LatticeMico SPI Flash Controller

The LatticeMico Serial Peripheral Interface (SPI) flash controller is a WISHBONE slave device that provides an industry-standard interface between a central processing unit (CPU) and an off-chip SPI flash memory device. The controller has two separate WISHBONE slave ports: Port S and Port C. Port S can be used by the CPU to read from, or write to, any memory location within the SPI flash. Port C provides a mechanism to configure the SPI flash and issue any command from the SPI Flash's command set.

Version

This document describes the 3.5 version of the LatticeMico SPI flash controller.

Features

The LatticeMico SPI flash controller includes the following features:

- Two WISHBONE B.3 slave interfaces: Port S (Data Port) and Port C (Control Port)
- Option to enable/disable Control Port
- Option to individually configure the data bus widths of Port S and Port C to 8 or 32 bits
- Option to configure controller with Page Program Buffer to speed up Page Program
- Option to configure controller with Page Read Buffer to speed up Page Read
- Configurable serial clock (SCLK) frequency

- Configurable SPI flash sector size
- Configurable SPI flash page size
- Configurable SPI flash command set

For additional details about the WISHBONE bus, refer to the *LatticeMico8 Processor Reference Manual* or the *LatticeMico32 Processor Reference Manual*.

Functional Description

The SPI flash controller hardware is composed of two modules:

- Module wb_intf receives and processes WISHBONE Data Port (Port S) and Control Port (Port C) signals and transfers the requested SPI flash command to the spi_flash_intf module.
- Module spi_flash_intf translates the SPI flash command in to serial form that is communicated to the SPI flash.

The types of transactions with the SPI flash controller can be categorized into four classes:

Byte/Halfword/Word read or write These read from (or write to) the SPI flash memory can only be initiated on WISHBONE Port S. The SPI flash controller will only initiate a new read from (or write to) SPI flash memory when it determines that the SPI flash is not busy processing another command. The SPI flash memory address is obtained from the lower 24 bits of the WISHBONE address. On a write to the SPI flash memory, the SPI flash controller obtains the write data from the WISHBONE Port S's input data bus. It asserts S_ACK_O as soon as the write command is transmitted to the SPI flash (i.e., the SPI flash controller does not wait for the write to complete in the SPI flash). On a read from the SPI flash memory, the SPI flash controller asserts S_ACK_O only when data is returned from the SPI flash. It then puts this data on the WISHBONE Port S's output data bus.

Note

The SPI flash is a non-volatile memory and therefore it is not possible to overwrite a location that has been written to previously without erasing it. In order to erase memory, the user must issue an erase command via WISHBONE Port C.

Erase, Write enable/disable, Status read/write, Power up/down These SPI flash commands can only be initiated on WISHBONE Port C. Each of these commands is assigned an address within the memory map shown in Table 3 on page 7. To initiate a particular SPI flash command, the user must perform a WISHBONE read or write to the command's assigned address in the memory map. If the command does not involve any return data from the SPI flash, the SPI flash controller core asserts C_ACK_O as soon as the command is issued to the SPI flash.

SPI Flash Controller configuration commands These commands can only be initiated on WISHBONE Port C. They are used to perform the following tasks:

- Configure the SPI flash command instruction set so that the core can be used with a variety of SPI flash vendors.
- Control whether the SPI flash reads will be performed as "fast reads" or "slow reads."
- Control the SPI flash page and sector sizes.

SPI flash page read/program The process of reading or writing an entire page via WISHBONE Port S can be slow, because each page read/write needs to be split up in to multiple WISHBONE read/writes. The user can speed up the SPI flash page read/program by using the page read/program buffers. These buffers can only be accessed via WISHBONE Port C (see Memory Map in Table 3).

Configuration

The following sections describe the graphical user interface (UI) parameters and the I/O ports that you can use to configure and operate the LatticeMico SPI flash controller.

UI Parameters

Table 1 shows the user interface parameters available for configuring the LatticeMico SPI flash ROM controller through the Mico System Builder (MSB) interface.

Table 1: SPI Flash Controller UI Parameters

Dialog Box Option	Description	Allowable Values	Default Value
Instance Name	Specifies the name of the SPI flash controller instance.	Alphanumeric and underscores	SPIFlash
Memory Base Address	Specifies the base address for the Data Port of the SPI flash controller instance. The minimum boundary alignment is 0x4.	0x00000000 – 0xFFFFFFFF	16777216
Control Port Settings			
Control Port	Enable WISHBONE slave port that can perform all SPI flash commands such as chip/ sector erase, write enable/disable, read/write status register, power up/down, etc.	1, 0	0
Page Program Buffer	Enable Page Program Buffer to speed up page program by locally storing a page before writing entire page to SPI flash. Size of buffer is equal to page size.	1, 0	0

Table 1: SPI Flash Controller UI Parameters (Continued)

Dialog Box Option	Description	Allowable Values	Default Value
Page Read Buffer	Enable Page Read Buffer to speed up page reads by reading an entire page from SPI flash and locally storing it. Size of buffer is equal to page size.	1, 0	0
Control Base Address	Specifies the base address for the Control Port of SPI flash controller instance. The minimum boundary alignment is 0x800.	0x80000000 – 0xFFFFF800	0x80000000
SPI Flash Settings			
Page Size	Specifies the size of each page in the SPI flash. Refer to the SPI flash data sheet to obtain this value.		256
Sector Size	Specifies the size of each sector in the SPI flash. Refer to the SPI flash data sheet to obtain this value.		32768
SCLK Rate	Specifies the factor for deriving SCLK from the component input clock (processor clock CLK_I). SCLK is derived from the following equation: SCLK = CLK_I/(2 x SCLK_Rate) For example: For SCLK Rate = 0, SCLK = CLK_I For SCLK Rate = 1, SCLK = CLK_1/2	0 – 15	0
SPI Flash Command Opcodes Specify the one-byte instruction opcode for each command within the SPI flash instruction set. Refer to the SPI flash data sheet to obtain these values.			
Slow Read			3
Fast Read			11
Byte Program			2
Page Program			2
Block Erase Type 1			32
Block Erase Type 2			82
Block Erase Type 3			216
Chip Erase			96
Write Enable			6
Write Disable			4
Read Status Register			5
Write Status Register			1
Deep Power Down			185
Resume from Power Down			171

Table 1: SPI Flash Controller UI Parameters (Continued)

Dialog Box Option	Description	Allowable Values	Default Value
Read Manufacturer ID			159
WISHBONE Data Bus Size			
Data Port	Size of WISHBONE data bus on the data (S) port	8, 32	32
Control Port	Size of WISHBONE data bus on the control (C) port	8, 32	32

I/O Ports

Table 2 describes the input and output ports of LatticeMico SPI flash controller.

Table 2: LatticeMico SPI Flash I/O Port Descriptions

I/O Port	Active	Direction	Initial State	Description
RST_I	HIGH	I	1	
CLK_I		I		
Data WISHBONE Port				
S_ADR_I [31:0]		I	0	Address from WISHBONE interface
S_DAT_I [31:0] or S_DAT_I [7:0]		I	0	Data input from WISHBONE interface. The bus can be configured to be 8 or 32 bits. The bus holds valid data when S_WE_I is 1. The bus is big-endian (byte 0 is on S_DAT_I[31:24]). S_SEL_I indicates which byte(s) have valid data.
S_WE_I		I	0	Write Enable signal indicates whether the current WISHBONE bus cycle will perform a read or a write (0 = read, 1 = write).
S_STB_I		I	0	Strobe signal indicates that valid data and address are being presented on S_DAT_I and S_ADR_I busses respectively.
S_CYC_I		I	0	Cycle signal indicates that a WISHBONE bus cycle is in progress.
S_SEL_I [3:0] or S_SEL_I [0:0]		I	0	Select input array signal indicating where the valid data should be expected on S_DAT_I and S_DAT_O busses. This bus is 4 bits when S_DAT_I is 32 bits; otherwise, it is 1 bit wide. Each bit in S_SEL_I corresponds to a byte in the two data busses. Similar to S_DAT_I, this bus is big-endian (byte 0 is encoded on S_SEL_I [3]).
S_CTI_I [2:0]		I	0	Cycle Type signal indicating which type of WISHBONE bus cycle is in progress. The only WISHBONE bus cycles that are permitted on this port are Classic Cycles (000 and 111).

Table 2: LatticeMico SPI Flash I/O Port Descriptions (Continued)

I/O Port	Active	Direction	Initial State	Description
S_BTE_I [1:0]		I	0	Unused
S_LOCK_I		i	0	Unused
S_DAT_O [31:0] or S_DAT_O [7:0]		O	X	Data output from WISHBONE interface. This bus can be configured to be 8 or 32 bits. This bus holds valid data when S_WE_I is 0. The bus is big-endian (byte 0 is on S_DAT_0 [31:24]). S_SEL_I indicates which byte(s) have valid data.
S_ACK_O		O	X	Acknowledge output indicates that the current WISHBONE bus cycle is normally terminated.
S_ERR_O		O	X	Unused
S_RTY_O		O	X	Unused
Control WISHBONE Port				
C_ADR_I [31:0]		I	O	Address from WISHBONE interface
C_DAT_I [31:0] or C_DAT_I [7:0]		I	O	Data input from WISHBONE interface. This bus can be configured to be 8 or 32 bits. The bus holds valid data when C_WE_I is 1. The bus is big-endian (byte 0 is on C_DAT_I [31:24]). C_SEL_I indicates which byte(s) have valid data.
C_WE_I		I	O	Write Enable signal indicates whether the current WISHBONE bus cycle will perform a read or a write (0 - read, 1 = write).
C_STB_I		I	O	Strobe signal indicates that valid data and address are being presented on C_DAT_I and C_ADR_I busses respectively.
C_CYC_I		I	O	Cycle signal indicates that a WISHBONE bus cycle is in progress.
C_SEL_I [3:0] or C_SEL_I [0:0]		I	O	Select input array signal indicating where the valid data should be expected on C_DAT_I and C_DAT_O busses. This bus is 4 bits when C_DAT_I is 32 bits; otherwise, it is 1 bit wide. Each bit in C_SEL_I corresponds to a byte in the two data busses. Similar to C_DAT_I, this bus is big-endian (byte 0 is encoded on C_SEL_I[3]).
C_CTI_I [2:0]		I	O	Cycle Type signal indicating which type of WISHBONE bus cycle is in progress. The only WISHBONE bus cycles that are permitted on this port are Classic Cycles (000 and 111) and Burst (010).
C_BTE_I [1:0]		i	O	Burst Type signal indicating type of burst. Permitted values are 00 (sequential incrementing address burst).
C_LOCK_I		I	O	Unused

Table 2: LatticeMico SPI Flash I/O Port Descriptions (Continued)

I/O Port	Active	Direction	Initial State	Description
C_DAT_O [31:0] or C_DAT_O [7:0]		O	X	Data output from WISHBONE interface. This bus can be configured to be 8 or 32 bits. This bus holds valid data when C_WE_I is 0 and C_ACK_O is 1. The bus is big-endian (byte 0 is on C_DAT_O[31:24]). C_SEL_I indicates which byte(s) have valid data.
C_ACK_O		O	X	Acknowledge output indicates that the current data transfer of an ongoing WISHBONE bus cycle is normally terminated. When C_CTI_I is 000 or 111, the WISHBONE bus cycle is terminated. When C_CTI_I is 010, the next data transfer in the ongoing WISHBONE burst cycle is commenced.
C_ERR_O		O	X	Unused
C_RTY_O		O	X	Unused
SPI Interface				
SI		O	0	Serial data from SPI flash controller (FPGA) to SPI flash
SO		I	X	Serial data to SPI flash controller (FPGA) FROM SPI flash
CS	LOW	I	1	SPI flash Chip Select
SCK		I	0	Serial clock to SPI flash
WP	LOW	I	1	SPI flash Write Protect

Register Descriptions

The LatticeMico SPI flash controller includes the registers shown in Table 3. The user must enable the Control WISHBONE Port in order to access these registers. Table 4 through Table 25 describe the individual registers in detail.

