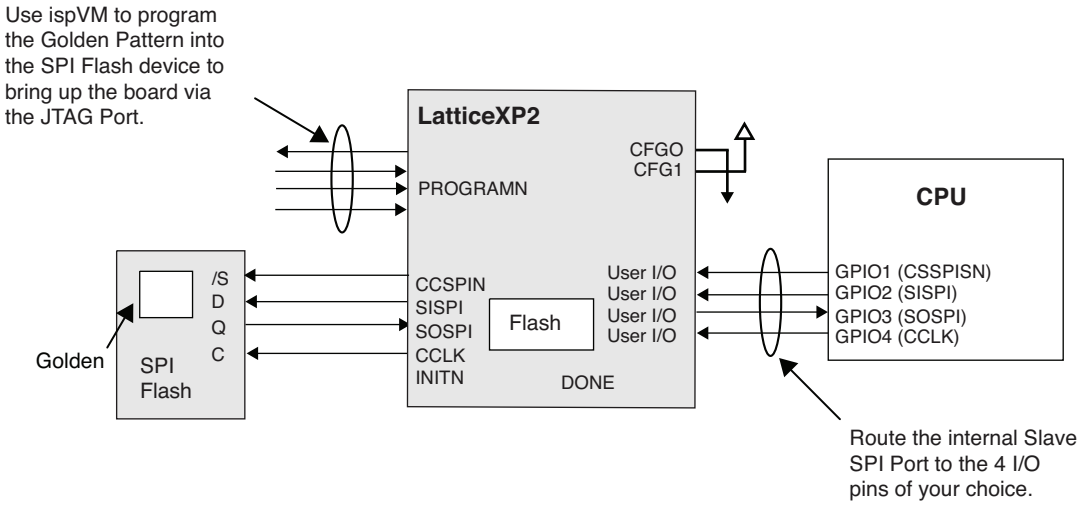


The connections shown in Figure 27 illustrate one method using the CPU to program the SPI Flash device and field-upgrade the LatticeXP2 device. The programming flow and sequence is strictly under software (user driver code) control. The only special requirement is that the GPIO pins of the CPU must be normally tri-stated. They should be enabled only when they are needed.

If the JTAG port is available, then ispVM can be used to program the Golden pattern into the SPI Flash device during the board design phase, as shown in Figure 28.The Golden pattern should then enable the internal SPI Port for

the CPU to program the embedded Flash of the LatticeXP2 device. Alternatively, the JTAG port can be used to program both the SPI Flash device and the LatticeXP2 device during the board development phase.

Figure 28. Example Embedded Dual Boot Block Diagram Using JTAG Port



It is recommended to program the Golden pattern into the SPI Flash during the board manufacturing phase. If the SPI Flash device cannot be pre-programmed, one alternative is to connect the JTAG port to four (4) GPIO pins of the CPU, port in the ispVME driver, and use ispVME instead of ispVM to program the SPI Flash devices during the board manufacturing phase. Optionally, the embedded Flash can also be programmed by ispVME as well during the manufacturing phase. This will save the internal SPI port only for the field upgrade. This approach is clean cut and there are no switches required. It is all done by software.

Reprogramming a Secured Device

When reprogramming the embedded Flash on a secured device, the device will fail verification. This is due to the security fuse latch is not cleared by the Erase command. Below are the two work-around solutions for reprogramming a secured LatticeXP2 device using the Slave SPI flow.

Direct Slave SPI Programming Mode Flow

1. Erase the device.
2. Disable Slave SPI configuration mode by issuing the ISC_DISABLE (0x78) command.
3. Issue the SLAVE_SPI_REFRESH command (0xC4) to wake-up the device and clear the security fuse latch.
4. Enable Slave SPI background programming mode back by issuing the XPROGRAM_ENABLE (0xAC) command and continue the programming flow.

Background Slave SPI Programming Mode Flow

1. Follow the regular Erase and Programming flow.
2. Skip the Flash verification.

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.
April 2011	01.1	Updated Figures 4-10.
		Updated Table 3. Slave SPI Command Table
		Updated Table 4. Slave SPI Command Usage Table
		Updated Figure 23. Internal and External Slave SPI Block Diagram
June 2012	01.2	Updated Program SED CRC and Verify Flow Diagram.
		Updated Refresh Flow Diagram.
		Updated Slave SPI Command Usage Table.