Section 52. Flash Memory with Support for Live Update

This section of the manual contains the following topics:

PIC32 Family Reference Manual

> **Note:** This family reference manual section is meant to serve as a complement to device data sheets. Depending on the device variant, this manual section may not apply to all PIC32 devices.
>
> Please consult the note at the beginning of the **"Flash Program Memory"** chapter in the current device data sheet to check whether this document supports the device you are using.
>
> Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Web site at: http://www.microchip.com

52.1 INTRODUCTION

This document describes techniques for programming the Flash memory on PIC32 devices that support Live Update. These devices contain up to two banks of Flash memory each with their own Boot Flash Memory (BFM) partition and Program Flash Memory (PFM) partition for storing user code or non-volatile data. The dual memory bank feature allows Flash to be programmed in one bank while executing from another for live updates of program memory. There are three methods by which the user can program this memory:

- Run-Time Self-Programming (RTSP) – performed by the user's software
- In-Circuit Serial Programming™ (ICSP™) – performed using a serial data connection to the device, which allows much faster programming than RTSP
- Enhanced Joint Test Action Group Programming (EJTAG) – performed by an EJTAG-capable programmer, using the EJTAG port of the device

RTSP techniques are described in this chapter. The ICSP and EJTAG methods are described in the *"PIC32 Flash Programming Specification"* (DS60001145), which is available for download from the Microchip Web site (www.microchip.com).

52.1.1 Run-Time Self-Programming (RTSP)

RTSP is used for applications where field upgradable firmware or non-volatile data storage is needed. Software that implements field upgradable firmware capability is called a Bootloader. Non-volatile data storage in Flash memory is often implemented using EEPROM simulation software which incorporates wear leveling that extends the usable life of the memory used for data storage. Complete example code for both of these techniques is available for download from the Microchip Web site (www.microchip.com).

52.1.2 Dual Memory Bank

The dual memory bank features supplied on some PIC32 devices lend significant advantages for Bootloader and EEPROM simulation software. Dual Flash memory banks allow code to be executing in one bank, while another bank is being erased or programmed, thereby avoiding CPU stalling during programming operations. Flash banks can be aliased or mapped into memory and selected for execution at start-up, either automatically or manually, allowing for high reliability field updates to application and Bootloader software. Error-Correcting Code (ECC) memory has also been incorporated on certain PIC32 devices, which extends the usable life of the Flash memory.

52.1.2.1 FLASH PARTITIONS

Each bank of Flash memory is divided into two logical Flash partitions: the PFM and the BFM (some PIC32 devices have two BFM partitions). The two BFM partitions are aliased into two regions at start-up. The order of the aliasing is determined by the Flash Sequence Codes (FSEQs) stored in special Configuration words in each bank at start-up. This ordering determines which partition of BFM memory is selected to be used as the start-up code upon a device Reset. The PFM partitions of each bank are mapped into upper and lower regions of the memory map. Start-up code, executing from BFM, can select the mapping of the two PFM partitions into the two PFM regions. Refer to the **"Memory Organization"** chapter in the specific device data sheet for detailed information on devices memory maps and the available options for Flash memory.

52.1.3   Addressing

PIC32 devices implement two address schemes: virtual and physical. Virtual addresses are exclusively used by the CPU to fetch and execute instructions as well as access peripherals. When programming or erasing Flash memory, the physical addresses is always the address used for the target operation.

Code protection on BFM is implemented by page, is enabled at reset and must be disabled prior to programming any BFM. Code protection of PFM is implemented using a watermark register, is disabled at reset and must be configured in the startup code at initialization to avoid inadvertent program or erasure of PFM.

52.2 CONTROL REGISTERS

Flash program, erase, and write protection operations are controlled using the following Non-Volatile Memory (NVM) control registers:

- **NVMCON: Programming Control Register**

  The NVMCON register is the control register for Flash program/erase operations. This register is used to select the operation to be performed, initiate the operation, and provide status of the result when the operation is complete.

- **NVMKEY: Programming Unlock Register**

  NVMKEY is a write-only register that is used to implement an unlock sequence to help prevent accidental writes/erasures of Flash or EEPROM memory or accidental changing the NVMSWAP and write permission settings.

- **NVMADDR: Flash Address Register**

  This register is used to store the physical target address for row, quad word and word programming as well as page erasing.

- **NVMDATAx: Flash Data Register ('x' = 0-3)**

  These registers hold the data to be programmed during Flash Word program operations. NVMDATA3 through NVMDATA0 are used for Quad Word (128-bit) programming while only NVMDATA0 is used for Word (32-bit) Programming.

- **NVMSRCADDR: Source Data Address Register**

  This register is used to point to the physical address of the data to be programmed when executing a row program operation.

- **NVMPWP: Program Flash Write-Protect Register**

  This register is used to set the boundary of the last write protected page in the Program Flash partition.

- **NVMBWP: Flash Boot (Page) Write-Protect Register**

  This register is used to configure which Boot Flash partition pages are write-protected.

- **NVMCON2: Flash Programming Control Register 2**

  This register is used to control whether the Boot and Program Flash memory regions can be swapped during run-time (see **Note**).

> **Note:** This register is not available on all devices. Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine availability.

Table 52-1 provides a brief summary of all of the Flash-programming-related registers. Corresponding registers appear after the summary, followed by a detailed description.

Table 52-1: Flash Controller SFR Summary

Register Name	Bit Range	Bit 31/15	Bit 30/14	Bit 29/13	Bit 28/12	Bit 27/11	Bit 26/10	Bit 25/9	Bit 24/8	Bit 23/7	Bit 22/6	Bit 21/5	Bit 20/4	Bit 19/3	Bit 118/2	Bit 17/1	Bit 16/0
NVMCON[1]	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15:0	WR	WREN	WRERR	LVDERR	—	—	—	—	SWAP[2] / PFSWAP[2]	BFSWAP[2]	—	—	NVMOP<3:0>			
NVMKEY	31:16	NVMKEY<31:16>															
	15:0	NVMKEY<15:0>															
NVMADDR[1]	31:16	NVMADDR<31:16>															
	15:0	NVMADDR<15:0>															
NVMDATA3[1]	31:16	NVMDATA<31:16>															
	15:0	NVMDATA<15:0>															
NVMDATA2[1]	31:16	NVMDATA<31:16>															
	15:0	NVMDATA<15:0>															
NVMDATA1[1]	31:16	NVMDATA<31:16>															
	15:0	NVMDATA<15:0>															
NVMDATA0[1]	31:16	NVMDATA<31:16>															
	15:0	NVMDATA<15:0>															
NVM SRCADDR[1]	31:16	NVMSRCADDR<31:16>															
	15:0	NVMSRCADDR<15:0>															
NVMPWP[1]	31:16	PWPULOCK	—	—	—	—	—	—	—	PWP<23:16>							
	15:0	PWP<15:0>															
NVMBWP[1]	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15:0	LBWPULOCK	—	—	LBWP4	LBWP3	LBWP2	LBWP1	LBWP0	UBWPULOCK[2]	—	—	UBWP4[2]	UBWP3[2]	UBWP2[2]	UBWP1[2]	UBWP0[2]
NVMCON2[1,2]	31:16	NVMERS<3:0>				—	—	—	—	—	—	—	NVMWS<4:0>				
	15:0	NVMLPRD	—	NVMCRD	NVMVRD	—	—	NVMRETRY<1:0>		SWAPLOCK<1:0>		—	—	—	—	—	—

Legend: — = unimplemented, read as '0'.

Note 1: This register has an associated Clear, Set, and Invert register at an offset of 0x4, 0x8, and 0xC bytes, respectively. These registers have the same name with CLR, SET, or INV appended to the end of the register name (e.g., NVMCONCLR). Writing a '1' to any bit position in these registers will clear, set, or invert valid bits in the associated register. Reads from these registers should be ignored.

2: This bit or register is not available on all devices. Refer to the "**Flash Program Memory**" chapter in the specific device data sheet to determine availability.

