LibXil Isf (v5.6)

LibXil Isf Library Overview

The LibXil Isf library:

- Allows you to Write, Read, and Erase the Serial Flash.
- Allows protection of the data stored in the Serial Flash from unwarranted modification by enabling the Sector Protection feature.
- Supports multiple instances of Serial Flash at a time, provided they are of the same device family (Atmel, Intel, STM, Winbond, SST, or Spansion) as the device family is selected at compile time.
- Allows the user application to perform Control operations on Intel, STM, Winbond, SST, and Spansion Serial Flash.
- Requires the underlying hardware platform to contain the axi_quad_spi, ps7_spi, ps7_qspi, psu_qspi or psu_spi device for accessing the Serial Flash.
- Uses the Xilinx® SPI interface drivers in interrupt-driven mode or polled mode for communicating with the Serial Flash. In interrupt mode, the user application must acknowledge any associated interrupts from the Interrupt Controller.

Additional information:

- In interrupt mode, the application is required to register a callback to the library and the library registers an internal status handler to the selected interface driver.
- When the user application requests a library operation, it is initiated and control is given back to the application. The library tracks the status of the interface transfers, and notifies the user application upon completion of the selected library operation.
- Added support in the library for SPI PS and QSPI PS. You must select one of the interfaces at compile time.
- Added support for QSPIPSU and SPIPS flash interface on Zynq® UltraScale™+ MPSoC.
- When the user application requests selection of QSPIPS interface during compilation, the QSPI PS or QSPI PSU interface, based on the hardware platform, are selected. Similarly, if the SPIPS interface is selected during compilation, SPI PS or SPI PSU interface are selected.

Supported Devices

Table 1 lists the supported Xilinx In-System Flash and external Serial Flash Memories.

Table 1: Xilinx In-System Flash and External Serial Flash Memories

Device Series	Manufacturer
AT45DB011D AT45DB021D AT45DB041D AT45DB081D AT45DB161D AT45DB321D AT45DB642D	Atmel
W25Q16 W25Q32 W25Q64 W25Q80 W25Q128 W25X10 W25X20 W25X40 W25X80 W25X16 W25X32 W25X64	Winbond
S25FL004 S25FL008 S25FL016 S25FL032 S25FL064 S25FL128 S25FL129 S25FL256 S25FL512 S70FL01G	Spansion
SST25WF080	SST
N25Q032 N25Q064 N25Q128 N25Q256 N25Q512 N25Q00AA MT25Q01 MT25Q02	Micron[1]

1. Intel, STM, and Numonyx Serial Flash devices are now a part of Serial Flash devices provided by Micron.

LibXil Isf Library APIs

This section provides a linked summary and detailed descriptions of the LibXil Isf library APIs.

API Summary

The following is a summary list of APIs provided by the LibXil Isf library. The list is linked to the API description. Click the API name to go to the description.

int XIsf_Initialize(XIsf *InstancePtr, XSpi *SpiInstPtr, u32 SlaveSelect, u8 *WritePtr)
int XIsf_GetStatus(XIsf *InstancePtr, u8 *ReadPtr)
int XIsf_GetStatusReg2(XIsf *InstancePtr, u8 *ReadPtr)
int XIsf_GetDeviceInfo(XIsf *InstancePtr, u8 *ReadPtr)
int XIsf_Read(XIsf *InstancePtr, XIsf_ReadOperation Operation, void *OpParamPtr)
int XIsf_Write(XIsf *InstancePtr, XIsf_WriteOperation Operation, void *OpParamPtr)
int XIsf_Erase(XIsf *InstancePtr, XIsf_EraseOperation Operation, u32 Address)
void XIsf_SetStatusHandler(XIsf*InstancePtr, XIsf_Iface *XIfaceInstancePtr XIsf_StatusHandler XilIsf_Handler);
int XIsf_SectorProtect(XIsf *InstancePtr, XIsf_SpOperation Operation, u8 *BufferPtr)
int XIsf_WriteEnable(XIsf *InstancePtr, u8 WriteEnable)
int XIsf_Ioctl (XIsf *InstancePtr, XIsf_IoctlOperation Operation)
int XIsf_SetSpiConfiguration(XIsf *InstancePtr, XIsf_Iface *SpiInstPtr, u32 Options, u8 PreScaler)
inline void XIsf_SetTransferMode(XIsf *InstancePtr, u8 Mode)
int XIsf_MicronFlashEnter4BAddMode(XIsf *InstancePtr)
int XIsf_MicronFlashExit4BAddMode(XIsf *InstancePtr)

