Section 4. Prefetch Cache Module

HIGHLIGHTS

This section of the manual contains the following topics:

4.1 INTRODUCTION

Note:	Prefetch cache is available in select devices only. Refer to the appropriate data sheet for the availability of a prefetch cache module on specific devices.

This section describes the features and operation of the prefetch cache module in the PIC32MX device family. Prefetch cache features increase system performance for most applications.

PFM cache and prefetch cache modules increase performance for applications that execute out of the cacheable Program Flash Memory (PFM) region by implementing the following features:

- Instruction Caching

  The 16-line cache supplies an instruction every clock, for loops up to 256 bytes long.
- Data Caching

  Prefetch cache also allows the allocation of up to 4 cache lines for data storage to provide improved access for Flash-stored constant data.
- Predictive Prefetching

  The prefetch cache module provides instructions once per clock for linear code even without caching by prefetching ahead of the current program counter, hiding the access time of the Flash memory.

4.1.1 Additional Prefetch Cache Module Features

The prefetch cache module also include the following features:

- 16 Fully Associative Lockable Cache Lines
- 16-Byte Cache Lines
- Up to 4 Cache Lines Allocated to Data
- 2 Cache Lines with Address Mask to Hold Repeated Instructions
- Pseudo Least-Recently-Used (LRU) Replacement Policy
- All Cache Lines are Software Writable
- 16-Byte Parallel Memory Fetch
- Predictive Instruction Prefetch Cache

4.2 CACHE OVERVIEW

The prefetch cache module is a performance enhancing module included in some processors of the PIC32MX. When running at high clock rates, Wait states must be inserted into PFM Read transactions to meet the access time of the PFM. Wait states can be hidden to the core by prefetching and storing instructions in a temporary holding area that the CPU can access quickly. Although the data path to the CPU is 32-bits wide, the data path to the Program Memory Flash is 128-bits wide. This wide data path provides the same bandwidth to the CPU as a 32-bit path running at four times the frequency.

There are two main functions that the prefetch cache module performs: caching instructions when they are accessed, and prefetching instructions from the PFM before they are needed.

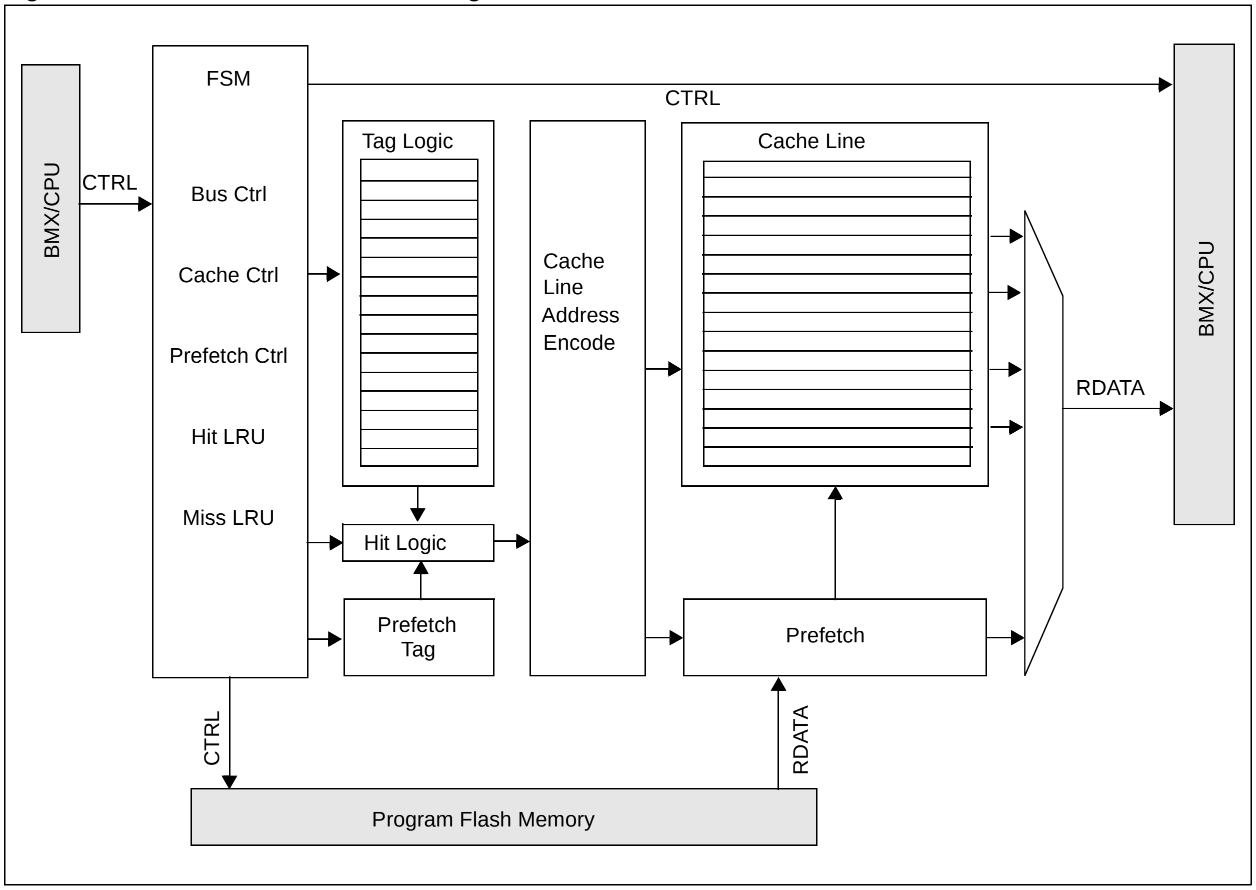
The cache holds a subset of the cacheable memory in temporary holding spaces known as cache lines. Each cache line has a tag describing what it is currently holding, and the address where it is mapped. Normally, the cache lines just hold a copy of what is currently in memory to make data available to the CPU without Wait states.