**** Important Note ****

Register 52-1: **NVMCON:** Programming Control Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0, HC	R/W-0	R-0, HS, HC	R-0, HS, HC	U-0	U-0	U-0	U-0
	WR[2]	WREN	WRERR	LVDERR[2]	—	—	—	—
7:0	R/W-0 SWAP[3] PFSWAP[3]	R/W-0 BFSWAP[3]	U-0 —	U-0 —	R/W-0	R/W-0	R/W-0	R/W-0
						NVMOP<3:0>		

Legend:		HS = Set by Hardware	HC = Cleared by Hardware
R = Readable bit		W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR		'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

bit 31-16 **Unimplemented:** Read as '0'

bit 15 **WR:** Write Control bit[2]

This bit cannot be cleared and can be set only when WREN = 1 and the unlock sequence has been performed.
1 = Initiate a Flash operation
0 = Flash operation is complete or inactive

bit 14 **WREN:** Write Enable bit

1 = Enable writes to the WR bit and disables writes to the SWAP or PFSWAP bit and the BFSWAP and NVMOP<3:0> bits
0 = Disable writes to WR bit and enables writes to the SWAP or PFSWAP bit and the BFSWAP and NVMOP<3:0> bits

bit 13 **WRERR:** Write Error bit[2]

This bit can be cleared only by setting the NVMOP<3:0> bits = 0000 (NOP) and initiating a Flash operation.
1 = Program or erase sequence did not complete successfully
0 = Program or erase sequence completed normally

bit 12 **LVDERR:** Low-Voltage Detect Error bit[2]

This bit can be cleared only by setting the NVMOP<3:0> bits = 0000 (NOP) and initiating a Flash operation.
1 = Low-voltage condition was detected during a program and erase operation (possible data corruption, if WRERR is set)
0 = No low-voltage condition occurred during a program or erase operation

bit 11-8 **Unimplemented:** Read as '0'

Note 1: This operation results in a "no operation" (NOP) when the Dynamic Flash ECC Configuration bits = 00 (FECCCTRL<1:0> (DVCFG0<9:8>)), which enables ECC at all times. For all other FECCTRL<1:0> bit settings, this command will execute, but will not write the ECC bits for the word and can cause DED errors if dynamic Flash ECC is enabled (FECCTRL<1:0> = 01).

2: This bit is only reset on a Power-on reset (POR) and are unaffected by other reset sources.

3: This bit is not available on all devices. Please refer to the **"Flash Program Memory"** chapter to determine which bits are available on your device.

Register 52-1:   NVMCON: Programming Control Register (Continued)

bit 7   **SWAP:** Program Flash Bank Swap Control bit**(3)**

This bit is only writable when WREN = 0 and the unlock sequence has been performed.

1 = Program Flash Bank 2 is mapped to the lower mapped region and program Flash Bank 1 is mapped to the upper mapped region

0 = Program Flash Bank 1 is mapped to the lower mapped region and program Flash Bank 2 is mapped to the upper mapped region

PFSWAP: Program Flash Bank Swap Control bit**(3)**

This bit is only writable when WREN = 0 and SWAPLOCK = 00 and the unlock sequence has been performed.

1 = Program Flash Bank 2 is mapped to the lower mapped region and program Flash Bank 1 is mapped to the upper mapped region

0 = Program Flash Bank 1 is mapped to the lower mapped region and program Flash Bank 2 is mapped to the upper mapped region

bit 6   **BFSWAP:** Boot Flash Bank Alias Swap Control bit**(3)**

This bit is only writable when WREN = 0 and SWAPLOCK = 00 and the unlock sequence has been performed.

The reset value of BFSWAP is determined by the comparison of the values of the user-programmed BFxSEQ0 word in each Boot Flash bank.

1 = Boot Flash Bank 2 is mapped to the lower boot alias and Boot Flash Bank 1 is mapped to the upper boot alias

0 = Boot Flash Bank 1 is mapped to the lower boot alias and Boot Flash Bank 2 is mapped to the upper boot alias

bit 5-4   **Unimplemented:** Read as '0'

bit 3-0   **NVMOP<3:0>:** NVM Operation bits

These bits are only writable when WREN = 0.

1111 = Reserved

•
•
•

1000 = Reserved

0111 = Program erase operation: erase all of program Flash memory (all pages must be unprotected, PWP<23:0> = 0x000000)

0110 = Upper program Flash memory erase operation: erases only the upper mapped region of program Flash (all pages in that region must be unprotected)

0101 = Lower program Flash memory erase operation: erases only the lower mapped region of program Flash (all pages in that region must be unprotected)

0100 = Page erase operation: erases page selected by NVMADDR, if it is not write-protected

0011 = Row program operation: programs row selected by NVMADDR, if it is not write-protected

0010 = Quad Word (128-bit) program operation: programs the 128-bit Flash Word selected by NVMADDR, if it is not write-protected

0001 = Word program operation: programs word selected by NVMADDR, if it is not write-protected**(1)**

0000 = No operation

Note 1:   This operation results in a "no operation" (NOP) when the Dynamic Flash ECC Configuration bits = 00 (FECCCTRL<1:0> (DVCFG0<9:8>)), which enables ECC at all times. For all other FECCTRL<1:0> bit settings, this command will execute, but will not write the ECC bits for the word and can cause DED errors if dynamic Flash ECC is enabled (FECCTRL<1:0> = 01).

2:   This bit is only reset on a Power-on reset (POR) and are unaffected by other reset sources.

3:   This bit is not available on all devices. Please refer to the **"Flash Program Memory"** chapter to determine which bits are available on your device.

Register 52-2: NVMKEY: Programming Unlock Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
	NVMKEY<31:24>							
23:16	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
	NVMKEY<23:16>							
15:8	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
	NVMKEY<15:8>							
7:0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
	NVMKEY<7:0>							

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

bit 31-0 **NVMKEY<31:0>:** Unlock Register bits
These bits are write-only, and read as '0' on any read.

Note: This register is used as part of the unlock sequence to prevent inadvertent writes to the PFM.

Register 52-3: NVMADDR: Flash Address Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMADDR<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMADDR<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMADDR<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMADDR<7:0>							

Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

bit 31-0 **NVMADDR<31:0>:** Flash Address bits

NVMOP<3:0> Selection (see Note 1)	Flash Address Bits
Page Erase	Address identifies the page to erase (lower address bits are ignored based on device page size)
Row Program	Address identifies the row to program (lower address bits are ignored based on device row size)
Word Program (32-bit)	Address identifies the word to program (NVMADDR<1:0> are ignored)
Quad Word Program (128-bit)	Address identifies the quad word (128-bit) to program (NVMADDR<3:0> bits are ignored).

Note 1: For all other NVMOP<3:0> bit settings, the Flash address is ignored.

Note: The bits in this register are only reset on a Power-on reset (POR) and are unaffected by other reset sources.

Register 52-4:   NVMDATAx: Flash Data Register ('x' = 0-3)

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMDATA<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMDATA<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMDATA<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMDATA<7:0>							

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

bit 31-0   **NVMDATA<31:0>:** Flash Data bits
Word Program: Writes NVMDATA0 to the target Flash address defined in NVMADDR
Quad Word Program: Writes NVMDATA3:NVMDATA2:NVMDATA1:NVMDATA0 to the target Flash address defined in NVMADDR. NVMDATA0 is the least significant instruction word.

Note: The bits in this register are only reset by a Power-on Reset (POR). Other reset sources do not affect its contents.

Register 52-5:   NVMSRCADDR: Source Data Address Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMSRCADDR<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMSRCADDR<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMSRCADDR<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMSRCADDR<7:0>							

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

bit 31-0   **NVMSRCADDR<31:0>:** Source Data Address bits
The system physical address of the data to be programmed into the Flash when the NVMOP<3:0> bits (NVMCON<3:0>) are set to perform row programming.

Note: The bits in this register are only reset by a Power-on Reset (POR). Other reset sources do not affect its contents.