LibXil Isf API Descriptions

`int` **`XIsf_Initialize`**`(XIsf *InstancePtr, XSpi *SpiInstPtr, u32 SlaveSelect, u8 *WritePtr)`

Parameters

`InstancePtr` is a pointer to the XIsf instance.

`SpiInstPtr` is a pointer to the XSpi instance to be worked on.

`SlaveSelect` is a 32-bit mask with a 1 in the bit position of the slave being selected. Only one slave can be selected at a time.

`WritePtr` is a pointer to the buffer allocated for use by the In-system and Serial Flash Library to perform any read/write operations on the Serial Flash device.

User applications must initialize the Isf library by passing the address of this buffer to the Initialization API.

For Write operations:

- A minimum of one byte and a maximum of `ISF_PAGE_SIZE` bytes can be written to the Serial Flash, through a single Write operation.
- The buffer size must be equal to the number of bytes to be written to the Serial Flash + `XISF_CMD_MAX_EXTRA_BYTES`, and must be large enough for use across the applications that use a common instance of the Serial Flash.

For Non Write operations:

- The buffer size must be equal to `XISF_CMD_MAX_EXTRA_BYTES`.

Returns

`XST_SUCCESS` upon success.

`XST_DEVICE_IS_STOPPED` if the device must be started before transferring data.

`XST_FAILURE` upon failure.

Description

The geometry of the underlying Serial Flash is determined by reading the Joint Electron Device Engineering Council (JEDEC®) Device Information and the Serial Flash Status Register.

When called, this API initializes the SPI interface with default settings. With custom settings, the user should call `XIsf_SetSpiConfiguration()` before calling this API.

Note: The `XIsf_Initialize()` API is a blocking call (for both polled mode and interrupt mode of the SPI driver). It reads the JEDEC information of the device and waits till the transfer is complete before checking if the information is valid.

Support multiple instances of Serial Flash at a time, provided they are of the same device family (either Atmel, Intel, STM, Winbond, or SST) as the device family is selected at compile time.

Includes

xilisf.h

`int` **`XIsf_GetStatus`**`(XIsf *InstancePtr, u8 *ReadPtr)`

Parameters	*InstancePtr* is a pointer to the XIsf instance. *ReadPtr* is a pointer to the memory where the Status Register content is copied.
Returns	XST_SUCCESS upon success XST_FAILURE upon failure
Description	Reads the Serial Flash Status Register. ***Note:*** The status register content is stored at the second byte pointed by the `ReadPtr`.
Includes	xilisf.h

`int` **`XIsf_GetStatusReg2`**`(XIsf *InstancePtr, u8 *ReadPtr)`

Parameters	*InstancePtr* is a pointer to the XIsf instance. *ReadPtr* is a pointer to the memory where the Status Register content is copied.
Returns	XST_SUCCESS upon success XST_FAILURE upon failure
Description	Reads the Serial Flash Status Register2. this API is valid only for Windbond (W25QXX) flash devices. ***Note:*** The status register content is stored at the second byte pointed by the `ReadPtr`.
Includes	xilisf.h

`int` **`XIsf_GetDeviceInfo`**`(XIsf *InstancePtr, u8 *ReadPtr)`

Parameters	*InstancePtr* is a pointer to the XIsf instance. `ReadPtr` is a pointer to the memory where the Device information is copied.
Returns	XST_SUCCESS upon success. XST_FAILURE upon failure.
Description	Reads the JEDEC information of the Serial Flash. ***Note:*** The Device information is stored at the second byte pointed by the `ReadPtr`.
Includes	xilisf.h

```
int XIsf_Read(XIsf *InstancePtr, XIsf_ReadOperation
   Operation, void *OpParamPtr)
```

Parameters

InstancePtr is a pointer to the XIsf instance.

