Section 5. Flash Program Memory

This section of the manual contains the following topics:

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 PIC32MZ W1 family of devices. Please refer to 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 website at: http://www.microchip.com.

5.1 INTRODUCTION

This document describes techniques for programming the Flash memory on PIC32MZ W1 devices. These devices contain a bank of Flash memory with a Boot Flash Memory (BFM) partition and a Program Flash Memory (PFM) partition for storing user code or nonvolatile data. 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 Microchip's proprietary 2-wire serial data connection of the device. This method allows an external device, such as a programmer and/or debugger, to alter the contents of the internal BFM/PFM.
- Joint Test Action Group Programming (JTAG) – Performed by accessing the device's internal debug/program module, referred to as the Enhanced JTAG (EJTAG) logic. This access to the program/debug is done through an IEEE compliant JTAG port (TCK, TMS, TDI and TDO). Using the debug/program logic, allows an external device to alter the contents of the internal BFM/PFM.

The access to the EJTAG is done through a physical JTAG TAP controller. RTSP techniques are described in **5.1.1 "Run-Time Self-Programming (RTSP)"**. The ICSP and JTAG methods are described in the *"PIC32 Flash Programming Specification"* (DS60001145), which is available for download from the Microchip website (www.microchip.com).

5.1.1 Run-Time Self-Programming (RTSP)

Run-Time Self-Programming (RTSP) describes the internal CPU running user firmware, which has been designed to allow altering (programming and/or erasing) the device's flash contents. In general, this RTSP method is used by applications that require a field upgrade mechanism, such as a Bootloader or Over-the-Air (OTA) update applications.

5.1.2 Addressing

PIC32MZ W1 internal CPU uses two address schemes: virtual and physical. Virtual addresses are exclusively used by the CPU to fetch and execute instructions, as well as, to access peripherals. Due to the design of the Flash controller's internal logic, when programming or erasing Flash memory, the physical addresses are always used for target operations (see Register 5-4).

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.

5.2 CONTROL REGISTERS

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

- **NVMCON: Programming Control Register**

  This 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 the status of the result when the operation is complete.

- **NVMCON2: Programming Control2 Register**

  This register is the control and status register for Flash program/erase operations.

- **NVMKEY: Programming Unlock Register**

  This is a write-only register that is used to implement an unlock sequence to help prevent accidental writes/erasures of Flash and write permission settings.

- **NVMADDR: Flash Address Register**

  This register is used to store the physical target address for row, Quad Double Word and Single Double Word programming, as well as, page erasing.

- **NVMDATAx: Flash Program Data Register (x = 0-7)**

  These registers hold the data to be programmed during Flash Word program operations.

- **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.

- **NVMPWPLT: Flash Program Write Protect Less Than Register**

  This register is used to set the program Flash pages lower than the provided address as write protected.

- **NVMPWPGTE: Flash Program Write Protect Greater Than Register**

  This register is used to set the program Flash pages greater than the provided address as write protected.

- **NVMLBWP: Flash Lower Boot Write Protect Register**

  This register is used to set the boot Flash lower partition pages as write-protected.

- **NVMUBWP: Flash Upper Boot Write Protect Register**

  This register is used to set the boot Flash upper partition pages as write-protected.

Table 5-1 provides a brief summary of all of the Flash-programming-related registers. Corresponding registers appear after the summary, followed by a detailed description.