Register 52-6:   NVMPWP: Program Flash Write-Protect Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	PWPULOCK	—	—	—	—	—	—	—
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PWP<23:16>							
15:8	R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
	PWP<15:8>							
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	PWP<7:0>							

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared x = Bit is unknown

bit 31   **PWPULOCK:** Program Flash Memory Page Write-protect Unlock bit

  1 = Register is not locked and can be modified
  0 = Register is locked and cannot be modified
  This bit is only clearable and cannot be set except by any reset.

bit 30-24 **Unimplemented:** Read as '0'

bit 23-0 **PWP<23:0>:** Flash Program Write-protect (Page) Address bits

  Physical program Flash memory from address 0x1D000000 to address 0x1Dxxxxxx is write protected, where 'xxxxxx' is specified by PWP<23:0>. When PWP<23:0> has a value of '0', write protection is disabled for the entire program Flash. If the specified address falls within the page, the entire page and all pages below the current page will be protected.

Note: This register is only writable when the NVMKEY unlock sequence has been followed and the PWPULOCK bit is set.

Register 52-7: NVMBWP: Flash Boot (Page) Write-Protect Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-1	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	LBWPU LOCK	—	—	LBWP4[1]	LBWP3[1]	LBWP2[1]	LBWP1[1]	LBWP0[1]
7:0	R/W-1	r-1	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	UBWPU LOCK[2]	—	—	UBWP4[2,3]	UBWP3[2,3]	UBWP2[2,3]	UBWP1[2,3]	UBWP0[2,3]

Legend:		r = Reserved	
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 31-16 **Unimplemented:** Read as '0'

bit 15 **LBWPULOCK:** Lower Boot Alias Write-protect Unlock bit
1 = LBWPx bits are not locked and can be modified
0 = LBWPx bits are locked and cannot be modified
This bit is only clearable and cannot be set except by any reset.

bit 14-13 **Unimplemented:** Read as '0'

bit 12 **LBWP4:** Lower Boot Alias Page 4 Write-protect bit[1]
1 = Write protection enabled
0 = Write protection disabled

bit 11 **LBWP3:** Lower Boot Alias Page 3 Write-protect bit[1]
1 = Write protection enabled
0 = Write protection disabled

bit 10 **LBWP2:** Lower Boot Alias Page 2 Write-protect bit[1]
1 = Write protection enabled
0 = Write protection disabled

bit 9 **LBWP1:** Lower Boot Alias Page 1 Write-protect bit[1]
1 = Write protection enabled
0 = Write protection disabled

bit 8 **LBWP0:** Lower Boot Alias Page 0 Write-protect bit[1]
1 = Write protection enabled
0 = Write protection disabled

bit 7 **UBWPULOCK:** Upper Boot Alias Write-protect Unlock bit[3]
1 = UBWPx bits are not locked and can be modified
0 = UBWPx bits are locked and cannot be modified

This bit is only user-clearable and cannot be set except by any reset.

bit 6 **Reserved:** This bit is reserved for use by development tools

Note 1: This bit is only writable when the NVMKEY unlock sequence has been followed and the LBWPULOCK bit is set.

2: This bit is only writable when the NVMKEY unlock sequence has been followed and the UBWPULOCK bit is set.

3: This bit is not available on all devices. Refer to the **"Flash Controller"** chapter in the specific device data sheet to determine availability.

Register 52-7:   NVMBWP: Flash Boot (Page) Write-Protect Register (Continued)

bit 5   **Unimplemented:** Read as '0'

bit 4   **UBWP4:** Upper Boot Alias Page 4 Write-protect bit[2,3]

1 = Write protection enabled
0 = Write protection disabled

bit 3   **UBWP3:** Upper Boot Alias Page 3 Write-protect bit[2,3]

1 = Write protection enabled
0 = Write protection disabled

bit 2   **UBWP2:** Upper Boot Alias Page 2 Write-protect bit[2,3]

1 = Write protection enabled
0 = Write protection disabled

bit 1   **UBWP1:** Upper Boot Alias Page 1 Write-protect bit[2,3]

1 = Write protection enabled
0 = Write protection disabled

bit 0   **UBWP0:** Upper Boot Alias Page 0 Write-protect bit[2,3]

1 = Write protection enabled
0 = Write protection disabled

Note 1:   This bit is only writable when the NVMKEY unlock sequence has been followed and the LBWPULOCK bit is set.

2:   This bit is only writable when the NVMKEY unlock sequence has been followed and the UBWPULOCK bit is set.

3:   This bit is not available on all devices. Refer to the **"Flash Controller"** chapter in the specific device data sheet to determine availability.

Register 52-8: NVMCON2: Flash Programming Control Register 2

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
	NVMERS<3:0>				—	—	—	—
23:16	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	—	—	—	NVMWS<4:0>				
15:8	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
	NVMLPRD	—	NVMCRD	NVMVRD	—	—	NVMRETRY<1:0>	
7:0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0
	SWAPLOCK<1:0>		—	—	—	—	—	—

Legend:		HC = Hardware Set	HC = Hardware Cleared
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 31-28 **NVMERS<3:0>:** Flash Memory Erase Retry State bits

These bits are used by software to track the software state of the erase-retry procedure in the event of a system reset (i.e., Master Clear) or Brown-out Reset event. The hardware does not use these bits.

bit 27-21 **Unimplemented:** Read as '0'

bit 20-16 **NVMWS<4:0>:** Flash Memory Access Wait State Control bits

These bits are effective when NVMLPRD = 1 or NVMVRD = 1 and can be modified only when the NVM-KEY unlock sequence is satisfied.

11111 = 31 Wait States (32 total SYSCLK cycles)
11110 = 30 Wait States (31 total SYSCLK cycles)
•
•
•
00010 = 2 Wait States (3 total SYSCLK cycles)
00001 = 1 Wait State (2 total SYSCLK cycles)
00000 = 0 Wait State (1 total SYSCLK cycles)

bit 15 **NVMLPRD:** Flash Memory Low Power Read Control bit

When NVMLPRD = 1, Flash Wait State control is handled by the NVMWS<4:0> bits. These bits can be modified only when the WR bit (NVMCON<15>) = 0 and the NVMKEY unlock sequence is satisfied.

1 = Configures Flash for low-power reads (increases access time)
0 = Configures Flash for low-latency reads

bit 14 **Unimplemented:** Read as '0'

bit 13 **NVMCRD:** Flash Memory Compare Read of Logic 1 control bit
When using Erase Retry in an ECC Flash system, NVMCRD must be enabled. These bits can be modified only when the NVMKEY unlock sequence is satisfied.

1 = Compare Read Enabled
0 = Compare Read Disabled

bit 12 **NVMVRD:** Flash Memory Verify Read of Logic 1 Control bit
When NVMVRD = 1, Flash wait state control is handled by the NVMWS<4:0> bits for the panel containing NVMADDR. These bits can be modified only when the WR bit (NVMCON<15>) = 0 and the NVMKEY unlock sequence is satisfied.

1 = Selects Erase Retry procedure with Verify Read
0 = Selects Single Erase without Verify Read

bit 11-10 **Unimplemented:** Read as '0'

> **Note:** This register is not available on all devices. Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine availability.

Register 52-8: NVMCON2: Flash Programming Control Register 2 (Continued)

bit 9-8 **NVMRETRY<1:0>:** Flash Memory Erase Retry control bits

These bits can be modified only when the WR bit (NVMCON<15>) = 0.

11 = Erase Strength for Last Retry Cycle
10 = Erase Strength for Third Retry Cycle
01 = Erase Strength for Second Retry Cycle
00 = Erase Strength for First Retry Cycle

bit 7-6 **SWAPLOCK<1:0>:** Flash Memory Swap Lock Control bits

11 = PFSWAP and BFSWAP are not writable and SWAPLOCK is not writable
10 = PFSWAP and BFSWAP are not writable and SWAPLOCK is writable
01 = PFSWAP and BFSWAP are not writable and SWAPLOCK is writable
00 = PFSWAP and BFSWAP are writable and SWAPLOCK is writable

bit 5-0 **Unimplemented:** Read as '0'

Note: This register is not available on all devices. Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine availability.