Operation is the type of the read operation to be performed on the Serial Flash.

The *Operation* options are:

XISF_READ: Normal Read

XISF_FAST_READ: Fast Read

XISF_PAGE_TO_BUF_TRANS: Page to Buffer Transfer

XISF_BUFFER_READ: Buffer Read

XISF_FAST_BUFFER_READ: Fast Buffer Read

XISF_OTP_READ: One Time Programmable Area (OTP) Read.

XISF_DUAL_OP_FAST_READ: Dual Output Fast Read

XISF_DUAL_IO_FAST_READ: Dual Input/Output Fast Read

XISF_QUAD_OP_FAST_READ: Quad Output Fast Read

XISF_QUAD_IO_FAST_READ: Quad Input/Output Fast Read

OpParamPtr is the pointer to structure variable which contains operational parameter of specified Operation. This parameter type is dependent on the type of Operation to be performed.

When specifying Normal Read (XISF_READ), Fast Read (XISF_FAST_READ) and One Time Programmable Area Read(XISF_OTP_READ), Dual Output Fast Read

(XISF_DUAL_OP_FAST_READ), Dual Input/Output Fast Read

(XISF_DUAL_IO_FAST_READ), Quad Output Fast Read

(XISF_QUAD_OP_FAST_READ)and Quad Input/Output Fast Read (XISF_QUAD_IO_FAST_READ):

- *OpParamPtr* must be of type struct *XIsf_ReadParam*.
- *OpParamPtr->Address* is the start address in the Serial Flash.
- *OpParamPtr->ReadPtr* is a pointer to the memory where the data read from the Serial Flash is stored.
- *OpParamPtr->NumBytes* is number of bytes to read.
- *OpParamPtr->NumDummyBytes* is the number of dummy bytes to be transmitted for the Read command. This parameter is only used in case of Dual and Quad reads.

Normal Read and Fast Read operations are supported for Atmel, Intel, STM, Winbond, SST, and Spansion Serial Flash. Dual and quad reads are supported for Winbond (W25QXX),Micron (N25QXX) and Spansion (S25FL129) quad flash. OTP Read operation is only supported in Intel Serial Flash.

When specifying Page To Buffer Transfer (XISF_PAGE_TO_BUF_TRANS):

- *OpParamPtr* must be of type struct *XIsf_FlashToBufTransferParam*.
- *OpParamPtr->BufferNum* specifies the internal SRAM Buffer of the Serial Flash. The valid values are XISF_PAGE_BUFFER1 or XISF_PAGE_BUFFER2. XISF_PAGE_BUFFER2 is not valid in the case of AT45DB011D Flash as it contains a single buffer.
- *OpParamPtr->Address* is start address in the Serial Flash.

This operation is only supported in Atmel Serial Flash.

XIsf_Read (*continued*)