CPU requested data may or may not be in the cache. A cache-miss occurs if the CPU requests cacheable data that is not in the cache. In this case, a read is performed to the PFM at the correct address, the data is supplied to the cache and to the CPU. A cache-hit occurs if the cache contains the data that the CPU requests. In the case of a cache-hit, data is supplied to the CPU without Wait states.

The second main function of the prefetch cache module is to prefetch cache instructions. The module calculates the address of the next cache line and performs a read of the PFM to get the next 16-byte cache line. This line is placed into a 16-byte-wide prefetch cache buffer in anticipation of executing straight-line code.

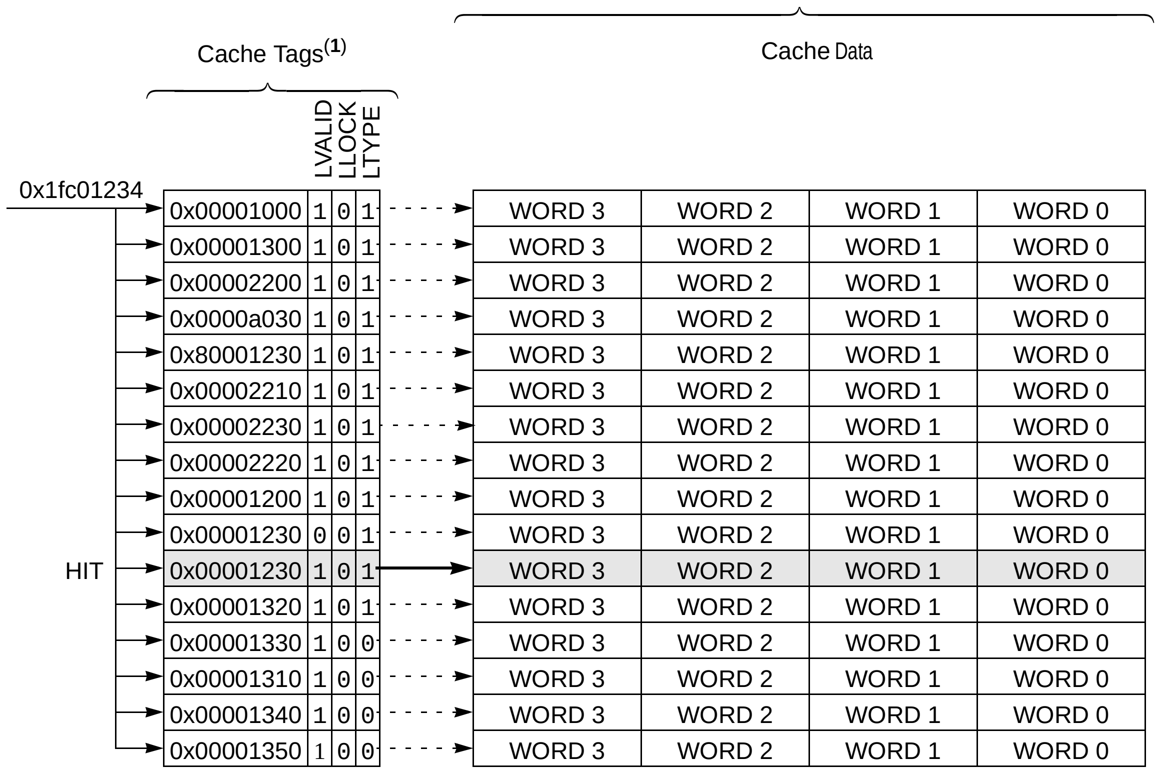
Figure 4-1 shows a block diagram of the prefetch cache module. Logically, the prefetch cache module fits between the Bus Matrix (BMX) module and the PFM module.