Table 5-1: Flash Controller Register Map

Register Name	Bit Range	Bits															
		31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
NVMCON(1)	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15:00	WR	WREN	WRERR	LVDERR	—	—	—	HTDPGM	—	—	—	—	NVMOP[3:0]			
NVMCON2	31:16	ERS[3:0]				—	—	—	SLEEP	—	—	—	WS[4:0]				
	15:00	—	TEMP	CREAD1	VREAD1	—	—	RETRY[1:0]		—	—	—	—	—	—	—	—
NVMKEY	31:16	NVMKEY[31:0]															
	15:00																
NVMADDR(1)	31:16	NVMADDR[31:0]															
	15:00																
NVMDATA0	31:16	NVMDATA0[31:0]															
	15:00																
NVMDATA1	31:16	NVMDATA1[31:0]															
	15:00																
NVMDATA2	31:16	NVMDATA2[31:0]															
	15:00																
NVMDATA3	31:16	NVMDATA3[31:0]															
	15:00																
NVMDATA4	31:16	NVMDATA4[31:0]															
	15:00																
NVMDATA5	31:16	NVMDATA5[31:0]															
	15:00																
NVMDATA6	31:16	NVMDATA6[31:0]															
	15:00																
NVMDATA7	31:16	NVMDATA7[31:0]															
	15:00																
NVMSRC ADDR	31:16	NVMSRCADDR[31:0]															
	15:00																
NVMPWPLT	31:16	ULOCK	—	—	—	—	—	—	—	PWPLT[23:16]							
	15:00	PWPLT[15:0]															
NVMPWPGTE	31:16	ULOCK	—	—	—	—	—	—	—	PGTE[23:16]							
	15:00	PGTE[15:0]															
NVMLBWP(1)	31:16	ULOCK	—	—	—	—	—	—	LBWP23	LBWP22	LBWP21	LBWP20	LBWP19	LBWP18	LBWP18	LBWP17	LBWP16
	15:00	LBWP15	LBWP14	LBWP13	LBWP12	LBWP11	LBWP10	LBWP9	LBWP8	LBWP7	LBWP6	LBWP5	LBWP4	LBWP3	LBWP2	LBWP1	LBWP0
NVMUBWP(1)	31:16	ULOCK	—	—	—	—	—	—	UBWP23	UBWP22	LBWP21	UBWP20	UBWP19	UBWP18	UBWP18	UBWP17	UBWP16
	15:00	UBWP15	UBWP14	UBWP13	UBWP12	UBWP11	UBWP10	UBWP9	UBWP8	UBWP7	UBWP6	UBWP5	UBWP4	UBWP3	UBWP2	UBWP1	UBWP0

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 must be ignored.

Register 5-1: 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/HS/HC-0	R/W-0	R/HS/HC-0	R/HS/HC-0	U-0	U-0	U-0	R/HS/HC-0
	WR(1)	WREN(1)	WRERR(1)	LVDERR(1)	—	—	—	HTDPGM
7:0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	—	—	NVMOP[3:0]			

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

bit 15 **WR**: Write Control Bit(1)
1 = Initiate a Flash operation. Hardware clears this bit when the operation completes.
0 = Flash operation complete or inactive

Note: This field can only be modified when WREN = 1, TEMP = 1 and the NVMKEY unlock sequence is satisfied.

bit 14 **WREN**: Write Enable Bit(1)
1 = Enables write to WR
0 = Disables write to WR

bit 13 **WRERR**: Write Error Bit(1)
1 = Program or erase sequence does not complete successfully
0 = Program or erase sequence completed normally

Note: Cleared by setting NVMOP == 0000b and initiating a Flash operation (WR).

bit 12 **LVDERR**: Low Voltage Detect Error Bit(1)
The error is only captured for programming/erase operations (when WR = 1).
1 = Low voltage is detected (possible data corruption if WRERR is set)
0 = Normal voltage is detected

Note: Cleared by setting NVMOP == 0000b and initiating a Flash operation (WR).

bit 11-9 **Unimplemented:** Read as '0'

bit 8 **HTDPGM**: High Temperature Detected during Program/Erase Operation bit
This status is only captured for programming/erase operations (when WR = 1).
1 = High temperature is detected (possible data corruption, verify operation)
0 = High temperature is not detected

Note: Cleared by setting NVMOP == 0000b and initiating a Flash operation (WR)

bit 7-4 **Unimplemented:** Read as '0'

Note 1: These bits are only reset by a POR and are not affected by other Reset sources.

2: This operation results in a No Operation (NOP) when the Dynamic Flash ECC Configuration bits = 00 (FECCCON[1:0] (CFGCON0[29:28])), which enables ECC at all times. For all other FECCCON[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 (FECCCON[1:0] = 01).

Register 5-1: NVMCON: Programming Control Register (Continued)

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)
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 Double Word (256-bit) program operation: programs the 256-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**(2)**
0000 = No operation

Note 1: These bits are only reset by a POR and are not affected by other Reset sources.