Table 3: Register Map

Register Name	Offset	31-24	23-16	15-8	7	6	5	4	3	2	1	0
Registers to issue SPI flash commands												
Page Program	0x000	Page Program Address										
		Byte 0 (LSB)	Byte 1	Byte 2 (MSB)	Reserved							Start
Page Read	0x004	Page Read Address										
		Byte 0 (LSB)	Byte 1	Byte 2 (MSB)	Reserved							Start
Block Erase Type 1	0x008	Erase Address										
		Byte 0 (LSB)	Byte 1	Byte 2 (MSB)	Reserved							Start

Table 3: Register Map (Continued)

Register Name	Offset	31-24	23-16	15-8	7	6	5	4	3	2	1	0
Block Erase Type 2	0x00C	Erase Address										
		Byte 0 (LSB)	Byte 1	Byte 2 (MSB)	Reserved							Start
Block Erase Type 3	0x010	Erase Address										
		Byte 0 (LSB)	Byte 1	Byte 2 (MSB)	Reserved							Start
Chip Erase	0x014	Reserved										Start
Write Enable	0x018	Reserved										Enable
Write Disable	0x01c	Reserved										Disable
Status Read	0x020	Reserved			SPI flash status register (Refer to SPI flash data sheet.)							
Status Write	0x024	Reserved			SPI flash status register (Refer to SPI flash data sheet.)							
Power Down	0x028	Reserved										Start
Power Up	0x02c	Reserved										Start
Manufacture ID	0x030	Reserved			SPI flash manufacturer's device ID (Refer to SPI flash data sheet.)							
Registers to issue a user-defined command to SPI flash												
USER CMD 0	0x034 to 0x040	Least-significant word of command (Byte 0 is the first byte sent to SPI flash.)										
		Byte 0 (LSB)	Byte 1	Byte 2 (MSB)	Byte 3							
USER CMD 1	0x038 to 0x044	Most-significant word of command (Byte 3 is the last byte sent to SPI flash.)										
		Byte 0 (LSB)	Byte 1	Byte 2 (MSB)	Byte 3							
User CMD Length	0x03C to 0x048											Length (in bytes)
User Return Length	0x040 to 0x04C											Length (in bytes)
User Return Data	0x044 to 0x050	Byte 0 (LSB)	Byte 1	Byte 2 (MSB)	Byte 3 (last byte received from SPI flash)							
User CMD	0x048 to 0x054	Reserved										Start

Table 3: Register Map (Continued)

Register Name	Offset	31-24	23-16	15-8	7	6	5	4	3	2	1	0
Registers to configure SPI flash controller internals												
Read Speed	0x100	Reserved										Fast
Page Program Size	0x104	Size (in bytes)										
		Reserved		Byte 0 (LSB)	Byte 1 (MSB)							
Page Read Size	0x108	Size (in bytes)										
		Reserved		Byte 0 (LSB)	Byte 1 (MSB)							
Registers to configure SPI flash controller for a SPI flash's instruction set												
Slow Read CFG	0x180	Reserved			Opcode							
Fast Read CFG	0x184	Reserved			Opcode							
Byte Program CFG	0x188	Reserved			Opcode							
Page Program CFG	0x18c	Reserved			Opcode							
Block Erase 1 CFG	0x190	Reserved			Opcode							
Block Erase 2 CFG	0x194	Reserved			Opcode							
Block Erase 3 CFG	0x198	Reserved			Opcode							
Chip Erase CFG	0x19c	Reserved			Opcode							
Write Enable CFG	0x1a0	Reserved			Opcode							
Write Disable CFG	0x1a4	Reserved			Opcode							
Read Status CFG	0x1a8	Reserved			Opcode							
Write Status CFG	0x1ac	Reserved			Opcode							
Power Down CFG	0x1b0	Reserved			Opcode							
Power Up CFG	0x1b4	Reserved			Opcode							
Read ID CFG	0x1b8	Reserved			Opcode							
Page Program Buffer	0x200-0x3FF	Byte 0 (LSB)	Byte 1	Byte 2	Byte 3 (MSB)							
Page Read Buffer	0x400-0x6FF	Byte 0 (LSB)	Byte 1	Byte 2	Byte 3 (MSB)							

Table 4: Page Program Register (ADDR = 0x000)

Name	Bit(s)	Access Mode	Description
Address	31-8	Read/Write	24-bit Page Address. Bits 31-24 contain least-significant byte of address and bits 15-8 contain most-significant byte.
Start	0	Write	Writing a 1 to this bit will commence a page program operation.

Table 5: Page Read Register (Addr = 0x004)

Name	Bit(s)	Access Mode	Description
Address	31-8	Read/Write	24-bit Page Address. Bits 31-24 contain least-significant byte of address and bits 15-8 contain most-significant byte.
Start	0	Write	Writing a 1 to this bit will commence a page read operation.

Table 6: Block Erase Type 1 Register (Addr = 0x00c)

Name	Bit(s)	Access Mode	Description
Address	31-8	Read/Write	24-bit Page Address. Bits 31-24 contain least-significant byte of address and bits 15-8 contain most-significant byte.
Start	0	Write	Writing a 1 to this bit will commence the 4K block erase operation.

Table 7: Block Erase Type 2 Register (Addr = 0x010)

Name	Bit(s)	Access Mode	Description
Address	31-8	Read/Write	24-bit Page Address. Bits 31-24 contain least-significant byte of address and bits 15-8 contain most-significant byte.
Start	0	Write	Writing a 1 to this bit will commence the 32K block erase operation.

Table 8: Block Erase Type 3 Register (Addr = 0x014)

Name	Bit(s)	Access Mode	Description
Address	31-8	Read/Write	24-bit Page Address. Bits 31-24 contain least-significant byte of address and bits 15-8 contain most-significant byte.
Start	0	Write	Writing a 1 to this bit will commence the 64K block erase operation.

Table 9: Chip Erase Register (Addr = 0x018)

Name	Bit(s)	Access Mode	Description
Start	0	Write-only	Write a 1 or 0 to this location to initiate a SPI flash chip erase command.

Table 10: Write Enable Register (Addr = 0x018)

Name	Bit(s)	Access Mode	Description
Enable	0	Write-only	Write 1 to this location to initiate a SPI flash “Write Enable” command.

Table 11: Write Disable Register (Addr = 0x01c)

Name	Bit(s)	Access Mode	Description
Disable	0	Write-only	Write 1 to this location to initiate a SPI flash “Write Disable” command.

Table 12: Status Read Register (Addr = 0x020)

Name	Bit(s)	Access Mode	Description
Value	7:0	Read-only	Read the Status Register in SPI flash.

Table 13: Status Write Register (Addr = 0x024)

Name	Bit(s)	Access Mode	Description
Value	7:0	Write-only	Write to the Status Register in SPI flash.

Table 14: Power Down Register (Addr = 0x028)

Name	Bit(s)	Access Mode	Description
Start	0	Write-only	Initiate a “Deep Power Down” command in SPI flash by writing a 1 or 0.

Table 15: Power Up Register (Addr = 0x02c)

Name	Bit(s)	Access Mode	Description
Start	0	Write-only	Initiate a “Resume from Power Down” command in SPI flash by writing a 1 or 0.

Table 16: Manufacturer ID Register (Addr = 0x030)

Name	Bit(s)	Access Mode	Description
Value	7:0	Read-only	Read the Manufacturer ID in the SPI flash.

Table 17: User CMD 0 Register (Addr = 0x034)

Name	Bit(s)	Access Mode	Description
LSW	31:0	Read/Write	The least-significant word of a user-defined command that will be issued to SPI flash.

Table 18: User CMD 1 Register (Addr = 0x038)

Name	Bit(s)	Access Mode	Description
MSW	31:0	Read/Write	The most-significant word of a user-defined command that will be issued to SPI flash.

Table 19: User CMD Length Register (Addr = 0x03c)

Name	Bit(s)	Access Mode	Description
Length	2:0	Read/Write	The number of bytes that constitute a user-defined command (command is written to registers User CFG 0 and User CFG 1). The number of bytes is represented using the formula: Total Bytes = Length + 1 The minimum value is 1 and maximum value is 8.

Table 20: User Return Length Register (Addr = 0x040)

Name	Bit(s)	Access Mode	Description
Length	2:0	Read/Write	The number of bytes returned by the SPI flash on a user-defined command. The return data is available in the User Return Data register. The number of bytes is represented using the formula: Total Bytes = Length The minimum value is 0 and maximum value is 1.

Table 21: User Return Data Register (Addr = 0x044)

Name	Bit(s)	Access Mode	Description
Return Data	31:0	Read-only	The data that is returned by SPI flash on a user-defined command. The total bytes is determined by value in the User Return Length register.

Table 22: User CMD Register (Addr = 0x048)

Name	Bit(s)	Access Mode	Description
Start	0	Write-only	Issue a user-defined command to SPI flash. The user must make sure that the following registers have valid contents prior to writing to this register: ▶ User CMD 0 ▶ User CMD 1 ▶ User CMD Length ▶ User Return Length The user can read the User Return Data register after a write to this register to get the returned data from SPI flash.

Table 23: Read Speed Register (Addr = 0x100)

Name	Bit(s)	Access Mode	Description
Fast	0	Write-only	Configure the SPI flash controller to perform fast or slow reads. 0 – Slow reads 1 – Fast reads

Table 24: Page Program Size REgister (Addr = 0x104)

Name	Bit(s)	Access Mode	Description
Size	15:0	Read/Write	The number of bytes to be written to SPI flash on a Page Program. The minimum value is 0 and maximum value is PAGE_SIZE. The SPI flash controller will write the number of bytes specified in this field from the Page Program Buffer to the SPI flash.

Table 25: Page Read Size Register (0x108)

Name	Bit(s)	Access Mode	Description
Size	15:0	Read/Write	The number of bytes to be read from SPI flash on a Page Read. The minimum value is 0 and maximum value is PAGE_SIZE. The SPI flash controller will read the number of bytes specified in this field from SPI flash in to the Page Read Buffer.

EBR Utilization

This component does not use any EBRs.

LatticeMico32 Microprocessor Software Support

This section describes the LatticeMico32 microprocessor software support provided for the LatticeMico SPI flash controller. Please note that the supporting routines are meant for use in a single-threaded environment. If you want to use them in a multi-tasking environment, you must provide re-entrance protections.

Device Driver

The SPI flash driver interacts directly with the SPI flash controller and, in turn, the SPI flash device. This section describes the type definitions, structure, and functions of the SPI flash device driver.

Context Structure

This section describes the type definitions for the SPI flash controller context structure. It is shown in Figure 1, which contains SPI flash controller-specific information that is dynamically generated in the DDStructs.h header file. This information is largely filled in by the MSB managed build process, which extracts SPI flash controller-specific information from the platform definition file. The members should not be manipulated directly since this structure is for exclusive use by the device driver. Table 26 on page 14 describes the SPI flash controller context structure shown in Figure 1.