52.3   MEMORY CONFIGURATION

52.3.1   Flash Bank Construction

Each bank of Flash memory is divided into pages. A page is the smallest unit of memory that can be erased at one time. Each page of memory is segmented into eight rows. A row is the largest unit of memory that can be programmed at one time. A row consists of 128 Quad (128-bit) words. Each Quad word consists of four instruction (32-bit) words. Flash memory can be programmed in rows, Quad word (128-bit) or Word (32-bit) units.

> **Note:**   Page size varies by device. Please refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine the Flash page size for your device.

Figure 52-1:   Flash Construction
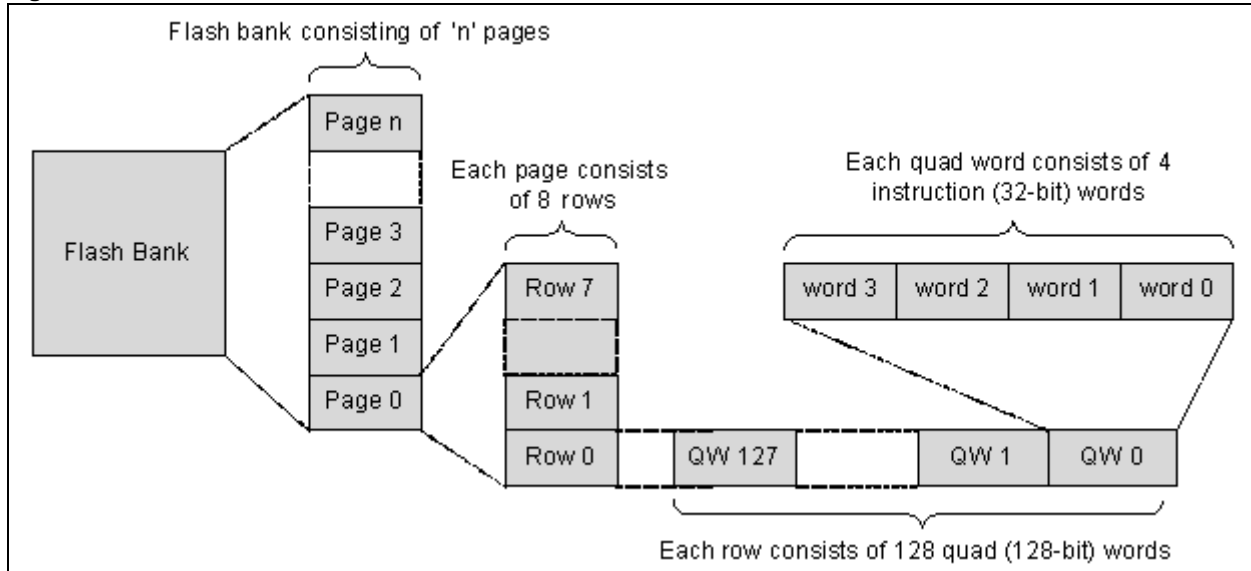
52.3.2   Dual Flash Banks

On certain PIC32 devices, Flash memory is divided into two equal banks, each with a BFM partition and a PFM partition.   Both the BFM and PFM partitions can be erased in page increments. The PFM partition(s) can be erased in their entirety with a single command.

52.3.3   Programming or Erasing Flash in the Same Bank Where Code is Executing

Code cannot be fetched by the CPU from the same Flash bank that is the target of the programming operation, which is either BFM or PFM. When this operation is attempted, the CPU will cease to execute code (stall) while the programming operation is in progress. This includes Interrupt Service Routines (and their vectors) when they are located in the same bank as the target Flash operation. If the system software requires execution of code during Flash operations, the code must reside in system RAM or the Flash bank that is not the target of the programming operation.

> **Note:**   Instruction code that is already stored in the cache when the programming operation is initiated will continue to execute.

52.3.4   Programming or Erasing Flash in the Opposite Bank Where Code is Executing

To avoid CPU stalling during programming operations, ensure that all target addresses for programming operations are located in a Flash bank where no executable code is being fetched. When this requirement is met, the CPU can fetch and execute code during any of the programming operations without stalling, including Interrupt Service Routines. To accomplish this, the application must place all executable code in the BFM and PFM partitions of one bank and execute programming operations on the BFM or PFM partitions of the other bank.

52.4 BOOT FLASH MEMORY (BFM) PARTITIONS

52.4.1 BFM Bank Aliasing

The BFM partitions of each Flash bank are mapped into fixed and aliased regions. The order of mapping to the aliased regions is determined by the device Configuration memory words BFxSEQ0 (BFxSEQ1 through BFxSEQ3 are not used). The mapping occurs at reset prior to execution of any instruction code. Whichever BFM has the larger value in the associated sequence word register, that bank will be mapped into the lower aliased region. If the values are equal, Bank 1 is mapped into the lower aliased region, Bank 2 and the other BFM is mapped into the upper aliased region.

When programming the BFxSEQ0 value, the sequence number value is stored in the lower 16 bits and the compliment of this value is stored in the upper 16 bits. As an example, using a sequence number of three, the BFxSEQ0 value would be 0xFFFC0003.

Figure 52-2 illustrates the effect of the BFxSEQ0 comparison. In either case, Bank 1 and Bank 2 BFM are mapped identically in the fixed regions, with Bank 1 in the lower fixed region, and Bank 2 in the upper fixed region. The mapping to the aliased regions changes based on the BFxSEQ0 values. This example shows a device with 80 KB BFM banks (see **Note**).

> **Note:** Refer to the **"Memory Organization"** chapter of the specific device data sheet to determine the size and address mapping for the BFM banks of your particular device.

The mapping of the Boot Flash into the aliased regions is indicated by the BFSWAP bit (NVMCON<6>). This bit is set or cleared on startup based on the mapping that takes place on Reset. It can also be controlled through software at run time. Note that the SWAPLOCK bits (NVMCON2<7:6>) and the WREN bit (NVMCON<14>) must be clear, and the NVMKEY unlock sequence executed before BFSWAP can be changed. It is recommended that this bit is only changed by the code executing from the PFM region.

For devices that have the NVMCON2 register, the control of the Boot Flash panel swapping can be locked by setting the SWAPLOCK bits (NVMCON2<7:6>) to any value other than '00' (see **Note**).

> **Note:** Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine availability of the NVMCON2 register.

Figure 52-2: Example of Aliased and Fixed BFM Regions Based on BFxSEQ0 Values

Note: Refer to the **"Special Features"** chapter in the specific device data sheet for the Configuration Word locations and the fixed and aliased region locations.
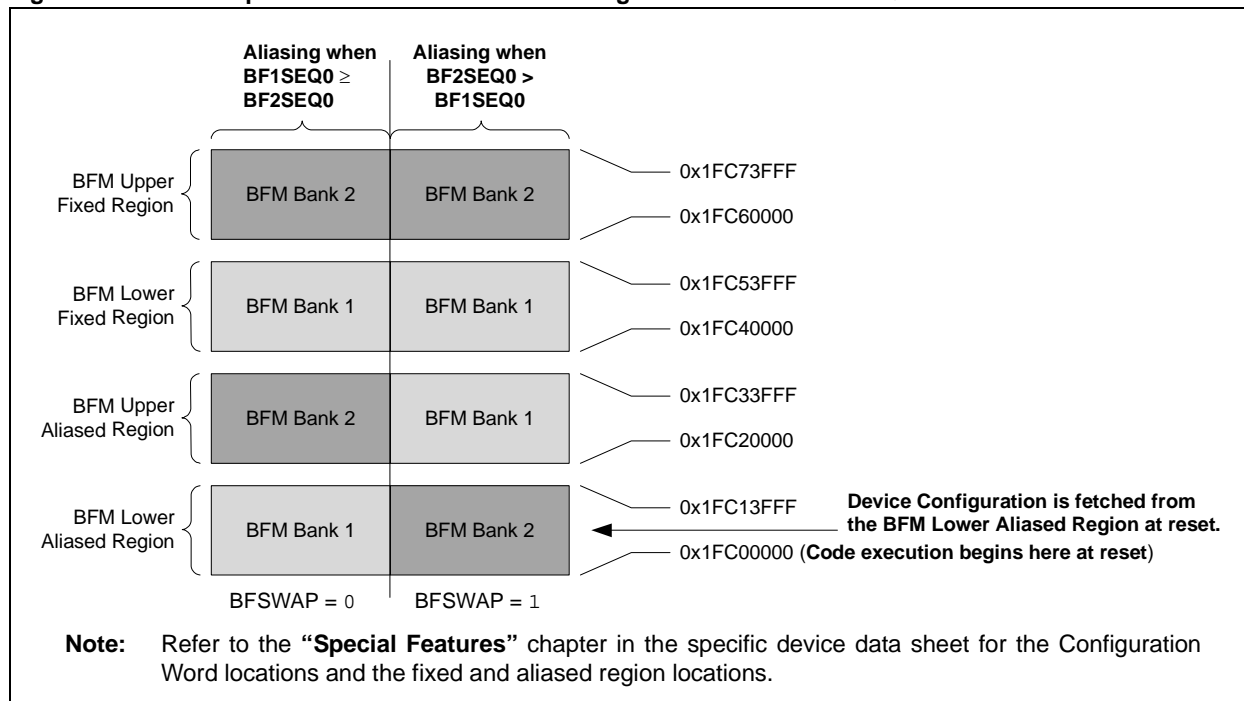
52.4.2   BFM Write Protection