Figure 4-1: Prefetch Cache Block Diagram



To illustrate the basic operation of the prefetch cache, Figure 4-2 shows an example of the CPU requesting data from physical address 0x1FC01234. The prefetch cache simultaneously compares this address to all of the tags marked “valid”. Since the shaded entry below has this address, and is marked as valid, this is a cache hit. The proper data word from the data array is then directed to the CPU in a single clock period.

Figure 4-2: Cache Look-up Example[(2)]



Tag	LVALID	LLOCK	LTYPE				
0x00001000	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00001300	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00002200	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x0000a030	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x80001230	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00002210	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00002230	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00002220	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00001200	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00001230	0	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00001230	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00001320	1	0	1	WORD 3	WORD 2	WORD 1	WORD 0
0x00001330	1	0	0	WORD 3	WORD 2	WORD 1	WORD 0
0x00001310	1	0	0	WORD 3	WORD 2	WORD 1	WORD 0
0x00001340	1	0	0	WORD 3	WORD 2	WORD 1	WORD 0
0x00001350	1	0	0	WORD 3	WORD 2	WORD 1	WORD 0

Note 1: Bits 0-3 of the address in the Cache Tags register are always implied ‘`0`’.
2: Mask Fields are not shown and are assumed to be ‘`0`’.

4.2.1 Cache Organization

The cache consists of two arrays: tag and data. A data array could consist of program instructions or program data. The cache is physically tagged and address matches are based on the physical address not the virtual address.

Each line in the tag array contains the following information:

- Mask – address mask value
- Tag – tag address to match against
- Valid bit
- Lock bit
- Type – an instruction and/or data type-indicator bit

Each line in the data array contains 16-bytes of program instruction, or program data, depending on the value of the type-indicator bit.

Figure 4-3 shows the organization of a line. Note that the LMASK (CHEMSK<15:5>) and LTYPE (CHETAG<1>) fields are not programmable for every line. The LTAG (CHETAG<23:4>) field only implements the number of bits needed to fully map to the size of the PFM, e.g., if the Flash size is 512 KB, the LTAG (CHETAG<23:4>) field only implements bits 18 through 4.

Figure 4-3: Mask Line

31 … 16	15 … 5	4 … 0
RSVD	LMASK<15:5>	RSVD

Figure 4-4: Tag Line

31	30 … 24	23 … 4	3	2	1	0
LTAGBOOT	RSVD	LTAG<23:4>	LVALID	LLOCK	LTYPE	RSVD

Figure 4-5: Data Line

31 … 0
WORD 3
WORD 2
WORD 1
WORD 0

Cache arrays are shown in Table 4-1. Software can modify values in both the Tag Line and the Data Line of the cache. Configuration register field CHEIDX (CHEACC<3:0>) selects a line for access. That line can then be modified via the CHETAG, CHEMSK, CHEW0, CHEW1, CHEW2, and CHEW3 registers.

Table 4-1: Cache Arrays

Line #	Tag Array					Data Array[2]			
0	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
1	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
2	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
3	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
4	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
5	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
6	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
7	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
8	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
9	000h[1]	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
A	MASK	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
B	MASK	TAG	V	L	T[3]	Word 3	Word 2	Word 1	Word 0
C	000h[1]	TAG	V	L	T	Word 3	Word 2	Word 1	Word 0
D	000h[1]	TAG	V	L	T	Word 3	Word 2	Word 1	Word 0
E	000h[1]	TAG	V	L	T	Word 3	Word 2	Word 1	Word 0
F	000h[1]	TAG	V	L	T	Word 3	Word 2	Word 1	Word 0

Note 1: Read-only field.
2: Read zeros when device is code-protected. Read/write otherwise.
3: Type is fixed as instruction.