Figure 1: Type Definitions for SPI Flash Controller Context Structure

```
typedef struct st_MicoSPIFlashCtx_t {
    const char *name;
    unsigned int memory_base;
    unsigned int memory_size; unsigned int memory_wbSize;
    unsigned int control_base;
    unsigned int control_port;
    unsigned int control_wbSize;
    unsigned int program_buf_en;
    unsigned int read_buf_en;
    unsigned int page_size;
    unsigned int sector_size;
    DeviceReg_t lookupReg;
    void *prev;
    void *next;
} MicoSPIFlashCtx_t;
```

Table 26: LatticeMico SPI Flash Controller Context Structure Parameters

Parameter	Data Type	Description
name	const char *	Pointer to the SPI flash controller's instance name
memory_base	unsigned int	Base address of the Data WISHBONE Port (memory)

Table 26: LatticeMico SPI Flash Controller Context Structure Parameters

Parameter	Data Type	Description
memory_size	unsigned int	Size (in bytes) of the memory
memory_wbSize	unsigned int	Indicates whether the Data WISHBONE Port has an 8-bit data bus or a 32-bit data bus.
control_base	unsigned int	Base address of the Control WISHBONE port
control_port	unsigned int	Indicates whether the Control WISHBONE Port is enabled in hardware and supports the entire SPI flash instruction set.
control_wbSize	unsigned int	Indicates whether the Control WISHBONE port has an 8-bit data bus or a 32-bit data bus.
program_buf_en	unsigned int	Indicates whether the Page Program Buffer is enabled in hardware. Used by the device driver to decide whether page programming is to be done via the faster method of using the Page Program Buffer to do bulk writes or via the slower method of performing individual byte writes to the memory.
read_buf_en	unsigned int	Indicates whether the Page Read Buffer is enabled in hardware. Used by the device driver to decide whether page read is to be done via the faster method of initiating bulk reads to the flash and storing in the Page Read Buffer or via the slower method of performing individual byte reads from the memory.
page_size	unsigned int	Indicates the size of each page in the SPI flash.
sector_size	unsigned int	Indicates the size of each sector in the SPI flash.
lookup_reg	DeviceReg_t	Used by the device driver to register the SPI flash controller component instance with the LatticeMico32 lookup service. Refer to the *LatticeMico32 Software Developer User Guide* for a description of the DeviceReg_t data type.
prev	void *	Used by the device driver service to keep track of registered SPI instances
next	void *	Used by the device driver service to keep track of registered SPI instances

Functions

This section describes the implemented device driver-specific functions.

MicoSPIFlash_Initialize Function

```
void MicoSPIFlash_Initialize (MicoSPIFlashCtx_t *ctx);
```

This function initializes a LatticeMico SPI flash controller device instance. It is automatically called as part of platform initialization for managed builds for each instance of the SPI flash controller. It sets the controller to a known

stopped state for future use and registers this SPI flash controller instance for the device lookup service.

Table 27: MicoSPIFlashCtx_t Parameter

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.

MicoSPIFlash_ReadID Function

```
int MicoSPIFlash_ReadID (MicoSPIFlashCtx_t *ctx);
```

This function reads the Manufacturer ID from the SPI flash device to which this SPI flash controller instance is connected.

Table 28: MicoSPIFlash_ReadID Parameter

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance

Table 29: MicoSPIFlashReadID Return Values

Return Value	Description
0x00 to 0xFF	SPI flash manufacturer ID

MicoSPIFlash_StatusRead Function

```
int MicoSPIFlash_StatusRead (MicoSPIFlashCtx_t *ctx);
```

This function reads the contents of the Status Register from the SPI flash device to which this SPI flash controller instance is connected. Since SPI flash manufacturers define the contents of this status register in a manufacturer-specific format, it is the responsibility of the software developer to refer to the SPI flash data sheet to interpret the returned data.

Table 30: MicoSPIFlash_StatusRead Parameter

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.

Table 31: MicoSPIFlash_StatusRead Return Values

Return Value	Description
0 – 255	SPI flash status register
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.

MicoSPIFlash_WriteEnable Function

```
int MicoSPIFlash_WriteEnable (MicoSPIFlashCtx_t *ctx);
```

This function issues a Write Enable command to the SPI flash device that is connected to the SPI flash controller instance. This function must be invoked before erasing or before programming the SPI flash.

Table 32: MicoSPIFlash_WriteEnable Parameter

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.

Table 33: MicoSPIFlash_WriteEnable Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The “Write Enable” command was successfully sent to SPI flash.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.

MicoSPIFlash_WriteDisable Function

```
int MicoSPIFlash_WriteDisable (MicoSPIFlashCtx_t *ctx);
```

This function issues a Write Disable command to the SPI flash device that is connected to the SPI flash controller instance. It sets up the SPI flash to disallow any erase or programming operation that might be initiated in the future.

Table 34: MicoSPIFlash_WriteDisable Parameter

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.

Table 35: MicoSPIFlash_WriteDisable Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The “Write Disable” command was successfully sent to SPI flash.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.

MicoSPIFlash_ChipErase Function

```
int MicoSPIFlash_ChipErase (MicoSPIFlashCtx_t *ctx);
```

This function erases the SPI flash device that is connected to the SPI flash controller instance. This function requests that the SPI flash perform a chip erase, and then it exits. It does not wait for the SPI flash to actually complete erasing the chip. This allows the software execution to continue even though the SPI flash is still in the process of erasing all data.

Note

Since it is possible that the SPI flash controller might receive a new SPI flash request from any of the two WISHBONE ports while an erase is in progress, the SPI flash controller hardware is designed to issue a new command to the SPI flash only when the SPI flash indicates it is not busy.

Table 36: MicoSPIFlash_ChipErase Parameter

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.

Table 37: MicoSPIFlash_ChipErase Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The “Chip Erase” command was successfully sent to SPI flash.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.

MicoSPIFlash_BlockErase Function

```
int MicoSPIFlash_BlockErase (MicoSPIFlashCtx_t *ctx, unsigned
int anAddress unsigned int aType);
```

This function issues a command to the SPI flash device that is connected to the SPI flash controller instance to erase the block (also known as section) at the specified address. This function can issue one of the three types of block erases that are specified by the SPI flash command instructions defined in Block Erase 1 CFG, Block Erase 2 CFG, and Block Erase 3 CFG registers shown in Table 3 on page 7.

This function requests that the SPI flash perform a block erase, and then it exits. It does not wait for the SPI flash to actually complete erasing the block. This allows the software execution to continue even though the SPI flash is still in the process of erasing all data in specified block.

Note

Since it is possible that the SPI flash controller might receive a new SPI flash request from any of the two WISHBONE ports while an erase is in progress, the SPI flash controller hardware is designed to issue a new command to the SPI flash only when the SPI flash indicates it is not busy.

Table 38: MicoSPIFlash_BlockErase Parameters

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.
unsigned int anAddress	Address of SPI flash sector/block to be erased
unsigned int aType	Indicates the type of SPI flash block erase command to be used. This, in turn, also determines the size of block that is being erased. The three legal values are: 1 – Type 1 for block size of 4K 2 – Type 2 for block size of 32K 3 – Type 3 for block size of 64K

Table 39: MicoSPIFlash_BlockErase Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The “Block Erase” command was successfully sent to SPI flash.
SPI_COMMAND_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.
SPI_INVALID_BLOCK_ERASE_TYPE	Failure. The SPI flash block erase type is not one of the three legal values.
SPI_ADDRESS_OUT_OF_RANGE	Failure. The address is invalid and does not point to any block within the SPI flash.

MicoSPIFlash_PageProgram Function

```
int MicoSPIFlash_PageProgram (MicoSPIFlashCtx_t *ctx, unsigned
int anAddress, unsigned int aLength, char *rData);
```

This function is used to perform bulk writes to SPI flash. There are two ways that bulk writes to SPI flash can be performed, depending on whether the Page Program Buffer is enabled in hardware. When the Page Program Buffer is enabled, the function performs bulk writes by first writing the data to the SPI flash controller's Page Program Buffer and then sending this data to the SPI flash in one command. When the Page Program Buffer is disabled in hardware, the function performs these writes via the significantly slower method of writing the given data to SPI flash one byte per SPI flash command.

The function will only write to a single page, and it can start at any address within the page. The software developer must note that this function will stop writing data to SPI flash as soon as it reaches the end of a page.

Table 40: MicoSPIFlash_Page Program Parameters

Parameter	Description
MicoSPIFlashCtx_t*ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.
unsigned int anAddress	The first location in SPI flash memory where program will commence. It can be any location within the page.
unsigned int aLength	The total number of bytes to be written to SPI flash. The hardware will terminate the writes at the end of a page even if there are additional bytes to be written.
char *rData	The array of bytes to be written to SPI flash. The software developer must ensure that the byte array contains the number of bytes specified in the aLength argument. Otherwise, the program can unexpectedly crash.

Table 41: MicoSPIFlash_Page Program Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The "Page Program" command was successfully completed.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.
SPI_ADDRESS_OUT_OF_RANGE	Failure. The address is invalid and does not point to any block within the SPI flash.

MicoSPIFlash_PageRead Function

```
int MicoSPIFlash_PageRead (MicoSPIFlashCtx_t *ctx, unsigned int
anAddress, unsigned int aLength char *rData);
```

This function is used to perform bulk reads from SPI flash. There are two ways that bulk reads from SPI flash can be performed, depending on whether the Page Read Buffer is enabled in hardware. When the Page Read Buffer is enabled, the function reads an entire page in a single SPI flash command and stores it to the SPI flash controller's Page Read Buffer before returning it to the caller software code. When the Page Read Buffer is disabled in hardware, the function performs these reads via a significantly slower method of reading data from SPI flash one byte per SPI flash command. The maximum number of bytes the function can read is limited to the number of bytes in a page.

Table 42: MicoSPIFlash_PageRead Parameters

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.
unsigned int anAddress	The first location in SPI flash memory where read will commence. It can be any location within the page.
unsigned int aLength	The total number of bytes to be read from SPI flash. The hardware will terminate the reads at the end of a page even if there are additional bytes to be read.
char *rData	The data from SPI flash is returned in this array. The function will not perform malloc. The task of allocating space for aLength bytes in this array is left to the software that calls this function.

Table 43: MicoSPIFlash_PageRead Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The "Page Read" command was successfully completed.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.
SPI_ADDRESS_OUT_OF_RANGE	Failure. The address is invalid and does not point to any block within the SPI flash.

MicoSPIFlash_AlignedPageProgram Function

```
int MicoSPIFlash_AlignedPageProgram (MicoSPIFlashCtx_t *ctx,
unsigned int anAddress, unsigned int aLength, unsigned int
*rData);
```

This function is used to perform bulk writes to SPI flash. The difference between this function and MicoSPIFlash_PageProgram function is that this function is optimized for the scenario in which the address is word-aligned and the number of bytes to be written is a multiple of 4. There are two ways that bulk writes to SPI flash can be performed, depending on whether the Page Program Buffer is enabled in hardware. When the Page Program Buffer is enabled, the function performs bulk writes by first writing the data to the SPI flash controller's Page Program Buffer and then sending this data to the SPI flash in one command. When the Page Program Buffer is disabled in hardware, the function performs these writes via a significantly slower method of writing the given data to SPI flash one byte per SPI flash command.

The function will only write to a single page and it can start at any address within the page. The software developer must note that this function will stop writing data to SPI flash as soon as it reaches the end of a page.