Pages in the upper and lower aliased regions can be protected individually using bits in the NVMBWP register. Since these bits correspond to pages referenced in the aliased region, they are affected by the mapping of the aliasing at start-up. At reset, all pages are in a write-protected state and must be disabled prior to performing any programming operations on the BFM regions. There are also Upper Region and Lower Region unlock bits, UBWPULOCK (NVMBWP<7>) and LBWPULOCK (NVMBWP<15>), which are set at reset and can be cleared by user software. When cleared, changes to write protection for that region can no longer be made. Once cleared, the UBWPULOCK and LBWPULOCK bits can only be set by a reset.

The NVMBWP write-protect register can only be changed when the unlock sequence is followed. See **52.9 "NVMKEY Register Unlocking Sequence"** for more information.

52.5   PROGRAM FLASH MEMORY (PFM) PARTITIONS

52.5.1   PFM Mapping to Address Space

Each Flash bank has an equally sized PFM region. The mapping of the banks of PFM into address space is determined by the state of the SWAP (or PFSWAP) control bit (NVMCON<7>), as shown in Figure 52-3 (see **Note**). It is recommended that this bit is only changed by the code executing from the BFM region.

> **Note:**   Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine which bit, either SWAP or PFSWAP, exists on your device.

Figure 52-3:   Example of PFM Mapping Based on SWAP (or PFSWAP) Setting

> **Note:**   This example shows addresses for a 2 MB device. Refer to the **"Memory Organization"** chapter in the specific device data sheet for the actual addresses.

The SWAP (or PFSWAP) bit setting can only be changed when the unlock sequence is first completed. See **52.9 "NVMKEY Register Unlocking Sequence"** for more information.

For devices that have the NVMCON2 register, the control of the Program Flash panel swapping can be locked by setting the SWAPLOCK bits (NVMCON2<7:6>) (see **Note**) to any value other than '00'.

> **Note:**   Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine whether the NVMCON2 register exists on your device.

52.5.2   PFM Write Protection

Write protection for the PFM region is implemented by pages using a watermark address, defined by the NVMPWP<23:0> bits (NVMPWP<23:0>). Write protection is always implemented on page boundaries; therefore, for a device with a 0x4000 (16 KB) page size, address bits 0 through 13 are ignored and will always read '0'. When the NVMPWP is equal to '0', no PFM pages are write-protected. When not equal to '0', the NVMPWP bit will always point to the first address of the last page that is write-protected. All pages with lesser addresses will be write-protected. At reset no PFM memory is write-protected.

There is also an unlock bit, PFMULOCK (NVMPWP<31>), which is set at reset and can be cleared by user software. When cleared, changes to write protection of the PFM can no longer be made, including the PFMULOCK bit. The NVMPWP write-protect register can only be changed when the unlock sequence is followed. See **52.9 "NVMKEY Register Unlocking Sequence"** for more information.

52.6   ERROR CORRECTING CODE (ECC) AND FLASH PROGRAMMING

Some PIC32 devices incorporate Error Correcting Code (ECC) features, which detect and correct errors resulting in extended Flash memory life. This feature is explained in detail in **Section 41. "Prefetch Module for Devices with L1 CPU Cache"** (DS60001183).

ECC is implemented in 128-bit wide Flash words or four 32-bit instruction word groups. As a result, when programming Flash memory on a device where ECC is employed, the programming operation must be at minimum four instructions words or in groups of four instruction words. This is the reason that the Quad Word programming command exists and why row programming always programs multiples of four words.

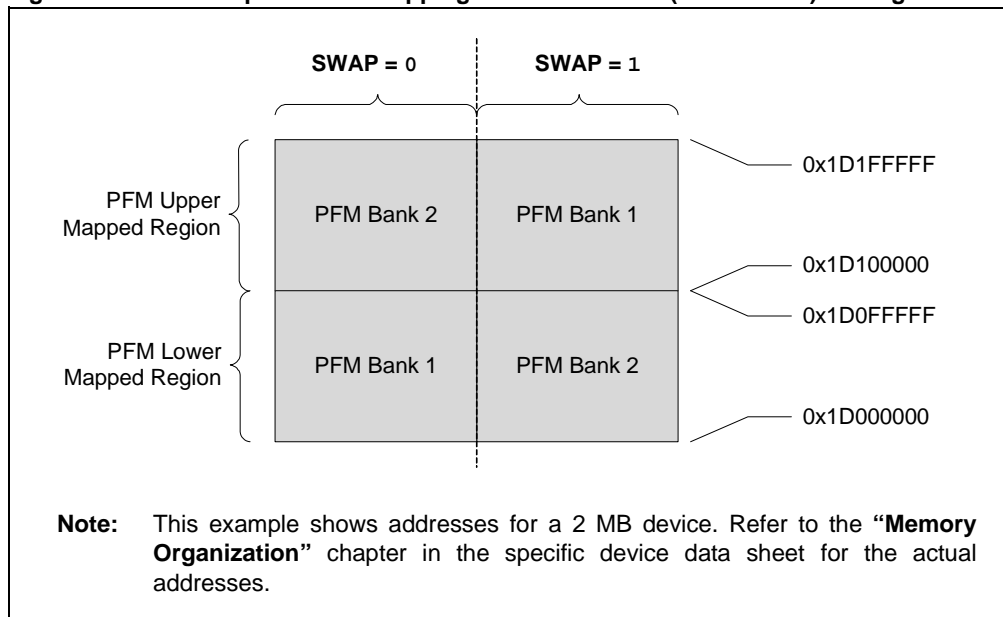For a given software application, ECC can be enabled at all times, disabled at all times, or dynamically enabled using the FECCCON Configuration bits. When ECC is enabled at all times, the Single Word NVMOP programming command does not function, and the quad word is the smallest unit of memory that can be programmed. When ECC is disabled or enabled dynamically, both the Word and Quad word programming NVMOP commands are functional and the programming method used determines how ECC is handled.

In the case of dynamic ECC, if the memory was programmed with the Word command, ECC is turned off for that word, and when it is read, no error correction is performed. If the memory was programmed with the Quad Word or Row Programming commands, ECC data is written and tested for errors (and corrected if needed) when read. Table 52-2 describes the different ECC scenarios.