Parameters	When specifying Buffer Read (XISF_BUFFER_READ) and Fast Buffer Read (XISF_FAST_BUFFER_READ): ″ *OpParamPtr* must be of type struct *XIsf_BufferReadParam*. ″ *OpParamPtr->BufferNum* specifies the internal SRAM Buffer of the Serial Flash. The valid values are XISF_PAGE_BUFFER1 or XISF_PAGE_BUFFER2. XISF_PAGE_BUFFER2 is not valid in the case of AT45DB011D Flash as it contains a single buffer. ″ *OpParamPtr->ReadPtr* is pointer to the memory where the data read from the SRAM buffer is to be stored. ″ *OpParamPtr->ByteOffset* is byte offset in the SRAM buffer from where the first byte is read. ″ *OpParamPtr->NumBytes* is the number of bytes to be read from the Buffer. These operations are supported only in Atmel Serial Flash.
Returns	XST_SUCCESS upon success. XST_FAILURE upon failure.
Description	Reads the data from the Serial Flash. ***Note:*** Application must fill the structure elements of the third argument and pass its pointer by type casting it with void pointer. The valid data is available from the fourth location pointed to by the ReadPtr for Normal Read and Buffer Read operations. The valid data is available from the fifth location pointed to by the ReadPtr for Fast Read, Fast Buffer Read, and OTP Read operations. The valid data is available from the (4 + *NumDummyBytes*) location pointed to by ReadPtr for Dual/Quad Read operations.
Includes	xilisf.h

```
int XIsf_Write(XIsf *InstancePtr, XIsf_WriteOperation
   Operation, void *OpParamPtr)
```

Parameters

InstancePtr is a pointer to the XIsf instance.

Operation is the type of write operation to be performed on the Serial Flash.

The *Operation* options are:

- XISF_WRITE: Normal Write
- XISF_DUAL_IP_PAGE_WRITE: Dual Input Fast Program
- XISF_DUAL_IP_EXT_PAGE_WRITE: Dual Input Extended Fast Program
- XISF_QUAD_IP_PAGE_WRITE: Quad Input Fast Program
- XISF_QUAD_IP_EXT_PAGE_WRITE: Quad Input Extended Fast Program
- XISF_AUTO_PAGE_WRITE: Auto Page Write
- XISF_BUFFER_WRITE: Buffer Write
- XISF_BUF_TO_PAGE_WRITE_WITH_ERASE: Buffer to Page Transfer with Erase
- XISF_BUF_TO_PAGE_WRITE_WITHOUT_ERASE: Buffer to Page Transfer without Erase
- XISF_WRITE_STATUS_REG: Status Register Write
- XISF_WRITE_STATUS_REG2: 2 byte Status Register Write
- XISF_OTP_WRITE: OTP Write.

OpParamPtr is the pointer to a structure variable which contains operational parameters of specified operation.

This parameter type is dependant upon the value of first argument (*Operation*).

When specifying Normal Write (XISF_WRITE): Dual Input Fast Program (XISF_DUAL_IP_PAGE_WRITE), Dual Input Extended Fast Program (XISF_DUAL_IP_EXT_PAGE_WRITE), Quad Input Fast Program (XISF_QUAD_IP_PAGE_WRITE), Quad Input Extended Fast Program (XISF_QUAD_IP_EXT_PAGE_WRITE):

- *OpParamPtr* must be of type struct *XIsf_WriteParam*.
- *OpParamPtr->Address* is the start address in the Serial Flash.
- *OpParamPtr->WritePtr* is a pointer to the data to be written to the Serial Flash.
- *OpParamPtr->NumBytes* is the number of bytes to be written to the Serial Flash.

This operation is supported for Atmel, Intel, STM, Winbond, and Spansion Serial Flash.

For SST, only normal write is applicable.

When specifying the Auto Page Write (XISF_AUTO_PAGE_WRITE):

- *OpParamPtr* must be of 32 bit unsigned integer variable. This is the address of page number in the Serial Flash which is to be refreshed.

This operation is only supported in Atmel Serial Flash.

When specifying the Buffer Write (XISF_BUFFER_WRITE):

- *OpParamPtr* must be of type struct *XIsf_BufferWriteParam*.
- *OpParamPtr->BufferNum* specifies the internal SRAM Buffer of the Serial Flash. The valid values are XISF_PAGE_BUFFER1 or XISF_PAGE_BUFFER2. XISF_PAGE_BUFFER2 is not valid in the case of AT45DB011D Flash as it contains a single buffer.
- *OpParamPtr->WritePtr* is a pointer to the data to be written to the Serial Flash SRAM Buffer.
- *OpParamPtr->ByteOffset* is byte offset in the buffer from where the data is to be written.
- *OpParamPtr->NumBytes* is number of bytes to be written to the Buffer.