It is recommended that cache lines be modified while executing from non-cacheable addresses, since the cache controller does not protect against modifying the cache while executing from cacheable address.

Not all fields are writable. The LMASK (CHEMSK<15:5>) field is only writable for lines 10 and 11, and the LTYPE (CHETAG<1>) field is fixed to the “Instruction” setting for lines 0 through 11.

Note that lines allocated for Lock and Data affect the selection of the line to replace on a miss. However, they do not affect the usage order or pseudo LRU value.

4.3 CONTROL REGISTERS

Note:	Some devices in the PIC32MX family do not contain a prefetch cache module. For these devices, all prefetch cache register locations are reserved and should not be accessed.

The prefetch cache module contains the following Special Functions Registers (SFRs):

- CHECON: Prefetch Cache Control Register
  Manages configuration of the Prefetch Cache and controls Wait states.
- CHECONCLR, CHECONSET, CHECONINV: Atomic Bit Manipulation Write-only Registers for CHECON
- CHEACC: Prefetch Cache Access Register
  Points to one of the 16 cache lines to access using the CHETAG, CHEMSK, CHEW0, CHEW1, CHEW2, and CHEW3 registers.
- CHEACCCLR, CHEACCSET, CHEACCINV: Atomic Bit Manipulation Write-only Registers for CHEACC
- CHETAG: Prefetch Cache TAG Register
  Contains the address and type of information stored in a cache line.
- CHETAGCLR, CHETAGSET, CHETAGINV: Atomic Bit Manipulation Write-only Registers for CHETAG
- CHEMSK: Prefetch Cache TAG Mask Register
  Provides a mechanism to ignore TAG bits in CHETAG.
- CHEMSKCLR, CHEMSKSET, CHEMSKINV: Atomic Bit Manipulation Write-only Registers for CHEMSK
- CHEW0: Cache Word 0 Register
  Provides Access to the Prefetch Cache Data Array
- CHEW1: Cache Word 1 Register
  Provides Access to the Prefetch Cache Data Array
- CHEW2: Cache Word 2 Register
  Provides Access to the Prefetch Cache Data Array
- CHEW3: Cache Word 3 Register
  Provides Access to the Prefetch Cache Data Array
- CHELRU: Cache LRU Register
- CHEHIT: Cache Hit Statistics Register
- CHEMIS: Cache Miss Statistics Register
- PFABT: Prefetch Cache Abort Statistics Register
  A statistical register that contains the number of aborted Prefetch Cache operations.

The following table provides a brief summary of prefetch cache-related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

Table 4-2: Prefetch Cache SFRs Summary

Name		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
CHECON	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	CHECOH
	15:8	—	—	—	—	—	—	DCSZ<1:0>	
	7:0	—	—	PREFEN<1:0>		—	PFMWS<2:0>		
CHECONCLR	31:0	Clears selected bits in CHECON, read yields undefined value							
CHECONSET	31:0	Sets selected bits in CHECON, read yields undefined value							
CHECONINV	31:0	Inverts selected bits in CHECON, read yields undefined value							
CHEACC	31:24	CHEWEN	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	—
	7:0	—	—	—	—	CHEIDX<3:0>			
CHEACCCLR	31:0	Clears selected bits in CHEACC, read yields undefined value							
CHEACCSET	31:0	Sets selected bits in CHEACC, read yields undefined value							
CHEACCINV	31:0	Inverts selected bits CHEACC, read yields undefined value							
CHETAG	31:24	LTAGBOOT	—	—	—	—	—	—	—
	23:16	LTAG<23:16>							
	15:8	LTAG<15:8>							
	7:0	LTAG<7:4>				LVALID	LLOCK	LTYPE	—
CHETAGCLR	31:0	Clears selected bits in CHETAG, read yields undefined value							
CHETAGSET	31:0	Sets selected bits in CHETAG, read yields undefined value							
CHETAGINV	31:0	Inverts selected bits CHETAG, read yields undefined value							
CHEMSK	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	LMASK<15:8>							
	7:0	LMASK<7:5>			—	—	—	—	—
CHEMSKCLR	31:0	Clears selected bits in CHEMSK, read yields undefined value							
CHEMSKSET	31:0	Sets selected bits in CHEMSK, read yields undefined value							
CHEMSKINV	31:0	Inverts selected bits CHEMSK, read yields undefined value							
CHEW0	31:24	CHEW0<31:24>							
	23:16	CHEW0<23:16>							
	15:8	CHEW0<15:8>							
	7:0	CHEW0<7:0>							
CHEW1	31:24	CHEW1<31:24>							
	23:16	CHEW1<23:16>							
	15:8	CHEW1<15:8>							
	7:0	CHEW1<7:0>							
CHEW2	31:24	CHEW2<31:24>							
	23:16	CHEW2<23:16>							
	15:8	CHEW2<15:8>							
	7:0	CHEW2<7:0>							
CHEW3	31:24	CHEW3<31:24>							
	23:16	CHEW3<23:16>							
	15:8	CHEW3<15:8>							
	7:0	CHEW3<7:0>							