Table 52-2:   ECC Programming Summary

FECCCON Setting	Programming Operation			Data Read
	Word Write	**Quad Word Write**	**Row Write**	**Data Read**
Disabled	Allowed	Allowed	Allowed	ECC is never applied on a Flash read
Enabled	Not allowed	Allowed	Allowed	ECC is applied on every Flash word read.
Dynamic	Allowed, but when used, the programmed word is flagged to NOT USE ECC	Writes ECC data and flags programmed words to USE ECC	Writes ECC data and flags programmed words to USE ECC	ECC is only applied on words that are flagged to USE ECC

Note: When using dynamic ECC, all non-ECC locations must be programmed with the 32-bit Word programming command, while all ECC enabled locations must be programmed with a 128-bit Quad Word or Row programming command. Divisions between ECC and non-ECC memory must be on even quad word boundaries (address bits 0 through 3 are equal to '0').

52.7   INTERRUPTS

An interrupt is generated when the WR bit is cleared by the Flash Controller upon completion of a Flash program or erase operation. The interrupt event will cause a CPU interrupt if it has been configured and enabled in the Interrupt Controller. The interrupt occurs regardless of the outcome of the program or erase operation; successful or unsuccessful. The only exception is the No Operation (NOP) programming operation (NVMOP = 0), which is used to manually clear the error flags and does not create an interrupt event on completion, but does clear the WR bit.

Flash Controller interrupts are not persistent, and therefore, no additional steps are required to clear the cause or source of the interrupt. It is only necessary to clear the appropriate IFSx bit prior to exiting the interrupt service routine.

Once the Interrupt Controller is configured, the Flash event will cause the CPU to jump to the vector assigned to the Flash event. The CPU will then begin executing code at the vector address. The user software at this vector address should perform the required operations, and then exit. For more information on interrupts, how to configure them, and the vector address table details, refer to the **Section 8. "Interrupts"** (DS60001108) in the *"PIC32 Family Reference Manual"* and the **"Interrupt Controller"** chapter in the specific device data sheet.

52.7.1   Interrupts and CPU Stalling

Code cannot be fetched by the CPU from the same Flash bank, either BFM or PFM, which is the target of the programming operation. When this operation is attempted, the CPU will cease to execute code (stall) while the programming operation is in progress. CPU code execution does not resume until the programming operation is complete, and when this occurs, any pending interrupts, including those from the Flash Controller, will be processed in order of priority.

Note that code that is already loaded into the processor cache will continue to execute up to the point where an attempt is made to fetch code or data from the same Flash panel as the active programming operation. At this point the CPU will stall.

In situations where all executable code and data constants are located in the bank opposite of the target of all programming operations, the Flash event interrupt allows the system to initiate a Flash operation, and then commence code execution and use the interrupt service routine to flag the completion of the programming operation. Using this technique, it is possible to design a system with background Flash programming functionality for live updates of Flash memory.

Stalling can also be avoided by placing any needed executable code in SRAM during Flash programming.

52.8 ERROR DETECTION

The NVMCON register includes two bits for detecting error conditions during a program or erase operation. They are Low-Voltage detect error, LVDERR bit (NVMCOM<12>) and Write Error, WRERR bit (NVMCON<13>).

The WRERR is set each time the WR bit (NVMCON<15>) is set, initiating a programming operation. WRERR is cleared on successful completion of the Flash operation and at the same time that WR is cleared and as such should not be polled until the Flash Controller clears WR. When the WRERR is set, any attempt to initiate programming or erase operation is ignored. WRERR must be cleared before commencing Flash program or erase operations.

The LVDERR bit is set when a Brown-out Reset (BOR) occurs during a programming operation. The only reset which clears the LVDERR bit is a Power-on Reset (POR). Other reset types do not affect the LVDERR bit. When the LVDERR bit is set, any attempt to initiate programming or erase operation is ignored. The LVDERR bit must be cleared before commencing Flash program or erase operations.

Both the WRERR and LVDERR bits can be cleared manually in software by initiating a Flash operation (setting WR) with NVMOP set to NOP (0x00). Note that executing the NVMOP NOP command clears WRERR, LVDERR and WR bits, but does not generate an interrupt event on completion.

Table 52-3: Programming Error Cause and Effects

Cause of Error	Effect on Programming Erase Operation	Indication
A low-voltage event occurred during a programming sequence.	The last programming or erase operation may not have completed.	LVDERR = 1, WRERR = 1
A non-POR reset occurred during programming.	Programming or erase operation is aborted.	WRERR = 1
Attempt to program or erase a page out of the Flash memory range.	Erase or programming operation is not initiated.	WRERR = 1
Attempt to erase or program a write protected PFM page.	Erase or programming operation is not initiated.	WRERR = 1
Attempt to erase or program a write protected BFM page.	Operation occurs, but the page is not programmed or erased.	WRERR = 0
Bus master error or row programming data underrun error during programming.	Programming or erase operation is aborted.	WRERR = 1

52.9 NVMKEY REGISTER UNLOCKING SEQUENCE

Important register settings that could compromise the Flash memory if inadvertently changed are protected by a register unlocking sequence. This feature is implemented using the NVMKEY register. The NVMKEY register is a write-only register that is used to implement an unlock sequence to help prevent accidental writes or erasures of Flash memory or accidental changes to the SWAP (or PFSWAP) bit (NVMCON<7>) and the BFSWAP bit (NVMCON<6>) and write permission settings (see **Note**).

> **Note:** Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine which bits exist on your device.

In some instances the operation is also dependent on the setting of the WREN bit (NVMCON<14>), as shown in Table 52-4.

Table 52-4: NVMKEY Register Unlocking and WREN

Operation	WREN Setting	SWAPLOCK Setting (see Note 1)	Unlock Sequence Required
Changing value of SWAP or PFSWAP (NVMCON<7>)	0	0 0	Yes
Changing value of BFSWAP (NVMCON<6>) (see **Note 1**)	0	0 0	Yes
Changing value of NVMOP<3:0> (NVMCON<3:0>)	0	Don't care	No
Setting WR (NVMCON<15>) to start a write or erase operation	1	Don't care	Yes
Changing any fields in the NVMPWP register	Don't care	Don't care	Yes
Changing any fields in the NVMBWP register	Don't care	Don't care	Yes

Note 1: This bit is not available on all devices. Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine which bits exist on your device.

The following steps must be followed in the exact order as shown to enable writes to registers that require this unlock sequence:

1. Write 0x00000000 to NVMKEY.
2. Write 0xAA996655 to NVMKEY.
3. Write 0x556699AA to NVMKEY.
4. Write the value to the register requiring the unlock sequence.

When using the unlock sequence to set or clear bits in the NVMCON register, as shown in Step 4, Steps 2 through 4 must be executed without any other activity on the peripheral bus that is in use by the Flash Controller. Interrupts and DMA transfers that access the same peripheral bus as the Flash Controller must be disabled. Refer to the specific device data sheet to determine which peripherals share the same peripheral bus as the Flash Controller. In addition, the operation in Step 4 must be atomic. The Set, Clear, and Invert registers may be used, where applicable, for the target register in Step 4.

Example 52-1 shows code written in the C language to initiate a NVM Operation (`NVMOP`) command. In this particular example, the WR bit is being set in the NVMCON register, and therefore, must include the unlock sequence. Note the use of the NVMCONSET register, which sets the WR bit in a single instruction without changing other bits in the register. Using `NVMCONbits.WR = 1` will fail, as this line of code compiles to a read-modify-write sequence.

Example 52-1: Initiate NVM Operation (Unlock Sequence Example)

```c
void NVMInitiateOperation(void)
{
    int    int_status;     // storage for current Interrupt Enable state
    int    dma_susp;       // storage for current DMA state

    // Disable Interrupts
    asm volatile("di   %0" : "=r"(int_status));

    // Disable DMA
    if(!(dma_susp=DMACONbits.SUSPEND))
    {
        DMACONSET=_DMACON_SUSPEND_MASK;      // suspend
        while((DMACONbits.DMABUSY));         // wait to be actually suspended
    }

    NVMKEY = 0x0;
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;
    NVMCONSET = 1 << 15;                     // must be an atomic instruction

    // Restore DMA
    if(!dma_susp)
    {
        DMACONCLR=_DMACON_SUSPEND_MASK;      // resume DMA activity
    }

    // Restore Interrupts
    if(int_status & 0x00000001)
    {
        asm volatile("ei");
    }
}
```

> **Note:** Once the unlock codes have been written to the NVMKEY register, the next activity on the same peripheral bus as the Flash Controller will reset the lock. As a result, only atomic operations can be used. Setting register fields using structures compile into a read-modify-write operation will fail.