XIsf_Write (*continued*)

Parameters

This operation is supported only for Atmel Serial Flash.When specifying Buffer To Memory Write With Erase (XISF_BUF_TO_PAGE_WRITE_WITH_ERASE) or Buffer To Memory Write Without Erase (XISF_BUF_TO_PAGE_WRITE_WITHOUT_ERASE):

- *OpParamPtr* must be of type struct *XIsf_BufferToFlashWriteParam*.
- *OpParamPtr->BufferNum* specifies the internal SRAM Buffer of the Serial Flash. The valid values are XISF_PAGE_BUFFER1 or XISF_PAGE_BUFFER2. XISF_PAGE_BUFFER2 is not valid in the case of AT45DB011D Flash as it contains a single buffer.
- *OpParamPtr->Address* is starting address in the Serial Flash memory from where the data is to be written.

These operations are only supported in Atmel Serial Flash.

When specifying Write Status Register (XISF_WRITE_STATUS_REG), the *OpParamPtr* must be an 8-bit unsigned integer variable. This is the value to be written to the Status Register.

This operation is supported in Intel, STM, SST, and Winbond Serial Flash only.

When specifying Write 2 Byte Status Register (XISF_WRITE_STATUS_REG2), the *OpParamPtr* must be of type (u8 *) and should point to two 8 bit unsigned integer values. This is the value to be written to the 16 bit Status Register

Note: This operation is supported only in Winbond (W25QXX) Serial Flash.

When specifying One Time Programmable Area Write (XISF_OTP_WRITE):

- *OpParamPtr* must be of type struct XIsf_WriteParam.
- *OpParamPtr->Address* is the address in the SRAM Buffer of the Serial Flash to which the data is to be written.
- *OpParamPtr->WritePtr* is a pointer to the data to be written to the Serial Flash.
- *OpParamPtr->NumBytes* should be set to 1 when performing OTPWrite operation.

This operation is only supported in Intel Serial Flash.

Returns

XST_SUCCESS upon success.

XST_FAILURE upon failure.

Description

Writes data to the Serial Flash.

Note: Application must fill the structure elements of the third argument and pass its pointer by type casting it with void pointer.

For Intel, STM, Winbond, SST, and Spansion Serial Flash the user application must call the XIsf_WriteEnable() API by passing XISF_WRITE_ENABLE as an argument before calling the XIsf_Write() API.

Includes

xilisf.h

`int` **`XIsf_Erase`**`(XIsf *InstancePtr, XIsf_EraseOperation Operation, u32 Address)`

Parameters	*InstancePtr* is a pointer to the XIsf instance. *Operation* is the type of Erase operation to be performed on the Serial Flash. The different operations are " XISF_PAGE_ERASE: Page Erase " XISF_BLOCK_ERASE: Block Erase " XISF_SECTOR_ERASE: Sector Erase " XISF_BULK_ERASE: Bulk Erase *Address* is the address of the Page/Block/Sector to be erased. The address can be either Page address, Block address or Sector address based on the Erase operation to be performed.
Returns	XST_SUCCESS upon success. XST_FAILURE upon failure.
Description	Erases the contents of the specified memory in the Serial Flash. ***Note:*** The erased bytes will read as 0xFF. For Intel, STM, Winbond, and Spansion Serial Flash the user application must call `XIsf_WriteEnable()` API by passing `XISF_WRITE_ENABLE` as an argument before calling the `XIsf_Erase()` API. Atmel, Intel, STM Winbond, Micron (N25QXX), and Spansion Serial Flash devices support Sector/Block/Bulk Erase operations. SST devices support all Erase commands.
Includes	`xilisf.h`

`void` **`XIsf_SetStatusHandler`**`(XIsf*InstancePtr, XIsf_Iface *XIfaceInstancePtr XIsf_StatusHandler XilIsf_Handler);`