Table 4-2: Prefetch Cache SFRs Summary

Name		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
CHELRU	31:24	—	—	—	—	—	—	—	CHELRU<24>
	23:16	CHELRU<23:16>							
	15:8	CHELRU<15:8>							
	7:0	CHELRU<7:0>>							
CHEHIT	31:24	CHEHIT<31:24>							
	23:16	CHEHIT<23:16>							
	15:8	CHEHIT<15:8>							
	7:0	CHENIT<7:0>							
CHEMIS	31:24	CHEMIS<31:24>							
	23:16	CHEMIS<23:16>							
	15:8	CHEMIS<15:8>							
	7:0	CHEMIS<7:0>							
PFABT	31:24	PFABT<31:24>							
	23:16	PFABT<23:16>							
	15:8	PFABT<15:8>							
	7:0	PFABT<7:0>							

Register 4-1: CHECON: Cache Control Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	R/W-0
—	—	—	—	—	—	—	CHECOH
bit 23							bit 16

r-x	r-x	r-0	r-0	r-x	r-x	R/W-0	R/W-0
—	—	—	—	—	—	DCSZ<1:0>	
bit 15							bit 8

r-x	r-x	R/W-0	R/W-0	r-x	R/W-1	R/W-1	R/W-1
—	—	PREFEN<1:0>		—	PFMWS<2:0>		
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-17	**Reserved:** Write ‘0’; ignore read
bit 16	**CHECOH:** Cache Coherency setting on a PFM Program Cycle bit 1 = Invalidate all data and instruction lines 0 = Invalidate all data lnes and instruction lines that are not locked
bit 15-14	**Reserved:** Write ‘0’; ignore read
bit 13-12	**Reserved:** Must be written with zeros
bit 11-10	**Reserved:** Write ‘0’; ignore read
bit 9-8	**DCSZ<1:0>:** Data Cache Size in Lines bits 11 = Enable data caching with a size of 4 Lines 10 = Enable data caching with a size of 2 Lines 01 = Enable data caching with a size of 1 Line 00 = Disable data caching Changing this field causes all lines to be re-initialized to the “invalid” state.
bit 7-6	**Reserved:** Write ‘0’; ignore read
bit 5-4	**PREFEN<1:0>:** Predictive Prefetch Cache Enable bits 11 = Enable predictive prefetch cache for both cacheable and non-cacheable regions 10 = Enable predictive prefetch cache for non-cacheable regions only 01 = Enable predictive prefetch cache for cacheable regions only 00 = Disable predictive prefetch cache
bit 3	**Reserved:** Write ‘0’; ignore read

Register 4-1: CHECON: Cache Control Register (Continued)

bit 2-0	**PFMWS<2:0>:** PFM Access Time Defined in terms of SYSLK Wait states bits `111` = Seven Wait states `110` = Six Wait states `101` = Five Wait state `100` = Four Wait states `011` = Three Wait states `010` = Two Wait states `001` = One Wait state `000` = Zero Wait states

Register 4-2: CHECONCLR: CHECON Clear Register

Write clears selected bits in CHECON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in CHECON**

A write of '`1`' in one or more bit positions clears the corresponding bit(s) in CHECON register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHECONCLR = 0x00010020` will clear bits 16 and 5 in CHECON register.