52.10   WORD PROGRAMMING

The smallest block of data that can be programmed in a single operation is one 32-bit word. The data to be programmed must be written to the NVMDATA0 register, and the address of the word must be loaded into the NVMADDR register before the programming sequence is initiated. The instruction word at the physical location pointed to by the NVMADDR register is then programmed. Programming occurs on 32-bit word boundaries; therefore, bits 0 and 1 of the NVMADDR register are ignored.

Once a word is programmed, it must be erased before it can be programmed again, even if changing a bit from an erased '1' state to a '0' state.

Word programming will only succeed if the target address is in a page that is not write-protected. Programming to a write-protected PFM page will fail and result in the WRERR bit being set in the NVMCON register. Programming a write-protected BFM page will fail, but does not set the WRERR bit.

A programming sequence consists of the following steps:

1. Write 32-bit data to be programmed to the NVMDATA0 register.
2. Load the NVMADDR register with the address to be programmed.
3. Set the WREN bit = 1 and NVMOP bits = 1 in the NVMCON register. This defines and enables the programming operation.
4. Initiation of the programming operation (see **52.9 "NVMKEY Register Unlocking Sequence"**).
5. Monitor the WR bit of the NVMCON register to flag completion of the operation.
6. Clear the WREN bit in the NVMCON register.
7. Check for errors and process accordingly.

Example 52-2 shows code for Word programming, where a value of 0x12345678 is programmed into location 0x1D008000.

Example 52-2:   Word Programming

```
…

// Set up Address and Data Registers
   NVMADDR  = 0x1D008000;     // physical address
   NVMDATA0 = 0x12345678;     // value

// set the operation, assumes WREN = 0
   NVMCONbits.NVMOP = 0x1;    // NVMOP for Word programming

// Enable Flash for write operation and set the NVMOP
   NVMCONbits.WREN = 1;

// Start programming
   NVMInitiateOperation();    // see Example 52-1

// Wait for WR bit to clear
   while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
   NVMCONbits.WREN = 0;

// Check Error Status
   if(NVMCON & 0x3000)        // mask for WRERR and LVDERR
   {
       // process errors
   }
…
```

52.11 QUAD WORD PROGRAMMING

The process for Quad Word programming is identical to Word programming except that all four of the NVMDATAx registers are used. The value of the NVMDATA0 register is programmed at address NVMADDR, NVMDATA1 at NVMADDR + 0x4, NVMDATA2 at NVMADDR + 0x8, and NVMDATA3 at address NVMDATA + 0xC.

Quad Word programming is always performed on a quad word boundary; therefore, bits 3 through 0 are ignored.

Quad Word programming will only succeed if the target address is in a page that is not write-protected. Once a Quad Word is programmed, it must be erased before any word in it can be programmed again, even if changing a bit from an erased '1' state to a '0' state.

Example 52-3 shows code for Quad Word Programming where a value of 0x11111111 is programmed into location 0x1D008000, 0x22222222 into 0x1D008004, 0x33333333 into 0x1D008008, and 0x44444444 into location 0x1D00800C.

Example 52-3: Quad Word Programming

```
…

// Set up Address and Data Registers
   NVMADDR  = 0x1D008000;     // physical address
   NVMDATA0 = 0x11111111;     // value written to 0x1D008000
   NVMDATA1 = 0x22222222;     // value written to 0x1D008004
   NVMDATA2 = 0x33333333;     // value written to 0x1D008008
   NVMDATA3 = 0x44444444;     // value written to 0x1D00800C

// set the operation, assumes WREN = 0
   NVMCONbits.NVMOP = 0x2;    // NVMOP for Quad Word programming

// Enable Flash for write operation and set the NVMOP
   NVMCONbits.WREN = 1;

// Start programming
   NVMInitiateOperation();    // see Example 52-1

// Wait for WR bit to clear
   while(NVMCON & NVMCON_WR);

// Disable future Flash Write/Erase operations
   NVMCONbits.WREN = 0;

// Check Error Status
   if(NVMCON & 0x3000)        // mask for WRERR and LVDERR bits
   {
       // process errors
   }
…
```

52.12   ROW PROGRAMMING

The largest block of data that can be programmed is a row, which varies by device. Refer to the "**Flash Program Memory**" chapter in the specific device data sheet to determine the row size.

Unlike Word and Quad Word Programming, where the data source is stored in SFR memory, Row programming source data is stored in SRAM. The NVMSRCADDR register is a pointer to the physical location of the source data for Row programming.

Like other Non-volatile Memory (NVM) programming commands, the NVMADDR register points to the target address of the operation. Row programming always occurs on row boundaries; therefore, for a device with an instruction word row size of 512, bits 0 through 9 of the NVMADDR register are ignored.

Row Word programming will only succeed if the target address is in a page that is not write-protected. Once a row is programmed, it must be erased before any word in it can be programmed again, even if changing a bit from an erased '1' state to a '0' state.

Example 52-4 shows code for Row programming. Array `rowbuff` is populated with data and programmed into a row located at physical address 0x1D008000.

> **Note:**   When assigning the value to the NVMSRCADDR register, it must be converted to a physical address.

In Word and Quad word programming the data is stored in SFRs.
But in Row programming, the data is stored in SRAM
NVMSRCADDR register is the pointer to the location for Row.

Example 52-4:   Row Programming

```
…

unsigned int rowbuff[512]; // example is for a 512 word row size.
int x;                     // loop counter

// put some data in the source buffer
    for (x = 0; x < (sizeof(rowbuff) * sizeof (int)); x++)
        ((char *)rowbuff)[x] = x;

// set destination row address
    NVMADDR = 0x1D008000;      // row physical address

// set source address. Must be converted to a physical address.
    NVMSRCADDR = (unsigned int)((int)rowbuff & 0x1FFFFFFF);

// define Flash operation
    NVMCONbits.NVMOP = 0x3;    // NVMOP for Row programming

// Enable Flash Write
    NVMCONbits.WREN = 1;

// commence programming
    NVMInitiateOperation();    // see Example 52-1

// Wait for WR bit to clear
    while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)        // mask for WRERR and LVDERR bits
    {
        // process errors
    }
…
```

*is this an error ?? this goes to 2048*4 = 8192 which overflows the array, so it should just include sizeof(rowbuff) in the limit. ??*

Virtual address converted to physical address .

This bit is set in the NVMInitiateOperation() function, when this bit is set the actual writing on the flash memory starts.

52.13 PAGE ERASE

A page erase performs an erase of a single page of either PFM or BFM. Refer to the **"Flash Program Memory"** chapter in the specific device data sheet for the page size for your device.

The page to be erased is selected using the NVMADDR register. Pages are always erased on page boundaries; therefore, for a device with an instruction word page size of 4096, bits 0 through 11 of the NVMADDR register are ignored.

A page erase will only succeed if the target address is a page that is not write-protected. Erasing a write-protected page will fail and result in the WRERR bit being set in the NVMCON register.

Example 52-5 shows code for a single page erase operation at address 0x1D008000.

Example 52-5: Page Erase

```
…

// set destination page address
    NVMADDR = 0x1D008000;      // page physical address

// define Flash operation
    NVMCONbits.NVMOP = 0x4;    // NVMOP for Page Erase

// Enable Flash Write
    NVMCONbits.WREN = 1;

// commence programming
    NVMInitiateOperation();    // see Example 52-1

// Wait for WR bit to clear
    while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)        // mask for WRERR and LVDERR bits
    {
        // process errors
    }
…
```

52.13.1 Page Erase Retry

For PIC32 devices that support it, Erase Retry is a method to improve the life of a Flash panel by attempting the erase again if it didn't work. Erase Retry can only be used for a Page Erase.