Parameters	*InstancePtr* is a pointer to the XIsf instance. *XIfaceInstancePtr* is a pointer to the XIsf_Iface instance to be worked on. *XilIsf_Handler* is the status handler for the application.
Returns	**None**
Description	Sets the application status handler. The library will register an internal handler to the interface driver.
Includes	`xilisf.h`

`int` **`XIsf_SectorProtect`**`(XIsf *InstancePtr, XIsf_SpOperation Operation, u8 *BufferPtr)`

Parameters	*InstancePtr* is a pointer to the XIsf instance. *Operation* is the type of Sector Protect operation to be performed on the Serial Flash. The Operation options are ″ XISF_SPR_READ: Read Sector Protection Register ″ XISF_SPR_WRITE: Write Sector Protection Register ″ XISF_SPR_ERASE: Erase Sector Protection Register ″ XISF_SP_ENABLE: Enable Sector Protection ″ XISF_SP_DISABLE: Disable Sector Protection *BufferPtr* is a pointer to the memory where the SPR content is read to/written from. This argument can be NULL if the Operation is SprErase, SpEnable and SpDisable.
Returns	XST_SUCCESS upon success. XST_FAILURE upon failure.
Description	Performs Sector Protect operations. ***Note:*** The SPR content is stored at the fourth location pointed by the BufferPtr when performing XISF_SPR_READ operation. For Intel, STM, Winbond, and Spansion Serial Flash devices the user application must call the XIsf_WriteEnable() API by passing XISF_WRITE_ENABLE as an argument, before calling the XIsf_SectorProtect() API, for Sector Protect Register Write (XISF_SPR_WRITE) operation. Atmel Flash supports all these Sector Protect operations. Intel, STM, Winbond, and Spansion support only Sector Protect Read and Sector Protect Write operations.
Includes	xilisf.h

`int` **`XIsf_WriteEnable`**`(XIsf *InstancePtr, u8 WriteEnable)`

Parameters	*InstancePtr* is a pointer to the XIsf instance. *WriteEnable* specifies whether to enable (XISF_WRITE_ENABLE) or disable (XISF_WRITE_DISABLE) the writes to the Serial Flash.
Returns	XST_SUCCESS upon success. XST_FAILURE upon failure.
Description	Enables/Disables writes to the Intel, STM, Winbond, SST, and Spansion Serial Flash. ***Note:*** If this API is called for Atmel Flash, XST_FAILURE is returned.
Includes	xilisf.h

`int` **`XIsf_Ioctl`** `(XIsf *`*`InstancePtr`*`, XIsf_IoctlOperation` *`Operation`*`)`

Parameters	*`InstancePtr`* is a pointer to the XIsf instance. *`Operation`* is the type of Control operation to be performed on the Serial Flash. The control `Operations` options are: " `XISF_RELEASE_DPD`: Release from Deep Power Down (DPD) Mode " `XISF_ENTER_DPD`: Enter DPD Mode " `XISF_CLEAR_SR_FAIL_FLAGS`: Clear the Status Register Fail Flags.
Returns	`XST_SUCCESS` upon success. `XST_FAILURE` upon failure.
Description	This API configures and controls the Intel, STM, Winbond, and Spansion Serial Flash. ***Note:*** Atmel Serial Flash does not support any of these operations. Intel Serial Flash support Enter/Release from DPD Mode and Clear Status Register Fail Flags. STM, Winbond, and Spansion Serial Flash support Enter/Release from DPD Mode. Winbond (W25QXX) supports Enable High performance mode.
Includes	`xilisf.h`

`int` **`XIsf_SetSpiConfiguration`**`(XIsf *InstancePtr, XIsf_Iface *SpiInstPtr, u32 Options, u8 PreScaler)`