2: This operation results in a No Operation (NOP) when the Dynamic Flash ECC Configuration bits = 00 (FECCCON[1:0] (CFGCON0[29:28])), which enables ECC at all times. For all other FECCCON[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 (FECCCON[1:0] = 01).

Register 5-2: NVMCON2: Programming Control2 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	U-0	U-0	U-0	R/W-1
	ERS[3:0]				—	—	—	SLEEP
23:16	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	—	—	—	WS[4:0]				
15:8	U-0	R/W-cfg	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
	—	TEMP	CREAD1	VREAD1	—	—	RETRY[1:0]	
7:0	U-0(1)	U-0(1)	U-0(1)	U-0(1)	U-0(1)	U-0(1)	U-0(1)	U-0(1)
	—	—	—	—	—	—	—	—

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 **ERS[3:0]**: Erase Retry State

These bits are used by software to track the software state of the erase retry procedure in the event of a system Reset (MCLR) or brown-out Reset event.

bit 27-25 **Unimplemented:** Read as '0'

bit 24 **SLEEP**: Power Down in Sleep bit
1 = Configures Flash for power down when the system is in Sleep mode
0 = Configures Flash for standby when the system is in Sleep mode

Note: This field can only be modified when the NVMKEY unlock sequence is satisfied.

bit 23-21 **Unimplemented:** Read as '0'

bit 20-16 **WS[4:0]**: Flash Access Wait State Control for VREAD1 = 1
11111 = 31 wait states (32 total system clocks)
11110 = 30 wait states (31 total system clocks)
...
00010 = 2 wait states (3 total system clocks)
00001 = 1 wait state (2 total system clocks)
00000 = 0 wait state (1 total system clock)

Note 1: When VREAD1 = 1, WS[] only affects the panel containing NVMADDR[].

2: This field can only be modified when the NVMKEY unlock sequence is satisfied.

bit 15 **Unimplemented:** Read as '0'

bit 14 **TEMP**: Operating Temperature Control bit
1 = Configures Flash for standard temperature, low latency reads
0 = Configures Flash for high temperature, high latency reads

Note 1: When TEMP = 0, all NVMOP Operations are disabled because NVMWR cannot be written to '1'.

2: When TEMP = 0, firmware must adjust the Flash wait state control at the system level.

3: This field can only be modified when NVMCON.WR == 0 and the NVMKEY unlock sequence is satisfied.

Note 1: User must set these bits to 0. Writing 1 to these bits will cause unintended device operation.

Register 5-2: NVMCON2: Programming Control2 Register (Continued)

bit 13 **CREAD1**: Compare Read of Logic 1 bit

Compare read 1 causes all bits in a Flash Word (including ECC if it exists) to be evaluated during the read. If all bits are 1, the lowest word in the Flash Word evaluates to 0x0000_0001, all other words are 0x0001_0000. If any bit is 0, the read evaluates to 0x0000_0000 for all words in the Flash Word.

1 = Compare read enabled, only if VREAD1 = 1
0 = Compare read disabled

Note 1: When using erase retry in an ECC Flash system, CREAD1 = 1 must be used.

2: This field can only be modified when the NVMKEY unlock sequence is satisfied.

bit 12 **VREAD1**: Verify Read of logic 1 Control bit
1 = Selects erase retry procedure with verify read
0 = Selects single erase without verify read

Note 1: When VREAD1 = 1, the Flash wait state control derives from WS[] for the panel containing NVMADDR[].

2: Using the erase retry and verify read procedures increase the life of the Flash panel(s).

3: This field can only be modified when NVMCON.WR == 0 and the NVMKEY unlock sequence is satisfied.

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

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

bit 9-8 **RETRY[1:0]**: Erase Retry Control bit, only used when VREAD1 = 1
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

Note: This field can only be modified when NVMCON.WR == 0.

bit 7-0 **Unimplemented:** Read as '0'(1)

Note 1: User must set these bits to 0. Writing 1 to these bits will cause unintended device operation.

Register 5-3: 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 '0' on any read.