Erase Retry works by increasing the voltage used on the Flash when erasing. Initially, the minimum voltage necessary is applied by setting the NVMRETRY<1:0> bits (NVMCON2<9:8>) = 00. If the page erase does not work, the voltage may be increased by incrementing the setting of the NVMRETRY<1:0> bits. Note that each Flash page, as it ages and wears, may have different voltage requirements, so a higher setting on one Flash page does not indicate that the same setting should be used on all pages. The maximum voltage for page erase is used when the NVMRETRY<1:0> bits = 11. If page erase does not work when the NVMRETRY<1:0> bits = 11, this means that the Flash for that page, or the words that did not erase, should be considered "non-functional".

Together with the normal Page Erase controls, Erase Retry also uses the NVMWS<4:0>, NVMLPRD, NVMCRD, NVMVRD, and NVMRETRY<1:0> bits in the NVMCON2 register. The NVMERS<3:0> bits (NVMCON2<31:28>) are for the benefit of software performing the programming sequence in the event that a drop in power causes a BOR event, but not a POR event.

Perform the following steps to set up a Page Erase Retry:

1. Set the NVMADDR register with the address of the page to be erased.
2. Execute the write unlock sequence.
3. Save the value of the NVMCON2 register.
4. Do the following in the NVMCON2 register:
    a) Set the NVMERS<3:0> bits as desired.
    b) Set the NVMWS<4:0> bits per the description.
    c) Set the NVMVRD bit to '1'.
    d) Set the NVMCRD bit to '1'.
    e) Set the NVMRETRY<1:0> bits to '00'.
5. Run the unlock sequence using the Page Erase command to start the sequence.
6. Wait for the WR bit (NVMCON<15>) to be cleared by hardware.
7. Clear the WREN bit (NVMCON<14>)
8. Verify the erase using the CPU

    To shorten the verify time, use NVMCRD = 1 to perform a hardware compare to logic 1 of each bit in the Flash Word including ECC (if ECC is provided). NVMCRD = 1 only affects the panel containing NVMADDR. A successful compare yields a read of 0x00000001 in the lowest addressed word in a Flash Word (128 bits). This is the Compare Word. All other words are 0x00010000. If any bit is logic 0, all Words in the Flash Word read 0x00000000. Remember to increment the address by the number of bytes in a Flash Word between reads.

9. If all Compare Words verify correctly, the Erase Retry process is complete. Go to step 11.
10. If a Compare Word yields a read of 0x00000000, perform steps 4 through 9 up to three more times with the following change to step 4:
    a) Increment the NVMRETRY<1:0> bits by one.
    b) Maintain all other fields.
11. Restore the value of the NVMCON2 register, which was saved in step 3.

> **Note 1:** When the NVMVRD = 1, the Flash uses the NVMWS<3:0> bits for Flash access wait state generation to the panel selected by NVMADDR. Software is responsible for writing the NVMVRD bit back to '0' when both erase and verify is complete.
>
> **2:** Boot Page 3 (the page containing the DEVCFGx values) does not support Erase Retry.

Example 52-6 provides code for a single page erase operation at address 0x1D008000 where Erase Retry is used.

Example 52-6:   Page Erase Retry

```
uint32_tsaveNVMCON2;
uint32_t*cmpPtr;
uint8_t erased;
uint8_t tryCount;

// set destination page address
    NVMADDR = 0x1D008000;      // Page physical address

// define flash operation
    NVMCONbits.NVMOP = 0x4;    // NVMOP for Page Erase

// Unlock sequence
    NVMKEY = 0x0;
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;

// save NVMCON2
    saveNVMCON2 = NVMCON2;

// set up Erase Retry
    NVMCON2bits.NVMERS = 0;    // Stage 0 - SW use only
    NVMCON2bits.NVMVRD = 1;
    NVMCON2bits.NVMCRD = 1;
    NVMCON2bits.NVMRETRY = 0b00;

    tryCount = 0;              // Up to 4 attempts

    do {
        tryCount++;

        // commence programming
        NVMInitiateOperation();

        // Wait for WR bit to clear
        while(NVMCONbits.WR);

        // Turn off WREN
        NVMCONbits.WREN = 0;

        // Check that the page was erased
        erased = 1;
        cmpPtr = (uint32_t *)NVMADDR;
        erased &= (*cmpPtr == 0x00000001);
        cmpPtr++;
        erased &= (*cmpPtr == 0x00010000);
        cmpPtr++;
        erased &= (*cmpPtr == 0x00010000);
        cmpPtr++;
        erased &= (*cmpPtr == 0x00010000);

        if (!erased) {
            // Erase failed. Try with different settings.
            NVMCON2bits.NVMRETRY++;

            NVMCONbits.NVMOP = 0x4;
            NVMCONbits.WREN = 1;
        }
    } while (!erased && (tryCount < 4));

// Restore settings
    NVMCON2 = saveNVMCON2;
```

52.14   PROGRAM FLASH MEMORY ERASE

Program Flash memory can be erased in its entirety or by bank. Three discreet NVMOP commands are implemented to accomplish this:

- Erase the entire PFM area (both PFM banks)
- Erase the upper PFM bank
- Erase the lower PFM bank

When erasing the entire PFM area, code must be executing from BFM. When erasing a single upper or lower PFM bank, code must either be executing from BFM or from the PFM bank that is not being erased.

Program Flash memory erase operations will only succeed if no pages are write-protected in the bank being erased. When erasing the entire PFM area, PFM write protection must be completely disabled.

Example 52-7 shows code for erasing the upper program Flash bank.

Example 52-7:   Program Flash Erase

```
…

// define Flash operation
   NVMCONbits.NVMOP = 0x6;        // NVMOP for upper bank PFM erase

// Enable Flash Write
   NVMCONbits.WREN = 1;

// commence programming
   NVMInitiateOperation();        // see Example 52-1

// Wait for WR bit to clear
   while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
   NVMCONbits.WREN = 0;

// Check Error Status
   if(NVMCON & 0x3000)            // mask for WRERR and LVDERR bits
   {
       // process errors
   }
…
```

52.15   OPERATION IN POWER-SAVING MODES

The Flash Controller does not operate in power-saving modes. If a `WAIT` instruction is encountered when programming, the CPU will stop execution (stall), wait for the programming operation to complete, and then enter the Power-Saving mode.

52.16   OPERATION IN DEBUG MODE

Programming operations will continue to completion if processor execution is halted in Debug mode.

52.17   EFFECTS OF VARIOUS RESETS

Devices Resets other than a Power-on Reset (POR), reset the SWAP (or PFSWAP) bit in the NVMCON register and the entire contents of the NVMPWP and NVMBWP registers (see **Note**). All other register content persists through a non-POR reset.

> **Note:**   Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine which bit, either SWAP or PFSWAP, exists on your device.

All Flash Controller registers are forced to their reset states upon a POR.

52.18 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC32 device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to Flash Memory with Support for Live Update include the following:

Title	Application Note #
No related application notes at this time.	N/A

> **Note:** Please visit the Microchip Web site (www.microchip.com) for additional application notes and code examples for the PIC32 family of devices.

52.19   REVISION HISTORY

Revision A (May 2013)

This is the initial released version of the document.

Revision B (July 2015)

This revision includes the following updates:

- The BOOTSWAP bit in the NVMCON register was renamed to: BFSWAP (see Register 52-1)
- The NVMCON2 register was added (see Table 52-1 and Register 52-8)
- **52.4.1 "BFM Bank Aliasing"** was updated
- **52.5.1 "PFM Mapping to Address Space"** was updated
- **52.9 "NVMKEY Register Unlocking Sequence"** was updated
- **52.13.1 "Page Erase Retry"** was added
- All code examples were updated (see Example 52-1 though Example 52-7)
- In addition, minor updates to text and formatting were incorporated throughout the document

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.

- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.

- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.

- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/
support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Dusseldorf
Tel: 49-2129-3766400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Venice
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Poland - Warsaw
Tel: 48-22-3325737

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

07/14/15