Table 44: MicoSPIFlash_AlignedPageProgram Parameters

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.
unsigned int anAddress	The first location in SPI flash where program will commence. This location can be any address within the page as long as it is word-aligned.
unsigned int aLength	The total number of bytes to be written to SPI flash. The hardware will terminate the writes at the end of a page even if there are additional bytes to be written.
integer *rData	The array of integers to be written to SPI flash. The software developer must ensure that the byte array contains the number of bytes specified in the aLength argument. Otherwise, the program can unexpectedly crash.

Table 45: MicoSPIFlash_AlignedPageProgram Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The "Page Program" command was successfully completed.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.
SPI_ADDRESS_OUT_OF_RANGE	Failure. The address is invalid and does not point to any block within the SPI flash.

MicoSPIFlash_AlignedPageRead Function

```
int MicoSPIFlash_AlignedPageRead (MicoSPIFlashCtx_t *ctx,
unsigned int anAddress, unsigned int aLength, unsigned int
*rData);
```

This function is used to perform bulk reads from SPI flash. The difference between this function and the MicoSPIFlash_PageRead function is that this function is optimized for the scenario in which the address is word-aligned and the number of bytes to be read is a multiple of 4. There are two ways that bulk reads from SPI flash can be performed, depending on whether the Page Read Buffer is enabled in hardware. When the Page Read Buffer is enabled, the function reads an entire page in a single SPI flash command and stores it to the SPI flash controller's Page Read Buffer before returning it to the caller software code. When the Page Read Buffer is disabled in hardware, the function performs these reads via a significantly slower method of reading data from SPI flash one byte per SPI flash command. The maximum number of bytes the function can read is limited to the number of bytes in a page.

Table 46: MicoSPIFlash_AlignedPageRead Parameters

Parameter	'Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.
unsigned int anAddress	The first location in SPI flash memory where read will commence. It can be any location within the page as long as it is aligned to a word boundary.
unsigned int aLength	The total number of bytes to be read from SPI flash. The number must be a multiple of 4. The software developer must note that the hardware will terminate the reads at the end of a page even if there are additional bytes to be read.
integer *rData	The data from SPI flash is returned in this array. The function will not perform a malloc. The task of allocating space for (aLength/4) integers in this array is left to the software that calls this function.

Table 47: MicoSPIFlash_AlignedPageRead Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The "Page Read" command was successfully completed.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.
SPI_ADDRESS_OUT_OF_RANGE	Failure. The address is invalid and does not point to any block within the SPI flash.

MicoSPIFlash_ReadSpeed Function

```
int MicoSPIFlash_ReadSpeed (MicoSPIFlashCtx_t *ctx, unsigned
int aSpeed);
```

This function is used to configure the SPI flash controller instance so that it performs all read requests to SPI flash in either the slow or fast read mode. The slow and fast read instruction opcodes are defined in the Slow Read CFG or Fast Read CFG of the register map shown in Table 3 on page 7. The software developer must ensure that these opcodes match the SPI flash data sheet.

Table 48: MicoSPIFlash_ReadSpeed Parameters

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.
unsigned int aSpeed	0 – Slow reads 1 – Fast reads

Table 49: MicoSPIFlash_ReadSpeed Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The “Page Read” command was successfully completed.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.

MicoSPIFlash_ReadCmdOpcodes Function

```
int MicoSPIFlash_ReadCmdOpcode (MicoSPIFlashCtx_t *ctx,
MicoSPIFlashCmdOpcodeTable_t *opcode);
```

This function provides the SPI flash instruction set for which the SPI flash controller instance has been configured.

Table 50: MicoSPIFlash_ReadCmdOpcodes Parameters

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.
MicoSPIFlashCmdOpcodeTable_t *opcode	Pointer to a valid MicoSPIFlashCmdOpcodeTable_t structure representing a list of the SPI flash instruction opcodes. The function will not perform malloc. It is the responsibility of the software calling this function to allocate memory for this structure.

Table 51: MicoSPIFlash_ReadCmdOpcodes Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The “Page Read” command was successfully completed.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.

MicoSPIFlash_WriteCmdOpcodes Function

```
int MicoSPIFlash_WriteCmdOpcode (MicoSPIFlasCtx_t *ctx,
MicoSPIFlashCmdOpcodeTable_t *opcode);
```

This function provides the software developer with a mechanism to change the SPI flash instruction set for the SPI flash controller instance.

Table 52: MicoSPIFlash_WriteCmdOpcodes Parameters

Parameter	Description
MicoSPIFlashCtx_t *ctx	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash instance.
MicoSPIFlashCmdOpcodeTable_t *opcode	Pointer to a valid MicoSPIFlashCMDOpcodeTable_t structure representing a list of the SPI flash instruction opcodes.

Table 53: MicoSPIFlash_WriteCmdOpcodes Return Values

Return Value	Description
SPI_COMMAND_SUCCESS	The “Page Read” command was successfully completed.
SPI_CONTROL_PORT_ERROR	Failure. SPI flash controller hardware will not accept commands on WISHBONE C Port.

Macros

This section describes the implemented user C macros that can be used by the software developer to directly access the SPI flash controller’s registers, shown in Table 3 on page 7. There are two sets of macros, one for when the Control WISHBONE port’s data bus is 8 bits wide and another for when the data bus is 32 bits wide. These macros can be found in the file MicoSPIFlash.h.

Macros have either one or two input arguments. The first input argument "X" is always the base address of the SPI flash controller, as assigned by MSB. The second input argument "Y" is optional, depending on the macro’s function. Table 54 lists all the macros for a 32 bit data bus on the Control WISHBONE port. Table 55 on page 33 lists all the macros for an 8 bit data

bus on the Control WISHBONE port.

Table 54: Macros to be used for a 32-bit Data Bus on the WISHBONE Control Port

Macro Name	Second Argument to Macro	Description
MICO_SPI_PAGE_ PROGRAM	The 24-bit address in the SPI flash where data programming will commence.	This macro initiates a SPI flash "page program" by writing to the "Page Program" register. Usage: If SPI flash cotroller's base address is 0x80001000 and SPI flash address to be programmed is 0x100000, MICO_SPI_PAGE_PROGRAM (0x80001000, 0x100000); **NOTE** : The software developer must ensure that the Page Program Buffer is enabled in hardware and that it contains the data that needs to be programmed.
MICO_SPI_PAGE_READ	The 24-bit address in the SPI flash where data reading will commence.	This macro initiates a SPI flash "page read" by writing to the "Page Read" register. Usage: If SPI flash controller's base address is 0x80001000 and SPI flash address to be read from is 0x100000, MICO_SPI_PAGE_READ (0x80001000, 0x100000); **NOTE**: The software developer must ensure that the Page Read Buffer is enabled in hardware.
MICO_SPI_BLOCK_ ERASE_TYPE1	The 24-bit address of the sector in SPI flash that needs to be erased.	This macro initiates a SPI flash "sector erase" command to erase a 4Kbytes sector. Usage: If SPI flash controller's base a address is 0x80001000 and SPI flash sector address is 0x100000, MICO_SPI_BLOCK_ERASE_TYPE1 (0x80001000, 0x100000);
MICO_SPI_BLOCK_ ERASE_TYPE2	The 24-bit address of the sector in SPI flash that needs to be erased.	This macro initiates a SPI flash "sector erase" command to erase a 32Kbytes sector. Usage: If SPI flash controller's base a address is 0x80001000 and SPI flash sector address is 0x100000, MICO_SPI_BLOCK_ERASE_TYPE2 (0x80001000, 0x100000);
MICO_SPI_BLOCK_ ERASE_TYPE3	The 24-bit address of the sector in SPI flash that needs to be erased.	This macro initiates a SPI flash "sector erase" command to erase a 64Kbytes sector. Usage: If SPI flash controller's base a address is 0x80001000 and SPI flash sector address is 0x100000, MICO_SPI_BLOCK_ERASE_TYPE3 (0x80001000, 0x100000);

Table 54: Macros to be used for a 32-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CHIP_ERASE		Macros This macro initiates the SPI flash "chip erase" command to erase the entire chip. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_CHIP_ERASE (0x80001000);
MICO_SPI_WRITE_ ENABLE		This macro initiates the SPI flash "write enable" command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_WRITE_ENABLE (0x80001000);
MICO_SPI_WRITE_ DISABLE		This macro initiates the SPI flash "write disable" command. Usage: If SPI flash controller's base address is 0x800010000, MICO_SPI_WRITE_DISABLE (0x80001000);
MICO_SPI_STATUS_ READ	The status register read from the SPI flash.	This macro initiates the SPI flash "status register read" command. Usage: If SPI flash controller's base address is 0x80001000, unsigned int status; MICO_SPI_STATUS_READ (0x80001000, status);
MICO_SPI_STATUS_ WRITE	The value written to the status register in the SPI flash	This macro initiates the SPI flash "status register write" command. Usage: If SPI flash controller's base address is 0x80001000, and the value to be written to SPI flash's status register is 0x2, unsigned int status = 0x2; MICO_SPI_STATUS_WRITE (0x80001000, status);
MICO_SPI_POWER_ DOWN		This macro initiates the SPI flash "power down" command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_POWER_DOWN (0x80001000);
MICO_SPI_POWER_UP		This macro initiates the SPI flash "power up" command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_POWER_UP (0x80001000);

Table 54: Macros to be used for a 32-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_ MANUFACTURER_ID	The manufacturer ID read from the SPI flash	This macro initiates the SPI flash "read manufacturer ID" command. Usage: If SPI flash controller's base address is 0x80001000, unsigned int ID; MICO_SPI_MANUFACTURER_ID (0x80001000, ID);
MICO_SPI_CUSTOM_ LSW	The least-significant word of the user-defined command that is issued to the SPI flash. **NOTE**: The first byte to be transmitted to SPI flash is the most-significant byte of this word.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to provide the least-significant word of the user-defined command. Usage: If SPI flash controller's base address is 0x80001000, and the sequence of four bytes to be transmitted are 0x12, 0x34, 0x56, and then 0x78, MICO_SPI_CUSTOM_LSW (0x80001000, 0x12345678);
MICO_SPI_CUSTOM_ MSW	The most-significant word of the user-defined command that is issued to the SPI flash. **NOTE**: The first byte to be transmitted to SPI flash is the most-significant byte of this word.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to provide the most-significant word of the user-defined command. Usage: If SPI flash controller's base address is 0x80001000, and the sequence of four bytes to be transmitted are 0x12, 0x34, 0x56, and then 0x78, MICO_SPI_CUSTOM_MSW (0x80001000, 0x12345678); **NOTE**: This macro is optional since the SPI flash command could be four bytes or less.
MICO_SPI_CUSTOM_ LENGTH	The length, in bytes, of the user-defined command that is issued to the SPI flash.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to indicate the number of bytes transmitted as part of the user command. Usage: If SPI flash controller's base address is 0x80001000 and the number of bytes transmitted is 1, then MICO_SPI_CUSTOM_LENGTH (0x80001000, 0x1);

Table 54: Macros to be used for a 32-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CUSTOM_ RETURN_LENGTH	The length, in bytes, of the data returned by the SPI flash for the user-defined command that is issued.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to indicate the number of bytes returned from the SPI flash as part of the user command. The return length can be zero. Usage: If SPI flash controller's base address is 0x80001000 and the number of bytes returned is 4, then MICO_SPI_CUSTOM_RETURN_LENGTH (0x80001000, 0x4);
MICO_SPI_CUSTOM_ RETURN_DATA	The data returned by the SPI flash for the user-defined command. **NOTE**: The first byte returned from the SPI flash is the most-significant byte of this word.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to obtain the data returned by the SPI flash if return length is non-zero. Usage: If SPI flash controller's base address is 0x80001000 and the number of bytes returned is 4 and the sequence received from SPI flash is 0x12, 0x23, 0x56, then 0x78. unsigned int retValue; MICO_SPI_CUSTOM_RETURN_DATA (0x80001000, retValue); retValue will be equal to 0x12345678.
MICO_SPI_CUSTOM		This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to initiate transmission of user-defined command to SPI flash. **NOTE**: The software developer must make sure that the user-defined command, length, and return length are set up prior to use of this macro.