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

Register 5-4: 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 (Word) Address bits

NVMOP[3:0] Selection	Flash Address Bits (NVMADDR[31:0])
Page erase	Address identifies the page to erase. Any address within a 4-Kbytes page boundary will cause the page to be erased.
Row program	Address identifies the row to program. The value of the address must be aligned to a row boundary.
Double Word program	Address identifies the 64-bit DWord to program. NVMADDR[2:0] bits are ignored. Must be aligned to a DWORD boundary.
Quad Double Word program	Address identifies the 256-bit Quad DWord to program. NVMADDR[4:0] bits are ignored. Must be aligned to a Quad DWORD boundary.

Note 1: Hardware prevents writes to this register when NVMCON.WR = 1.

2: For all other NVMOP[3:0] bit settings, the Flash address is ignored. See the NVMCON register (Register 5-1) for additional information on these bits.

Note: The values written to this register must be the physical addresses not the virtual address.

Register 5-5: NVMDATAx: Flash Program Data Register (x = 0-7)

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 **NVMDATAx[31:0]:** Flash Programming Data bits
The value in this register is written to Flash when a program operation is commanded.

Single Double Word program (64-bit)
Writes NVMDATA1:NVMDATA0 to the target Flash address defined in NVMADDR.

Quad Double Word program (256-bit)
Writes the values in registers NVMDATA7 - NVMDATA0 starting at the target Flash address defined in NVMADDR. NVMDATA0 register contains the Least Significant Flash Write Word.

Note: Hardware prevents writes to this register when NVMCON.WR = 1.

Register 5-6: 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 (Word) Address bits

This is the system physical word address of the data (in DRM) to be programmed into the Flash when NVMCON.NVMOP is set to row programming.

Note: Hardware prevents writes to this register when NVMCON.WR = 1.

Register 5-7: NVMPWPLT: Flash Program Write Protect Less Than 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/C-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	ULOCK	—	—	—	—	—	—	—
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
	PWPLT[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
	PWPLT[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
	PWPLT[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 **ULOCK**: NVMPWPLT Register Unlock bit
1 = NVMPWPLT register is not locked and can be modified
0 = NVMPWPLT register is locked and cannot be modified

Note 1: This field can only be modified when the NVMKEY unlock sequence is satisfied.

2: This field can be cleared at the same time as writing to PWPLT[23:0].

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

bit 23-0 **PWPLT[23:0]**: Flash Program Write Protect Less Than Address
Pages at Flash addresses less than this value are write-protected.

Note 1: This field can only be modified when the NVMKEY unlock sequence is satisfied and ULOCK = 1.

2: This is a byte address forced to align to page boundaries.

Register 5-8: NVMPWPGTE: Flash Program Write Protect Greater Than 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/C-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	ULOCK	—	—	—	—	—	—	—
23:16	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	PWPGTE[23:16]							
15:8	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	PWPGTE[15:8]							
7:0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	PWPGTE[7:0]							

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 **ULOCK**: NVMPWPGTE Register Unlock bit
1 = NVMPWPGTE register is not locked and can be modified
0 = NVMPWPGTE register is locked and cannot be modified

Note 1: This field can only be modified when the NVMKEY unlock sequence is satisfied.

2: This field can be cleared at the same time as writing to PWPGTE[23:0].

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

bit 23-0 **PWPGTE[23:0]**: Flash Program Write Protect Address
Pages at Flash addresses greater than or equal to this value are write-protected.

Note 1: This field can only be modified when the NVMKEY unlock sequence is satisfied and ULOCK = 1.

2: This is a byte address forced to align to page boundaries.

Register 5-9: NVMLBWP: Flash Lower Boot 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/C-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	ULOCK	—	—	—	—	—	—	—
23:16	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	LBWP23	LBWP22	LBWP21	LBWP20	LBWP19	LBWP18	LBWP17	LBWP16
15:8	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	LBWP15	LBWP14	LBWP13	LBWP12	LBWP11	LBWP10	LBWP9	LBWP8
7:0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	LBWP7	LBWP6	LBWP5	LBWP4	LBWP3	LBWP2	LBWP1	LBWP0

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 **ULOCK**: Lower Boot Write Protect (LBWPn) Unlock bit
1 = LBWPn bits are not locked and can be modified
0 = LBWPn bits are locked and cannot be modified

Note 1: This field can only be modified when the NVMKEY unlock sequence is satisfied.

2: This field can be cleared at the same time as writing to LBWP[msb:lsb].