Parameters	*`InstancePtr`* is a pointer to the `XIsf` instance. *SpiInstPtr* is a pointer to the `XIsf_Iface` instance to be worked on. *Options* contains specified options to be set. *PreScaler* is the value of the clock prescaler to set.
Returns	`XST_SUCCESS` upon success. `XST_FAILURE` upon failure.
Description	Sets the configuration of SPI. This API can be called before calling XIsf_Initialize() to operate the SPI interface in a mode other than the default options mode. *PreScaler* is only applicable to PS SPI/QSPI.
Includes	`xilisf.h`

`inline void` **`XIsf_SetTransferMode`**`(XIsf *InstancePtr, u8 Mode)`

Parameters	*`InstancePtr`* is a pointer to the XIsf instance. *Mode* is the value to be set.
Returns	None.
Description	This API sets the interrupt/polling mode of transfer. By default, the xilisf library is designed to operate in polling mode. User needs to call this API, if operating in Interrupt Mode.

`int` **`XIsf_MicronFlashEnter4BAddMode`**`(XIsf *InstancePtr)`

Parameters	*`InstancePtr`* is a pointer to the XIsf instance.
Returns	XST_SUCCESS upon success. XST_FAILURE upon failure.
Description	By default, flash is in 3B Address mode. User needs to call this API to enter the flash in the 4B Address mode.

`int` **`XIsf_MicronFlashExit4BAddMode`**`(XIsf *InstancePtr)`

Parameters	*`InstancePtr`* is a pointer to the XIsf instance.
Returns	XST_SUCCESS upon success. XST_FAILURE upon failure.
Description	User needs to call this API, if the flash is operating in the 4B address mode. Calling this API exits the flash from the 4B address mode.

Libgen Customization

The LibXil Isf library can be integrated with a system using the following snippet in the Microprocessor Software Specification (MSS) file.

```
BEGIN LIBRARY
  parameter LIBRARY_NAME = xilisf
  parameter LIBRARY_VER = 5.6
  parameter serial_flash_family = 1
  parameter serial_flash_interface = 1
END
```

Where:

- `LIBRARY_NAME`—Is the library name (`xilisf`).
- `LIBRARY_VER`—Is the library version (`5.6`).
- `serial_flash_family`—Is a numerical value representing the serial flash family, where:
  - 1 = Xilinx In-system Flash or Atmel Serial Flash
  - 2 = Intel (Numonyx) S33 Serial Flash[1]
  - 3 = STM (Numonyx) M25PXX/N25QXX Serial Flash[1]
  - 4 = Winbond Serial Flash
  - 5 = Spansion Serial Flash/Micron Serial Flash
  - 6 = SST Serial Flash
- `serial_flash_interface` - Is a numerical value representing the serial flash interface, where:
  - 1 = AXI QSPI Interface
  - 2 = SPI PS Interface
  - 3 = QSPI PS Interface or QSPI PSU Interface

Intel/STM/Numonyx serial flash devices now belong to the Micron family.

Additional Resources

- *Spartan-3AN FPGA In-System Flash User Guide* (UG333): http://www.xilinx.com/support/documentation/user_guides/ug333.pdf
- Atmel Serial Flash Memory website (AT45XXXD): http://www.atmel.com/dyn/products/devices.asp?family_id=616#1802
- Intel (Numonyx) S33 Serial Flash Memory website (S33): http://www.numonyx.com/Documents/Datasheets/314822_S33_Discrete_DS.pdf
- STM (Numonyx) M25PXX Serial Flash Memory website (M25PXX): http://www.numonyx.com/en-US/MemoryProducts/NORserial/Pages/M25PTechnicalDocuments.aspx
- Winbond Serial Flash Page: http://www.winbond-usa.com/hq/enu/ProductAndSales/ProductLines/FlashMemory/SerialFlash/
- Spansion website: http://www.spansion.com/Support/Pages/DatasheetsIndex.aspx
- SST SST25WF080: http://www.sst.com/dotAsset/40369.pdf
- Micron N25Q flash family: http://www.micron.com/products/nor-flash/serial-nor-flash/n25q#/