MICO_SPI_CFG_WR_ PAGE_PROGRAM_SIZE	The number of bytes to be programmed on a SPI flash "page program".	This macro is used to indicate the number of bytes to be programmed in to SPI flash on a SPI flash "page program" command.
		Usage:
		If SPI flash controller's base address is 0x80001000 and the number of bytes to be programmed is 256, then
		unsigned int size = 0x100;
		MICO_SPI_CFG_WR_PAGE_PROGRAM_SIZE (0x80001000, size);

Table 54: Macros to be used for a 32-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CFG_WR_ PAGE_READ_SIZE	The number of bytes to be read on a SPI flash "page read".	This macro is used to indicate the number of bytes to be read from the SPI flash on a SPI flash "page read" command.
		Usage:
		If SPI flash controller's base address is 0x80001000 and the number of bytes to be read is 256, then
		unsigned int size = 0x100;
		MICO_SPI_CFG_WR_PAGE_READ_SIZE (0x80001000, size);

MICO_SPI_CMD_CFG_ WR_SLOW_READ	The opcode for the SPI flash "slow read" command.	This macro initialized the SPI flash controller with the SPI flash's "slow read" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_SLOW_READ (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_ WR_FAST_READ	The opcode for the SPI flash "fast read" command.	This macro initialized the SPI flash controller with the SPI flash's "fast read" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_FAST_READ(0x80001000, 0xAB);
MICO_SPI_CMD_CFG_ WR_BYTE_PROGRAM	The opcode for the SPI flash "byte program" command.	This macro initialized the SPI flash controller with the SPI flash's "byte program" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ BYTE_PROGRAM (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_ WR_PAGE_PROGRAM	The opcode for the SPI flash "page program" command.	This macro initialized the SPI flash controller with the SPI flash's "page program" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ PAGE_PROGRAM (0x80001000, 0xAB);

Table 54: Macros to be used for a 32-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CMD_CFG_WR_BLOCK_ERASE_TYPE1	The opcode for the SPI flash "block erase type 1" command.	This macro initialized the SPI flash controller with the SPI flash's "block erase type 1" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE1 (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_BLOCK_ERASE_TYPE2	The opcode for the SPI flash "block erase type 2" command.	This macro initialized the SPI flash controller with the SPI flash's "block erase type 2" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE2 (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_BLOCK_ERASE_TYPE3	The opcode for the SPI flash "block erase type 3" command.	This macro initialized the SPI flash controller with the SPI flash's "block erase type 3" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE3 (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_CHIP_ERASE	The opcode for the SPI flash "chip erase" command.	This macro initialized the SPI flash controller with the SPI flash's "chip erase" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ CHIP_ERASE (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_WRITE_ENABLE	The opcode for the SPI flash "write enable" command.	This macro initialized the SPI flash controller with the SPI flash's "write enable" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ WRITE_ENABLE (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_WRITE_DISABLE	The opcode for the SPI flash "write disable" command.	This macro initialized the SPI flash controller with the SPI flash's "write disable" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ WRITE_DISABLE (0x80001000, 0xAB);

Table 54: Macros to be used for a 32-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CMD_CFG_WR_STATUS_READ	The opcode for the SPI flash "status register read" command.	This macro initialized the SPI flash controller with the SPI flash's "status register read" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ STATUS_READ (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_STATUS_WRITE	The opcode for the SPI flash "status register write" command.	This macro initialized the SPI flash controller with the SPI flash's "status register write" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ STATUS_WRITE (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_POWER_DOWN	The opcode for the SPI flash "power down" command.	This macro initialized the SPI flash controller with the SPI flash's "power down" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ POWER_DOWN (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_POWER_UP	The opcode for the SPI flash "power up" command.	This macro initialized the SPI flash controller with the SPI flash's "power up" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ POWER_UP (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_MANUFACTURER_ID	The opcode for the SPI flash "read manufacturer ID" command.	This macro initialized the SPI flash controller with the SPI flash's "read manufacturer ID" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ MANUFACTURER_ID (0x80001000, 0xAB);

Table 55: Macros to be used for an 8-bit Data Bus on the WISHBONE Control Port

Macro Name	Second Argument to Macro	Description
MICO_SPI_PAGE_ PROGRAM_BYTEWISE	The 24-bit address in the SPI flash where data programming will commence.	This macro initiates a SPI flash "page program" by writing to the "Page Program" register. Usage: If SPI flash controller's base address is 0x80001000 and SPI flash address to be programmed is 0x100000, MICO_SPI_PAGE_PROGRAM (0x80001000, 0x100000); **NOTE** : The software developer must ensure that the Page Program Buffer is enabled in hardware and it contains the data that needs to be programmed.
MICO_SPI_PAGE_READ_ BYTEWISE	The 24-bit address in the SPI flash where data reading will commence.	This macro initiates a SPI flash "page read" by writing to the "Page Read" register. Usage: If SPI flash controller's base address is 0x80001000 and SPI flash address to be read from is 0x100000, MICO_SPI_PAGE_READ (0x80001000, 0x100000); **NOTE**: The software developer must ensure that the Page Read Buffer is enabled in hardware.
MICO_SPI_BLOCK_ERASE _TYPE1_BYTEWISE	The 24-bit address of the sector in SPI flash that needs to be erased.	This macro initiates a SPI flash "sector erase" command to erase a 4Kbytes sector. Usage: If SPI flash controller's base a address is 0x80001000 and SPI flash sector address is 0x100000, MICO_SPI_BLOCK_ERASE_TYPE1 (0x80001000, 0x100000);
MICO_SPI_BLOCK_ERASE _TYPE2_BYTEWISE	The 24-bit address of the sector in SPI flash that needs to be erased.	This macro initiates a SPI flash "sector erase" command to erase a 32Kbytes sector. Usage: If SPI flash controller's base a address is 0x80001000 and SPI flash sector address is 0x100000, MICO_SPI_BLOCK_ERASE_TYPE2 (0x80001000, 0x100000);
MICO_SPI_BLOCK_ERASE _TYPE3_BYTEWISE	The 24-bit address of the sector in SPI flash that needs to be erased.	This macro initiates a SPI flash "sector erase" command to erase a 64Kbytes sector. Usage: If SPI flash controller's base a address is 0x80001000 and SPI flash sector address is 0x100000, MICO_SPI_BLOCK_ERASE_TYPE3 (0x80001000, 0x100000);

Table 55: Macros to be used for an 8-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CHIP_ERASE_ BYTEWISE		This macro initiates the SPI flash "chip erase" command to erase the entire chip. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_CHIP_ERASE (0x80001000);
MICO_SPI_WRITE_ ENABLE_BYTEWISE		This macro initiates the SPI flash "write enable" command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_WRITE_ENABLE(0x80001000);
MICO_SPI_WRITE_ DISABLE_BYTEWISE		This macro initiates the SPI flash "write disable" command. Usage: If SPI flash controller's base address is 0x800010000, MICO_SPI_WRITE_DISABLE(0x80001000);
MICO_SPI_STATUS_READ _BYTEWISE	The status register read from the SPI flash.	This macro initiates the SPI flash "status register read" command. Usage: If SPI flash controller's base address is 0x80001000, unsigned int status; MICO_SPI_STATUS_READ (0x80001000, status);
MICO_SPI_STATUS_ WRITE_BYTEWISE	The value written to the status register in the SPI flash.	This macro initiates the SPI flash "status register write" command. Usage: If SPI flash controller's base address is 0x80001000, and the value to be written to SPI flash's status register is 0x2, unsigned int status = 0x2; MICO_SPI_STATUS_WRITE (0x80001000, status);
MICO_SPI_POWER_ DOWN_BYTEWISE		This macro initiates the SPI flash "power down" command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_POWER_DOWN (0x80001000);
MICO_SPI_POWER_UP_ BYTEWISE		This macro initiates the SPI flash "power up" command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_POWER_UP (0x80001000);

Table 55: Macros to be used for an 8-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_ MANUFACTURER_ID_ BYTEWISE	The manufacturer ID read from the SPI flash.	This macro initiates the SPI flash "read manufacturer ID" command. Usage: If SPI flash controller's base address is 0x80001000, unsigned int ID; MICO_SPI_MANUFACTURER_ID (0x80001000, ID);
MICO_SPI_CUSTOM_LSW _BYTEWISE	The least-significant word of the user-defined command that is issued to the SPI flash. **NOTE**: The first byte to be transmitted to SPI flash is the most-significant byte of this word.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to provide the least-significant word of the user-defined command. Usage: If SPI flash controller's base address is 0x80001000, and the sequence of four bytes to be transmitted are 0x12, 0x34, 0x56, and then 0x78, MICO_SPI_CUSTOM_LSW (0x80001000, 0x12345678);
MICO_SPI_CUSTOM_MSW _BYTEWISE	The most-significant word of the user-defined command that is issued to the SPI flash. **NOTE**: The first byte to be transmitted to SPI flash is the most-significant byte of this word.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to provide the most-significant word of the user-defined command. Usage: If SPI flash controller's base address is 0x80001000, and the sequence of four bytes to be transmitted are 0x12, 0x34, 0x56, and then 0x78, MICO_SPI_CUSTOM_MSW (0x80001000, 0x12345678); **NOTE**: This macro is optional since the SPI flash command could be four bytes or less.
MICO_SPI_CUSTOM_ LENGTH_BYTEWISE	The length, in bytes, of the user-defined command that is issued to the SPI flash.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to indicate the number of bytes transmitted as part of the user command. Usage: If SPI flash controller's base address is 0x80001000 and the number of bytes transmitted is 1, then MICO_SPI_CUSTOM_LENGTH (0x80001000, 0x1);

Table 55: Macros to be used for an 8-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CUSTOM_ RETURN_LENGTH_ BYTEWISE	The length, in bytes, of the data returned by the SPI flash for the user-defined command that is issued.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to indicate the number of bytes returned from the SPI flash as part of the user command. The return length can be zero. Usage: If SPI flash controller's base address is 0x80001000 and the number of bytes returned is 4, then MICO_SPI_CUSTOM_RETURN_LENGTH (0x80001000, 0x4);
MICO_SPI_CUSTOM_ RETURN_DATA_ BYTEWISE	The data returned by the SPI flash for the user-defined command. **NOTE**: The first byte returned from the SPI flash is the most-significant byte of this word.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to obtain the data returned by the SPI flash if return length is non-zero. Usage: If SPI flash controller's base address is 0x80001000 and the number of bytes returned is 4 and the sequence received from SPI flash is 0x12, 0x23, 0x56, then 0x78. unsigned int retValue; MICO_SPI_CUSTOM_RETURN_DATA (0x80001000, retValue); retValue will be equal to 0x12345678.
MICO_SPI_CUSTOM_ BYTEWISE		This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to initiate transmission of user-defined command to SPI flash. **NOTE**: The software developer must make sure that the user-defined command, length, and return length are set up prior to use of this macro.