Register 5-9: NVMLBWP: Flash Lower Boot Write Protect Register

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

bit 23-0 **LBWP[23:0]**: Lower Boot Pages Write Protect bits
LBWP[n] = 1: Erase and write protection for upper boot page n is enabled
LBWP[n] = 0: Erase and write protection for upper boot page n is disabled

Note: This field can only be modified when the NVMKEY unlock sequence is satisfied and ULOCK = 1.

Register 5-10: NVMUBWP: Flash Upper Boot 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/C-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	ULOCK	—	—	—	—	—	—	—
23:16	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	UBWP23	UBWP22	UBWP21	UBWP20	UBWP19	UBWP18	UBWP17	UBWP16
15:8	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	UBWP15	UBWP14	UBWP13	UBWP12	UBWP11	UBWP10	UBWP9	UBWP8
7:0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	UBWP7	UBWP6	UBWP5	UBWP4	UBWP3	UBWP2	UBWP1	UBWP0

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 **ULOCK**: Upper Boot Write Protect (UBWPn) Register Unlock bit
1 = UBWPn bits are not locked and can be modified
0 = UBWPn bits are locked and can be modified

Note 1: This field can only be modified when the NVMKEY unlock sequence is satisfied.

2: This field can be cleared at the same time as writing to UBWP[msb:lsb].

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

bit 23-0 **UBWP[23:0]**: Upper Boot Pages Write Protect bits
UBWP[n] = 1: Erase and write protection for upper boot page n is enabled
UBWP[n] = 0: Erase and write protection for upper boot page n is disabled

Note: This field can only be modified when the NVMKEY unlock sequence is satisfied and ULOCK = 1.

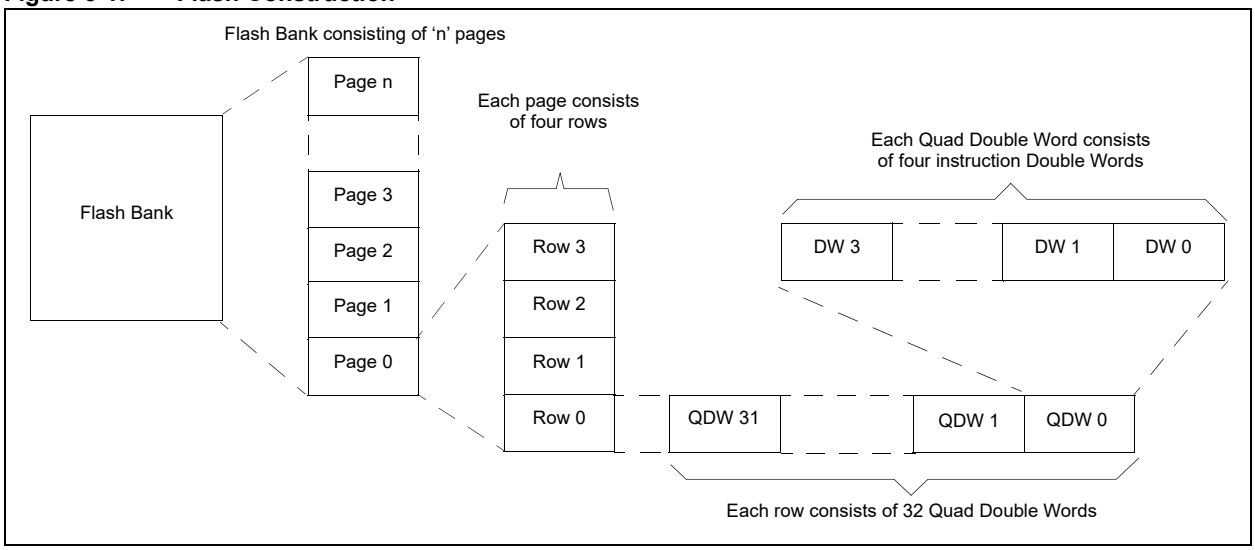
5.3 MEMORY CONFIGURATION

5.3.1 Flash Bank Construction

The BFM and PFM Flash memory bank 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 32 Quad Double Words (QDW). Each Quad Double Word consists of four instruction Double Words (DW). Flash memory can be programmed in rows, Quad Double Word (256-bit) or Single Double Word (64-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 5-1: Flash Construction



5.3.2 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 the system RAM.