Register 4-3: CHECONSET: CHECON Set Register

Write sets selected bits in CHECON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in CHECON**

A write of '`1`' in one or more bit positions sets the corresponding bit(s) in CHECON register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHECONSET = 0x00010020` will set bits 16 and 5 in CHECON register.

Register 4-4: CHECONINV: CHECON Invert Register

Write inverts selected bits in CHECON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in CHECON**

A write of '`1`' in one or more bit positions inverts the corresponding bit(s) in CHECON register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHECONINV = 0x00010020` will invert bits 16 and 5 in CHECON register.

Register 4-5: CHEACC: Cache Access

R/W-0	r-x	r-x	r-x	r-x	r-x	r-x	r-x
CHEWEN	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 15							bit 8

r-x	r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	CHEIDX<3:0>			
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31	**CHEWEN:** Cache Access Enable bits for registers CHETAG, CHEMSK, CHEW0, CHEW1, CHEW2, and CHEW3 1 = The cache line selected by CHEIDX is writeable 0 = The cache line selected by CHEIDX is not writeable
bit 30-4	**Reserved:** Write ‘0’; ignore read
bit 3-0	**CHEIDX<3:0>:** Cache Line Index bits The value selects the cache line for reading or writing.

Register 4-6: CHEACCCLR: CHEACC Clear Register

Write clears selected bits in CHEACC, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in CHEACC**

A write of '`1`' in one or more bit positions clears the corresponding bit(s) in CHEACC register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHEACCCLR = 0x80000000` will clear bit 31 in CHEACC register.

Register 4-7: CHEACCSET: CHEACC Set Register

Write sets selected bits in CHEACC, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in CHEACC**

A write of '`1`' in one or more bit positions sets the corresponding bit(s) in CHEACC register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHEACCSET = 0x80000000` will clear bit 31 in CHEACC register.

Register 4-8: CHEACCINV: CHEACC Invert Register

Write inverts selected bits in CHEACC, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in CHEACC**

A write of '`1`' in one or more bit positions inverts the corresponding bit(s) in CHEACC register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHEACCINV = 0x80000000` will invert bit 31 in CHEACC register.

Register 4-9: CHETAG(1): Cache TAG Register

R/W-0	r-x	r-x	r-x	r-x	r-x	r-x	r-x
LTAGBOOT	—	—	—	—	—	—	—
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
LTAG<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
LTAG<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-0	R/W-0	R/W-1	r-0
LTAG<7:4>				LVALID	LLOCK	LTYPE	—
bit 7							bit 0

Legend:

R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: ('0', '1', x = Unknown)		

bit 31	**LTAGBOOT:** Line TAG Address Boot 1 = The line is in the 0x1D000000 (physical) area of memory 0 = The line is in the 0x1FC00000 (physical) area of memory
bit 30-24	**Reserved:** Write '0'; ignore read
bit 23-4	**LTAG<23:4>:** Line TAG Address bits LTAG bits are compared against physical address <23:4> to determine a hit. Because its address range and position of Flash in kernel space and user space, the LTAG Flash address is identical for virtual addresses, (system) physical addresses, and Flash physical addresses.
bit 3	**LVALID:** Line Valid bit 1 = The line is valid and is compared to the physical address for hit detection 0 = The line is not valid and is not compared to the physical address for hit detection
bit 2	**LLOCK:** Line Lock bit 1 = The line is locked and will not be replaced 0 = The line is not locked and can be replaced
bit 1	**LTYPE:** Line Type bit 1 = The line caches instruction words 0 = The line caches data words
bit 0	**Reserved:** Write '0'; ignore read

Note 1: The TAG and Status of the Line pointed to by CHEIDX (CHEACC<3:0>).

Register 4-10: CHETAGCLR: CHETAG Clear Register

Write clears selected bits in CHETAG, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in CHETAG**

A write of '`1`' in one or more bit positions clears the corresponding bit(s) in CHETAG register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHETAGCLR = 0x0000000C` will clear bits 2 and 3 in CHETAG register.

Register 4-11: CHETAGSET: CHETAG Set Register

Write sets selected bits in CHETAG, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in CHETAG**

A write of '`1`' in one or more bit positions sets the corresponding bit(s) in CHETAG register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHETAGSET = 0x00000004` will set bit 2 in CHETAG register.

Register 4-12: CHETAGINV: CHETAG Invert Register

Write inverts selected bits in CHETAG, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in CHETAG**

A write of '`1`' in one or more bit positions inverts the corresponding bit(s) in CHETAG register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHETAGINV = 0x00000010` will invert bit 4 in CHETAG register.