MICO_SPI_CFG_WR_ PAGE_PROGRAM_SIZE_ BYTEWISE	The number of bytes to be programmed on a SPI flash "page program".	This macro is used to indicate the number of bytes to be programmed in to SPI flash on a SPI flash "page program" command.
		Usage:
		If SPI flash controller's base address is 0x80001000 and the number of bytes to be programmed is 256, then
		unsigned int size = 0x100;
		MICO_SPI_CFG_WR_PAGE_PROGRAM_SIZE (0x80001000, size);

Table 55: Macros to be used for an 8-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CFG_WR_ PAGE_READ_SIZE_ BYTEWISE	The number of bytes to be read on a SPI flash "page read".	This macro is used to indicate the number of bytes to be read from the SPI flash on a SPI flash "page read" command.
		Usage:
		If SPI flash controller's base address is 0x80001000 and the number of bytes to be read is 256, then
		unsigned int size = 0x100;
		MICO_SPI_CFG_WR_PAGE_READ_SIZE (0x80001000, size);

MICO_SPI_CMD_CFG_WR _SLOW_READ_BYTEWISE	The opcode for the SPI flash "slow read" command.	This macro initialized the SPI flash controller with the SPI flash's "slow read" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_SLOW_READ (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR _FAST_READ_BYTEWISE	The opcode for the SPI flash "fast read" command.	This macro initialized the SPI flash controller with the SPI flash's "fast read" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_FAST_READ (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR _BYTE_PROGRAM_ BYTEWISE	The opcode for the SPI flash "byte program" command.	This macro initialized the SPI flash controller with the SPI flash's "byte program" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ BYTE_PROGRAM (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR _PAGE_PROGRAM_ BYTEWISE	The opcode for the SPI flash "page program" command.	This macro initialized the SPI flash controller with the SPI flash's "page program" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ PAGE_PROGRAM (0x80001000, 0xAB);

Table 55: Macros to be used for an 8-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CMD_CFG_WR_BLOCK_ERASE_TYPE1_BYTEWISE	The opcode for the SPI flash "block erase type 1" command.	This macro initialized the SPI flash controller with the SPI flash's "block erase type 1" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE1 (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_BLOCK_ERASE_TYPE2_BYTEWISE	The opcode for the SPI flash "block erase type 2" command.	This macro initialized the SPI flash controller with the SPI flash's "block erase type 2" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE2 (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_BLOCK_ERASE_TYPE3_BYTEWISE	The opcode for the SPI flash "block erase type 3" command.	This macro initialized the SPI flash controller with the SPI flash's "block erase type 3" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE3 (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_CHIP_ERASE_BYTEWISE	The opcode for the SPI flash "chip erase" command.	This macro initialized the SPI flash controller with the SPI flash's "chip erase" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ CHIP_ERASE (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_WRITE_ENABLE_BYTEWISE	The opcode for the SPI flash "write enable" command.	This macro initialized the SPI flash controller with the SPI flash's "write enable" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ WRITE_ENABLE (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_WRITE_DISABLE_BYTEWISE	The opcode for the SPI flash "write disable" command.	This macro initialized the SPI flash controller with the SPI flash's "write disable" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ WRITE_DISABLE (0x80001000, 0xAB);

Table 55: Macros to be used for an 8-bit Data Bus on the WISHBONE Control Port (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CMD_CFG_WR_STATUS_READ_BYTEWISE	The opcode for the SPI flash "status register read" command.	This macro initialized the SPI flash controller with the SPI flash's "status register read" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ STATUS_READ (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_STATUS_WRITE_BYTEWISE	The opcode for the SPI flash "status register write" command.	This macro initialized the SPI flash controller with the SPI flash's "status register write" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ STATUS_WRITE (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_POWER_DOWN_BYTEWISE	The opcode for the SPI flash "power down" command.	This macro initialized the SPI flash controller with the SPI flash's "power down" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ POWER_DOWN (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_POWER_UP_BYTEWISE	The opcode for the SPI flash "power up" command.	This macro initialized the SPI flash controller with the SPI flash's "power up" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ POWER_UP (0x80001000, 0xAB);
MICO_SPI_CMD_CFG_WR_MANUFACTURER_ID_BYTEWISE	The opcode for the SPI flash "read manufacturer ID" command.	This macro initialized the SPI flash controller with the SPI flash's "read manufacturer ID" command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, MICO_SPI_CMD_CFG_WR_ MANUFACTURER_ID (0x80001000, 0xAB);

Accessing SPI Flash Controller without Device Drivers

The device driver functions and macros hide the SPI flash controller's implementation from the software developer by providing a software translation layer between the developer's application and the actual hardware-specific details. It is, nevertheless, possible to access the SPI flash controller directly without using Lattice-provided drivers. You can do this by accessing (reading from or writing to) the registers defined in Table 3 on page 7. The orientation of data is very important, because the LatticeMico32 microprocessor is a big-endian microprocessor. Therefore, the software developer is advised to read the following information to understand the impact of endianness when the microprocessor interacts with the component's registers. In a big-endian architecture, the most-significant byte of a multi-byte object is stored at the lowest address, and the least-significant byte of that object is stored at the highest address.

Assume that you have a design that contains the SPI flash controller and that it is assigned a base address of 0x80000000. Now let's consider that one wants to write to the "Slow Read CFG" register at an offset of 0x180 from the base address. Valid data is located in bits 7-0, while the rest of the bits (31-8) are ignored. There are two ways in C to write to this register depending on whether one is performing a byte ("unsigned char" or "signed char") write or whether one is performing a word ("unsigned int" or "signed int") write. Figure 2 shows sample code that uses a byte write.

Figure 2: Correct access to Slow Read CFG register using byte write

```
unsigned char opcode = 0xAB;
*(volatile unsigned char *)0x80000180 = opcode;
```

Figure 3 shows sample code that uses a word write.

Figure 3: Correct access to Slow Read CFG register using word write

```
unsigned int opcode = 0xAB000000;
*(volatile unsigned int *)0x80000180 = opcode;
```

As mentioned before, LatticeMico32 microprocessor is a big-endian microprocessor. Therefore from the programmer's perspective, the least-significant byte (bits 7-0) of the Slow Read CFG register will appear in the most-significant location of "opcode" variable. The sample code in Figure 3 will write a value of 0xAB to bits 7-0 of the Slow Read CFG register.

Figure 4: Incorrect access to Slow Read CFG register using word write

```
unsigned int opcode = 0xAB;
*(volatile unsigned int *)0x80000180 = opcode;
```

On the other hand, the sample code in Figure 4 will produce incorrect behavior. The C/C++ compiler will promote the 8-bit value 0xAB in to a 32-bit

value of 0x000000AB. Thus the most-significant byte is 0x00 and that gets written to bits 7-0 instead of the value 0xAB.

LatticeMico8 Microcontroller Software Support

This section describes the LatticeMico8 microcontroller software support provided for the LatticeMico SPI flash controller.

Device Driver

The SPI flash driver interacts directly with the SPI flash controller and, in turn, the SPI flash device. This section describes the type definitions, structure, and functions of the SPI flash device driver.

Context Structure

This section describes the type definitions for the SPI flash controller context structure. It is shown in Figure 5, which contains SPI flash controller-specific information that is dynamically generated in the DDStructs.h header file. This information is largely filled in by the MSB managed build process, which extracts SPI flash controller-specific information from the platform definition file. As part of the managed build process, designers can choose to control the size of the generated structure, and hence the software executable, by selectively enabling some of the elements in this structure via C preprocessor macro definitions. These C preprocessor macro definitions are explained later in this document. The members should not be manipulated directly since this structure is for exclusive use by the device driver. The SPI flash controller context structure is shown in Figure 5.

Figure 5: LatticeMico8 Type Definitions for SPI Flash Controller Context Structure

```
typedef  struct st_MicoSPIFlashCtx_t {
  const char *name;
#ifdef __MICOSPIFLASH_CONTROL__
  size_t control_base;
  unsigned int page_size;
  unsigned int sector_size;
#endif
}
```

Table 56 describes the parameters of the SPI Flash device context structure shown in Figure 5. The table also identifies any C preprocessor 'macro definition' that controls the the generation of the parameter. If the 'state' associated with a C preprocessor 'macro definition' is 'ifdef', then it means that the application must be compiled with this macro definition for the parameter to be generated. If the 'state' associated with a C preprocessor 'macro definition' is 'ifndef', then it means that the application must be compiled without this macro definition for the parameter to be generated.

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters

Macro Name	Second Argument to Macro	Description
MICO_SPI_PAGE_ PROGRAM	The 24-bit address in the SPI flash where data programming will commence.	This macro initiates a SPI flash ‘page program’ by writing to the ‘Page Program’ register. Usage: If SPI flash controller’s base address is 0x80001000 and SPI flash address to be programmed is 0x100000, MICO_SPI_PAGE_PROGRAM(0x80001000, 0x100000); NOTE : The software developer must ensure that the Page Program Buffer is enabled in hardware and it contains the data that needs to be programmed.
MICO_SPI_PAGE_READ	The 24-bit address in the SPI flash where data reading will commence.	This macro initiates a SPI flash ‘page read’ by writing to the ‘Page Read’ register. Usage: If SPI flash controller’s base address is 0x80001000 and SPI flash address to be read from is 0x100000, MICO_SPI_PAGE_READ(0x80001000, 0x100000); NOTE: The software developer must ensure that the Page Read Buffer is enabled in hardware.
MICO_SPI_BLOCK_ERASE_ TYPE1	The 24-bit address of the sector in SPI flash that needs to be erased.	This macro initiates a SPI flash ‘sector erase’ command to erase a 4Kbytes sector. Usage: If SPI flash controller’s base a address is 0x80001000 and SPI flash sector address is 0x100000, MICO_SPI_BLOCK_ERASE_TYPE1(0x80001000, 0x100000);
MICO_SPI_BLOCK_ERASE_ TYPE2	The 24-bit address of the sector in SPI flash that needs to be erased.	This macro initiates a SPI flash ‘sector erase’ command to erase a 32Kbytes sector. Usage: If SPI flash controller’s base a address is 0x80001000 and SPI flash sector address is 0x100000, MICO_SPI_BLOCK_ERASE_TYPE2(0x80001000, 0x100000);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_BLOCK_ERASE_TYPE3	The 24-bit address of the sector in SPI flash that needs to be erased.	This macro initiates a SPI flash 'sector erase' command to erase a 64Kbytes sector. Usage: If SPI flash controller's base a address is 0x80001000 and SPI flash sector address is 0x100000, MICO_SPI_BLOCK_ERASE_TYPE3(0x80001000, 0x100000);
MICO_SPI_CHIP_ERASE	-	This macro initiates the SPI flash 'chip erase' command to erase the entire chip. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_CHIP_ERASE(0x80001000);
MICO_SPI_WRITE_ENABLE	-	This macro initiates the SPI flash 'write enable' command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_WRITE_ENABLE(0x80001000);
MICO_SPI_WRITE_DISABLE	-	This macro initiates the SPI flash 'write disable' command. Usage: If SPI flash controller's base address is 0x800010000, MICO_SPI_WRITE_DISABLE(0x80001000);
MICO_SPI_STATUS_READ	The status register read from the SPI flash.	This macro initiates the SPI flash 'status register read' command. Usage: If SPI flash controller's base address is 0x80001000, unsigned char status; MICO_SPI_STATUS_READ(0x80001000, status);
MICO_SPI_STATUS_WRITE	The value written to the status register in the SPI flash.	This macro initiates the SPI flash 'status register write' command. Usage: If SPI flash controller's base address is 0x80001000, and the value to be written to SPI flash's status register is 0x2, unsigned char status = 0x2; MICO_SPI_STATUS_WRITE(0x80001000, status);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_POWER_DOWN	-	This macro initiates the SPI flash 'power down' command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_POWER_DOWN(0x80001000);
MICO_SPI_POWER_UP	-	This macro initiates the SPI flash 'power up' command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_POWER_UP(0x80001000);
MICO_SPI_ MANUFACTURER_ID	The manufacturer ID read from the SPI flash.	This macro initiates the SPI flash 'read manufacturer ID' command. Usage: If SPI flash controller's base address is 0x80001000, MICO_SPI_MANUFACTURER_ID(0x80001000);
MICO_SPI_CUSTOM_LSW	The least-significant word of the user-defined command that is issued to the SPI flash. NOTE: The first byte to be transmitted to SPI flash is the most-significant byte of this word.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to provide the least-significant word of the user-defined command. Usage: If SPI flash controller's base address is 0x80001000, and the sequence of four bytes to be transmitted are 0x12, 0x34, 0x56, and then 0x78, MICO_SPI_CUSTOM_LSW(0x80001000, 0x12345678);
MICO_SPI_CUSTOM_MSW	The most-significant word of the user-defined command that is issued to the SPI flash. NOTE: The first byte to be transmitted to SPI flash is the most-significant byte of this word.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to provide the most-significant word of the user-defined command. Usage: If SPI flash controller's base address is 0x80001000, and the sequence of four bytes to be transmitted are 0x12, 0x34, 0x56, and then 0x78, MICO_SPI_CUSTOM_MSW(0x80001000, 0x12345678); NOTE: This macro is optional since the SPI flash command could be four bytes or less.