Note:	Instruction code that is already stored in the cache when the programming operation is initiated will continue to execute. For more details, refer to **Section 41. "Prefetch Module for Devices with L1 CPU Cache"** (DS60001183).

5.4 BOOT FLASH MEMORY (BFM) PARTITIONS

5.4.1 BFM Write Protection

Pages in the upper and lower aliased regions can be protected individually using bits in the NVMUBWP and NVMLBWP register. See the specific device data sheet to check the upper region's Flash implementation. These bits correspond to pages referenced in the aliased region, so 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, ULOCK(NVMUBWP[31] and ULOCK(NVMLBWP[31])), that are set at reset and can be cleared by the user's software. When cleared, changes to write protection for that region can no longer be made. Once cleared, the ULOCK bits can only be set by a reset.

The write-protect registers, NVMUBWP and NVMLBWP, can only be changed when the unlock sequence is followed. See **Section 5.9 "NVMKEY Register Unlocking Sequence"** for more information.

5.5 PROGRAM FLASH MEMORY (PFM) BANK PARTITIONS

5.5.1 PFM Write Protection

Program memory page regions can be protected individually using bits in the NVMPWPLT and NVMPWPGTE registers. The NVMPWPLT register is used to set the program Flash pages lower than the provided address as write-protected. The NVMPWPGTE register is used to set the program Flash pages greater than the provided address as write-protected.

There is also an unlock bit, ULOCK (NVMPWPLT[31] and NVMPWPGTE[31]), that is set at Reset and can be cleared by the user's software. When cleared, changes to write protection of the PFM can no longer be made, including the ULOCK bit. The write-protect registers, NVMPWPLT and NVMPWPGTE, can only be changed when the unlock sequence is followed. See **Section 5.9 "NVMKEY Register Unlocking Sequence"** for more information.

5.6 ERROR CORRECTING CODE (ECC) AND FLASH PROGRAMMING

Some PIC32MZ W1 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).

For a given software application, ECC can be enabled at all times, disabled at all times or dynamically enabled using the FECCCON Configuration bits in the CFGCON0 register. When ECC is enabled at all times, the Single Word NVMOP programming command does not function and the Quad Double Word is the smallest unit of memory that can be programmed. When ECC is disabled or enabled dynamically, both the Single Double Word and Quad Double 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 Single DWord 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 Double Word or Row Programming commands, ECC data is written and tested for errors (and corrected if needed) when read. Table 5-2 describes the different ECC scenarios.

Table 5-2: ECC Programming Summary

FECCCON Setting	Programming Operation			Data Read
	Single Double Word Write	Quad Double Word Write	Row Write	
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 256-bit Quad Double Word or Row programming command. Divisions between ECC and non-ECC memory must be on even Quad Double Word boundaries (address bits 0 through 3 are equal to '0').

5.7 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's software at this vector address must 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.

5.7.1 Interrupts and CPU Stalling

Code cannot be fetched by the CPU from the same Flash bank, either BFM or PFM, that 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. 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.

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

5.8 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. When the Flash operation is complete, indicated by the hardware clearing the value of the WR bit (i.e., WR bit is set to '0'), the hardware will update the value in WRERR bit to indicate if an error occurred. The firmware must check the value of the WR bit to see if the flash operation has completed before checking the value of the WRERR bit. When the WRERR is set, any future 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 must be cleared manually in software by initiating a Flash operation (setting WR) referred to as NOP (0x00) (see the NVMOP bit fields in the Register 5-1). Note that executing the NVMOP NOP command clears WRERR, LVDERR and WR bits, but does not generate an interrupt event on completion.

Table 5-3: 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

5.9 NVMKEY REGISTER UNLOCKING SEQUENCE

Important register settings set incorrectly, like inadvertently setting the WR bit in the NVMCON register, could compromise the Flash memory. Therefore, certain bit fields are protected and the firmware must complete a register unlock sequence before changing the bit field value. This feature is implemented using the NVMKEY register.

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

Table 5-4: NVMKEY Register Unlocking and WREN

Operation	Unlock Sequence Required
Setting WR (NVMCON[15]) to start a write or erase operation	Yes
Changing any fields in the NVMPWPLT register	Yes
Changing any fields in the NVMPWPGTE register	Yes

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. If this happens, the unlock sequence will be aborted and firmware will not be able to change the value of the system lock-protected bit. In this scenario, the firmware must repeat steps 1-4. 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. Meaning, the write operation must be completed immediately after unlocking in step 3. Having the firmware perform a read-modify-write after step 3 would not be successful. The Set, Clear and Invert registers may be used, where applicable, for the target register in step 4.

Example 5-1 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 5-1: Initiate NVM Operation (Unlock Sequence Example)

```
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 compiled into a read-modify-write operation will fail.

5.10 WORD PROGRAMMING

The smallest block of data that can be programmed in a single operation is one Flash write word (64-bit). 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 two 32-bit data to be programmed to the NVMDATA0 register and NVMDATA1.
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. Initiate the programming operation (see **Section 5.9 "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 5-2 shows code for Word programming, where a value of 0x12345678 is programmed into location 0x1D008000.

Example 5-2: 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 5-1

// 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
    }
…
```

5.11 QUAD DOUBLE WORD PROGRAMMING

The process for Quad Double Word programming is identical to Word programming except that all eight of the NVMDATAx registers are used. The value of the NVMDATA0 register is programmed at address NVMADDR and so on. Refer to the following code example for details.

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

Quad Double Word programming will only succeed if the target address is in a page that is not write-protected. Once a Quad Double 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 5-3 shows code for Quad Double Word Programming. The value of 0x11111111 is programmed into location 0x1D008000 and so on. Refer to the following code example for details.

Example 5-3: Quad Double Word Programming Code Example

```
 …

// 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
    NVMDATA4 = 0x55555555   //value written to 0x1D008010
    NVMDATA5 = 0x66666666   //value written to 0x1D008014
    NVMDATA6 = 0x77777777   //value written to 0x1D008018
    NVMDATA7 = 0x88888888   //value written to 0x1D00801C

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

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

// Start programming
    NVMInitiateOperation();    // see Example 5-1

// 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
```

5.12 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 1024, 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 5-4 shows code for Row programming. Array `rowbuff` is populated with data and programmed into a row located at the physical address 0x10008000.

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

Example 5-4: Row Programming Example Code

```
 …

unsigned int rowbuff[1024]; // 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 5-1

// 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
    }
…
```

5.13 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 5-5 shows code for a single page erase operation at address 0x1D008000.

Example 5-5: 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 5-1

// 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
    }
…
```

5.14 PROGRAM FLASH MEMORY ERASE

Program Flash memory can be erased in the bank. The following discreet NVMOP command is implemented to accomplish this:

- Erase the entire PFM bank

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 5-6 shows code for erasing the program Flash bank.

Example 5-6: Program Flash Erase

```
…

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

// Enable Flash Write
    NVMCONbits.WREN = 1;

// Commence programming
    NVMInitiateOperation();        // see Example 5-1

// 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
    }
…
```

5.15 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.

5.16 OPERATION IN DEBUG MODE

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

5.17 EFFECTS OF VARIOUS RESETS

Device Resets, other than a Power-on Reset (POR), reset the entire contents of the NVMPWPLT and NVMPWPGTE registers. All other register content persists through a non-POR reset.

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

5.18 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 PIC32MZ W1 device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to Flash Program Memory include the following:

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

Note: Please visit the Microchip website (www.microchip.com) for additional application notes and code examples for the PIC32MZ W1 family of devices.

5.19 REVISION HISTORY

Revision A (September 2020)

This is the initial released version of the document.

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

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "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.

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

ISBN: 978-1-5224-6441-9

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

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
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

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

China - Beijing
Tel: 86-10-8569-7000

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

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

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

China - Guangzhou
Tel: 86-20-8755-8029

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

China - Hong Kong SAR
Tel: 852-2943-5100

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

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

China - Shanghai
Tel: 86-21-3326-8000

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

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

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

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

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

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

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

India - Pune
Tel: 91-20-4121-0141

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

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

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

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

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

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

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

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

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

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

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

Denmark - Copenhagen
Tel: 45-4485-5910
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-72400

Germany - Karlsruhe
Tel: 49-721-625370

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

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

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

Sweden - Gothenberg
Tel: 46-31-704-60-40

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

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