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CUSTOM_ LENGTH	The length, in bytes, of the user-defined command that is issued to the SPI flash.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to indicate the number of bytes transmitted as part of the user command. Usage: If SPI flash controller's base address is 0x80001000 and the number of bytes transmitted is 1, then MICO_SPI_CUSTOM_LENGTH(0x80001000, 0x1);
MICO_SPI_CUSTOM_ RETURN_LENGTH	The length, in bytes, of the data returned by the SPI flash for the user-defined command that is issued.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to indicate the number of bytes returned from the SPI flash as part of the user command. The return length can be zero. Usage: If SPI flash controller's base address is 0x80001000 and the number of bytes returned is 4, then MICO_SPI_CUSTOM_RETURN_LENGTH(0x80001000 , 0x4);
MICO_SPI_CUSTOM_ RETURN_DATA	The return data on a SPI flash user-defined command.	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to obtain the data returned by the SPI flash if return length is non-zero. Usage: If SPI flash controller's base address is 0x80001000 and the number of bytes returned is 4 and the sequence received from SPI flash is 0x12, 0x23, 0x56, then 0x78. unsigned long retValue; MICO_SPI_CUSTOM_RETURN_DATA(0x80001000, retValue); retValue will be equal to 0x12345678.

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description
MICO_SPI_CUSTOM	-	This macro is part of the group of macros that are used to create user-defined commands to be issued to the SPI flash. These macros are useful when the SPI flash supports additional commands that are not implemented in this SPI flash controller. This macro is used to initiate transmission of user-defined command to SPI flash. NOTE: The software developer must make sure that the user-defined command, length, and return length are set up prior to use of this macro.
MICO_SPI_CFG_RD_READ_SPEED	The return data.	This macro is used to obtain the contents of 'Read Speed' register from Table 3 on page 7. Usage: If SPI flash controller's base address is 0x80001000, unsigned char speed; MICO_SPI_CFG_RD_READ_SPEED(0x80001000, speed);
MICO_SPI_CFG_WR_READ_SPEED	Fast read (0x1) or Slow read (0x0)	This macro is used to set the default 'Read Speed' to be used on a SPI flash 'page read' command. Usage: If SPI flash controller's base address is 0x80001000, unsigned char speed = 0x1; // 0x1 (fast), 0x0 (slow) MICO_SPI_CFG_WR_READ_SPEED(0x80001000, speed);

MICO_SPI_CFG_RD_PAGE_PROGRAM_SIZE	The number of bytes to be programmed on a SPI flash 'page program' returned by the SPI flash controller.	This macro is used to obtain the contents of 'Page Program Size' register from Table 3 on page 7.
		
		Usage:
		If SPI flash controller’s base address is 0x80001000, then
		
		unsigned int size; // value from 'Page Program Size' register is stored in this variable
		MICO_SPI_CFG_RD_PAGE_PROGRAM_SIZE(0x80001000, size);

MICO_SPI_CFG_WR_PAGE_PROGRAM_SIZE	The number of bytes to be programmed on a SPI flash ‘page program’.	This macro is used to indicate the number of bytes to be programmed in to SPI flash on a SPI flash ‘page program’ command.
		
		Usage:
		If SPI flash controller’s base address is 0x80001000 and the number of bytes to be programmed is 256, then
		
		unsigned int size = 0x100;
		MICO_SPI_CFG_WR_PAGE_PROGRAM_SIZE(0x80001000, size);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CFG_RD_PAGE_READ_SIZE	Opcode for the SPI flash 'page read' command returned by the SPI flash controller.	This macro is used to obtain the contents of 'Page Read Size' register from Table 3 on page 7.
		
		Usage:
		If SPI flash controller's base address is 0x80001000, then
		
		unsigned int size; // value from 'Page Read Size' register is stored in this variable
		MICO_SPI_CFG_RD_PAGE_READ_SIZE(0x80001000, size);

MICO_SPI_CFG_WR_PAGE_READ_SIZE	The number of bytes to be read on a SPI flash 'page read'.	This macro is used to indicate the number of bytes to be read from the SPI flash on a SPI flash 'page read' command.
		
		Usage:
		If SPI flash controller's base address is 0x80001000 and the number of bytes to be read is 256, then
		
		unsigned int size = 0x100;
		MICO_SPI_CFG_WR_PAGE_PROGRAM_SIZE(0x80001000, size);

MICO_SPI_CMD_CFG_RD_SLOW_READ	Opcode for the SPI flash 'slow read' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'slow read' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Slow Read CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_SLOW_READ(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_SLOW_READ	The opcode for the SPI flash 'slow read' command.	This macro initialized the SPI flash controller with the SPI flash's 'slow read' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_SLOW_READ(0x80001000, opcode);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CMD_CFG_RD_FAST_READ	Opcode for the SPI flash 'fast read' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'fast read' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Fast Read CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_FAST_READ(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_FAST_READ	The opcode for the SPI flash 'fast read' command.	This macro initialized the SPI flash controller with the SPI flash's 'fast read' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_FAST_READ(0x80001000, opcode);

MICO_SPI_CMD_CFG_RD_BYTE_PROGRAM	Opcode for the SPI flash 'byte program' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'byte program' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Byte Program CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_BYTE_PROGRAM(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_BYTE_PROGRAM	The opcode for the SPI flash 'byte program' command.	This macro initialized the SPI flash controller with the SPI flash's 'byte program' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ BYTE_PROGRAM (0x80001000, opcode);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CMD_CFG_RD_ PAGE_PROGRAM	Opcode for the SPI flash 'page program' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'page program' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller’s base address is 0x80001000,
		
		unsigned char opcode; // value from 'Page Program CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_PAGE_PROGRAM(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_ PAGE_PROGRAM	The opcode for the SPI flash ‘page program’ command.	This macro initialized the SPI flash controller with the SPI flash’s ‘page program’ command opcode. Usage: If SPI flash controller’s base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ PAGE_PROGRAM (0x80001000, opcode);

MICO_SPI_CMD_CFG_RD_ BLOCK_ERASE_TYPE1	Opcode for the SPI flash 'block erase type 1' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'block erase (type 1)' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller’s base address is 0x80001000,
		
		unsigned char opcode; // value from 'Block Erase 1 CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_BLOCK_ERASE_TYPE1(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE1	The opcode for the SPI flash ‘block erase type 1’ command.	This macro initialized the SPI flash controller with the SPI flash’s ‘block erase type 1’ command opcode. Usage: If SPI flash controller’s base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE1 (0x80001000, opcode);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CMD_CFG_RD_ BLOCK_ERASE_TYPE2	Opcode for the SPI flash 'block erase type 2' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'block erase (type 2)' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Block Erase 2 CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_BLOCK_ERASE_TYPE2(0 x80001000, opcode);

MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE2	The opcode for the SPI flash 'block erase type 2' command.	This macro initialized the SPI flash controller with the SPI flash's 'block erase type 2' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE2 (0x80001000, opcode);

MICO_SPI_CMD_CFG_RD_ BLOCK_ERASE_TYPE3	Opcode for the SPI flash 'block erase type 3' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'block erase (type 3)' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Block Erase 3 CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_BLOCK_ERASE_TYPE3(0 x80001000, opcode);

MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE3	The opcode for the SPI flash 'block erase type 3' command.	This macro initialized the SPI flash controller with the SPI flash's 'block erase type 3' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ BLOCK_ERASE_TYPE3 (0x80001000, opcode);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CMD_CFG_RD_ CHIP_ERASE	Opcode for the SPI flash 'chip erase' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'chip erase' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Chip Erase CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_CHIP_ERASE(0x80001000 , opcode);

MICO_SPI_CMD_CFG_WR_ CHIP_ERASE	The opcode for the SPI flash 'chip erase' command.	This macro initialized the SPI flash controller with the SPI flash's 'chip erase' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ CHIP_ERASE (0x80001000, opcode);

MICO_SPI_CMD_CFG_RD_ WRITE_ENABLE	Opcode for the SPI flash 'write enable' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'write enable' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Write Enable CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_WRITE_ENABLE(0x80001 000, opcode);

MICO_SPI_CMD_CFG_WR_ WRITE_ENABLE	The opcode for the SPI flash 'write enable' command.	This macro initialized the SPI flash controller with the SPI flash's 'write enable' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ WRITE_ENABLE (0x80001000, opcode);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CMD_CFG_RD_ WRITE_DISABLE	Opcode for the SPI flash 'write disable' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'write disable' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Write Disable CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_WRITE_DISABLE(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_ WRITE_DISABLE	The opcode for the SPI flash 'write disable' command.	This macro initialized the SPI flash controller with the SPI flash's 'write disable' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ WRITE_DISABLE (0x80001000, opcode);

MICO_SPI_CMD_CFG_RD_ STATUS_READ	Opcode for the SPI flash 'status register read' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'status read' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Status Read CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_STATUS_READ(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_ STATUS_READ	The opcode for the SPI flash 'status register read' command.	This macro initialized the SPI flash controller with the SPI flash's 'status register read' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ STATUS_READ (0x80001000, opcode);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CMD_CFG_RD_ STATUS_WRITE	Opcode for the SPI flash 'status register write' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'status write' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller’s base address is 0x80001000,
		
		unsigned char opcode; // value from 'Status Write CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_STATUS_WRITE(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_ STATUS_WRITE	The opcode for the SPI flash ‘status register write’ command.	This macro initialized the SPI flash controller with the SPI flash’s ‘status register write’ command opcode. Usage: If SPI flash controller’s base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ STATUS_WRITE (0x80001000, opcode);

MICO_SPI_CMD_CFG_RD_ POWER_DOWN	Opcode for the SPI flash 'power down' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'power down' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller’s base address is 0x80001000,
		
		unsigned char opcode; // value from 'Power Down CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_POWER_DOWN(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_ POWER_DOWN	The opcode for the SPI flash ‘power down’ command.	This macro initialized the SPI flash controller with the SPI flash’s ‘power down’ command opcode. Usage: If SPI flash controller’s base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ POWER_DOWN (0x80001000, opcode);

Table 56: LatticeMico8 SPI Flash Controller Context Structure Parameters (Continued)

Macro Name	Second Argument to Macro	Description

MICO_SPI_CMD_CFG_RD_ POWER_UP	Opcode for the SPI flash 'power up' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'power up' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Power Up CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_POWER_UP(0x80001000, opcode);

MICO_SPI_CMD_CFG_WR_ POWER_UP	The opcode for the SPI flash 'power up' command.	This macro initialized the SPI flash controller with the SPI flash's 'power up' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ POWER_UP (0x80001000, opcode);

MICO_SPI_CMD_CFG_RD_ MANUFACTURER_ID	Opcode for the SPI flash 'read manufacturer ID' command returned by the SPI flash controller.	This macro is used to obtain the opcode of the SPI flash's 'manufacturer ID' command opcode that is current programmed within the SPI flash controller.
		
		Usage:
		If SPI flash controller's base address is 0x80001000,
		
		unsigned char opcode; // value from 'Manufacturer ID CFG' register is stored in this variable
		MICO_SPI_CMD_CFG_RD_MANUFACTURER_ID(0x80001000, opcode);

MICO_SPI_CMD_CFG_ WR_MANUFACTURER_ID	The opcode for the SPI flash 'read manufacturer ID' command.	This macro initialized the SPI flash controller with the SPI flash's 'read manufacturer ID' command opcode. Usage: If SPI flash controller's base address is 0x80001000 and the opcode is 0xAB, unsigned char opcode = 0xAB; MICO_SPI_CMD_CFG_WR_ MANUFACTURER_ID (0x80001000, opcode);

C Preprocessor Macro Definitions

This section describes the C preprocessor macro definitions that are available to the software developer. There are two types of macro definitions: 'object-like' and 'function-like'.

The 'object-like' macro definitions do not take any arguments and are used to control the size of the generated application executable. There are three ways an 'object-like' macro definition can be used by the software developer.

1. Manually adding the -D<macro name> option to the compiler's command-line in the application's 'Build Properties'. Please refer to the LatticeMico8 Software Developers User Guide for more information on how to manually add the macro definition in the the application's 'Build Properties' GUI.
2. Automatically adding the -D<macro name> option to the compiler's command-line in the application's 'Build Properties' by enabling the 'check-box' associated with the macro definition. Please refer to the LatticeMico8 Software Developers User Guide for more information on how to set up the check/uncheck the macro definitions in the application's 'Build Properties' GUI.
3. Manually adding the macro definition to the C code using the following syntax:

   ```
   #define <macro name>
   ```

It is recommended that the developer use options 1 or 2. The ‘object-like’ macro definitions are:

- __MICOSPIFLASH_CONTROL__

  This preprocessor macro definition enables code and data structures within the device driver that are used when the Control Port is enabled within the SPI flash controller. The Control Port is used for SPI flash operations that modify the state of the SPI flash (i.e., operations excluding a byte read or a byte write within an erased SPI flash page).

The 'function-like' macro definitions are used in the LatticeMico8 software drivers to access the component's Register Map in order to perform certain operations. All 'function-like' macro definitions take input parameters that are used in performing the operations encoded within the macro. Table 57 describes the 'function-like' macros available in the LatticeMico8 SPI Flash driver header file 'MicoSPIFlash.h'. Table 57 also shows how each macro can be used by the software developer in his application code.

Table 57: LatticeMico8 SPI Flash C Preprocessor Function-Like Macros

Parameters	Data Type	C Preprocessor Macro Definition		Description
		Name	State	
name	const char*			Instance-specific component name (entered in MSB)
control_base	size_t	__MICOSPIFLASH_CONTROL__	ifdef	MSB-assigned Control Port base address for this instance
page_size	unsigned int	__MICOSPIFLASH_CONTROL__	ifdef	Size of SPI Flash page in bytes
sector_size	unsigned int	__MICOSPIFLASH_CONTROL__	ifdef	Size of SPI Flash sector in bytes

Functions

This section describes the implemented device-driver-specific functions.

MicoSPIFlash_Initialize Function

```
void MicoSPIFlash_Initialize(MicoSPIFlashCtx *ctx);
```

This function initializes a LatticeMico SPI flash controller instance on the basis of the passed SPI flash controller context structure. This initialization function is responsible for initializing the interrupts or buffer parameters for interrupt-driven operation. As a part of the managed build process, the LatticeDDInit function calls this initialization routine for each SPI flash controller instance that is present in the platform. Table 58 describes the parameter in the MicoSPIFlash_Initialize function syntax.

Table 58: MicoSPIFlash_Initialize Function Parameter

Parameter	Description
MicoSPIFlashCtx_t *	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash controller instance.

MicoSPIFlash_PageOp Function

```
void MicoSPIFlash_PageOp(MicoSPIFlashCtx *ctx, unsigned long anAddress, char *rData, char read_op);
```

This function is used to perform bulk reads from, or writes to, the SPI flash. There are two ways that bulk reads/writes to SPI flash can be performed, depending on whether the page buffers (Page Program Buffer for writes and Page Read Buffer for reads) are enabled in hardware. When the Page Program Buffer is enabled, the function performs bulk writes by first writing the data to the SPI flash controller's Page Program Buffer and then sending this data to the SPI flash in one command. When the Page Program Buffer is disabled in hardware, the function performs these writes via the significantly slower method of writing the given data to SPI flash one byte per SPI flash command. Similarly, when the Page Read Buffer is enabled, the function performs buld reads by first reading the data from the SPI flash to the SPI flash controller's Page Read Buffer and then storing it in to the data array that is an input argument to the function. When the Page Read Buffer is disabled in hardware, the function performs these reads via the significantly slower method of reading the given data from SPI flash one byte per SPI flash command.

The function will only read/write a single page. The software developer must note that this function will stop reading data from or writing data to the SPI flash as soon as it reaches the end of a page. Table 59 describes the parameters in the MicoSPIFlash_PageOp function syntax.

Table 59: MicoSPIFlash_PageOp Function Parameters

Parameter	Description
MicoSPIFlashCtx_t *	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash controller instance.
unsigned long anAddress	The first location in the SPI flash memory where program will commence. It can be any location within the page.
char *rData	The array of bytes to be written to SPI flash. The software developer must ensure that the byte array is as large as the SPI flash page. Otherwise, the program can unexpectedly crash.
char read_op	Indicates whether we are performing a page read (0x1) or a page write (0x0).

MicoSPIFlash_BlockErase Function

```
void MicoSPIFlash_BlockErase(MicoSPIFlashCtx *ctx, unsigned long anAddress, unsigned char aType);
```

This function issues a command to the SPI flash device that is connected to the SPI flash controller instance to erase the block (also known as section) at the specified address. This function can issue one of the three types of block erases that are specified by the SPI flash command instructions defined in Block Erase 1 CFG, Block Erase 2 CFG, and Block Erase 3 CFG registers shown in Table 3 on page 7. This function requests that the SPI flash perform

a block erase, and then it exits. It does not wait for the SPI flash to actually complete erasing the block. This allows the software execution to continue even though the SPI flash is still in the process of erasing all data in specified block.

Note

Since it is possible that the SPI flash controller might receive a new SPI flash request from any of the two WISHBONE ports while an erase is in progress, the SPI flash controller hardware is designed to issue a new command to the SPI flash only when the SPI flash indicates it is not busy.

Table 60: MicoSPIFlash_BlockErase Function Parameters

Parameter	Description
MicoSPIFlashCtx_t *	Pointer to a valid MicoSPIFlashCtx_t structure representing a valid SPI flash controller instance.
unsigned long anAddress	Address of SPI flash sector/block to be erased.
unsigned char aType	Indicates the type of SPI flash block erase command to be used. This, in turn, also determines the size of block that is being erased. The three legal values are: 1 – Type 1 for block size of 4K 2 – Type 2 for block size of 32K 3 – Type 3 for block size of 64K

Software Usage Example

This section provides an example on how to use the LatticeMico8 device driver for the SPI flash controller. The example, shown in Figure 6, assumes the presence of a SPI flash controller named "SPIFlash" whose Control Port, Page Read Buffer, and Page Program Buffer are enabled.

Figure 6: Example of How to Use LatticeMico8 Device Driver for SPI Flash Controller

```
#include "DDStructs.h"
#include "MicoUtils.h"
#include "MicoSPIFlash.h"

char wdata[256];
char rdata[256];

int main(void)
{
  // Fetch the context for SPI flash controller named
"SPIFlash" from DDStructs.h
  MicoSPIFlashCtx_t *spiflash = &spi_flash_SPIFlash;

  // Set up read/write FIFOs with initial data
  unsigned int idx;
    for (idx = 0; idx < 256; idx++) {
    wdata[idx] = 0x12;
    rdata[idx] = 0x0;
    }

    // Perform a chip erase by calling the 'chip erase' macro
within MicoSPIFlash.h
    MICO_SPI_CHIP_ERASE (spiflash->control_base);

    // Fill the page at physical flash address 0 with contents
of 'wdata' array
    MicoSPIFlash_PageOp (spiflash, 0x0, wdata, 0);

  // Read the contents of page at physical flash address 0 in
to 'rdata' array
  MicoSPIFlash_PageOp (spiflash, 0x0, rdata, 1);

    return 0;
}
```

Accessing the SPI Flash from Data Port of SPI Flash Controller

The example shown in Figure 6 shows how a page can be read from (or written to) the SPI flash using the Control Port of the SPI flash controller. The software developer can also choose to bypass the SPI flash controller's device drivers and access the SPI flash directly using the Data (S) port of the SPI flash controller. The Data port provides a virtual one-to-one map of the physical SPI flash memory. This means that the software developer can directly access SPI flash contents by accessing it's equivalent virtual map. That is, physical address 0 within the SPI flash corresponds to virtual address 'Data port base address + 0x0' within the SPI flash controller.

In order to read a byte from physical address 0 of SPI flash,

```
unsigned char value = *(unsigned char *)0x01000000; // Data
port's base address is 0x01000000
```

In order to write a byte to physical address 10 of SPI flash,

```
*(unsigned char *)0x0200000A; // Data port's base address is
0x02000000
```

The software developer must follow the following rules when writing code to access the SPI flash from the Data Port:

1. Every read or write is a byte wide. Therefore it is recommended that the software developer work with 'char' data type.
2. The software developer must ensure that the Data port address being accessed is within the LatticeMico8 addressable range. The addressable range can be computed from two values set within the LatticeMico8 configuration GUI: Scratchpad base address, and Data addressable range. For example, if Scratchpad base is 0x00000000 and Data addressable range is 64K, then SPI flash controller's Data port address must be within 0x00010000.

Note

The byte read/write examples shown above require that LatticeMico8's addressable range be 4Gbyte in order to to be accessibly by LatticeMico8 when the Scratchpad base address is 0x00000000.

Revision History

Component Version	Description
1.0	Initial Release.
3.0 (7.0 SP2)	Version number change only. No RTL code change.
3.1	Added SPI flash programming feature.
3.2	Added read/write support.
3.3 (8.1 SP1)	The data busses on the two WISHBONE interfaces can be configured to be 8 or 32 bits. Register map updated to support 8-bit and 32-bit WISHBONE data bus.
3.4	Software support added for LatticeMico8.
3.5	Fixed custom command macros in the LatticeMico32 device drivers.

.

Trademarks