Register 4-13: CHEMSK[1]: Cache TAG Mask Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 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
LMASK<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	r-x	r-x
LMASK<7:5>			—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15-5 **LMASK<15:5>:** Line Mask bits
- 1 = Enables mask logic to force a match on the corresponding bit position in LTAG (CHETAG<23:4>) and the physical address.
- 0 = Only writeable for values of CHEIDX (CHEACC<3:0>) equal to OxOA and OxOB. Disables mask logic.

bit 4-0 **Reserved:** Write ‘0’; ignore read

Note 1: The TAG Mask of the Line pointed to by CHEIDX (CHEACC<3:07>).

Register 4-14: CHEMSKCLR: CHEMSK Clear Register

Write clears selected bits in CHEMSK, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in CHEMSK**

A write of '`1`' in one or more bit positions clears the corresponding bit(s) in CHEMSK register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHEMSKCLR = 0x00008020` will clear bits 15 and 5 in CHEMSK register.

Register 4-15: CHEMSKSET: CHEMSK Set Register

Write sets selected bits in CHEMSK, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in CHEMSK**

A write of '`1`' in one or more bit positions sets the corresponding bit(s) in CHEMSK register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHEMSKSET = 0x00008020` will set bits 15 and 5 in CHEMSK register.

Register 4-16: CHEMSKINV: CHEMSK Invert Register

Write inverts selected bits in CHEMSK, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in CHEMSK**

A write of '`1`' in one or more bit positions inverts the corresponding bit(s) in CHEMSK register and does not affect unimplemented or read-only bits. A write of '`0`' will not affect the register.

Example: `CHEMSKINV = 0x00008020` will invert bits 15 and 5 in CHEMSK register.

Register 4-17: CHEW0: Cache Word 0

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW0<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW0<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW0<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW0<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0	**CHEW0<31:0>:** Word 0 of the cache line selected by CHEACC.CHEIDX Readable only if the device is not code-protected.

Register 4-18: CHEW1: Cache Word 1

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW1<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW1<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW1<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW1<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0	**CHEW1<31:0>:** Word 1 of the cache line selected by CHEACC.CHEIDX Readable only if the device is not code-protected.

Register 4-19: CHEW2 Cache Word 2

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW2<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW2<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW2<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW2<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0	**CHEW2<31:0>:** Word 2 of the cache line selected by CHEACC.CHEIDX Readable only if the device is not code-protected.

Register 4-20: CHEW3[(1)]: Cache Word 3

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW3<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW3<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW3<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEW3<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **CHEW3<31:0>:** Word 3 of the cache line selected by CHEACC.CHEIDX
Readable only if the device is not code-protected.

Note 1: This register is a window into the cache data array and is readable only if the device is not code-protected.

Register 4-21: CHELRU: Cache LRU Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	R-0
—	—	—	—	—	—	—	CHELRU<24>
bit 31							bit 24

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHELRU<23-16>							
bit 23							bit 16

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHELRU<15-8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
CHELRU<7-0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-25	**Reserved:** Write ‘0’; ignore read
bit 24-0	**CHELRU<24:0>:** Cache Least Recently Used State Encoding bits CHELRU indicates the Pseudo-LRU state of the cache.

Register 4-22: CHEHIT: Cache Hit Statistics Register

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEHIT<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEHIT<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEHIT<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEHIT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0	**CHEHIT<31:0>:** Cache Hit Count bits Incremented each time the processor issues an instruction fetch or load that hits the prefetch cache from a cacheable region. Non-cacheable accesses do not modify this value.

Register 4-23: CHEMIS: Cache Miss Statistics Register

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEMIS<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEMIS<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEMIS<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
CHEMIS<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0	**CHEMIS<31:0>:** Cache Miss Count bits Incremented each time the processor issues an instruction fetch from a cacheable region that misses the prefetch cache. Non-cacheable accesses do not modify this value.

Register 4-24: PFABT: Prefetch Cache Abort Statistics Register

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
PFABT<31:24>							
bit 31							bit 24

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
PFABT<23:16>							
bit 23							bit 16

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
PFABT<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
PFABT<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0	**PFABT<31:0>:** Prefab Abort Count bits Incremented each time an automatic prefetch cache is aborted due to a non-sequential instruction fetch, load or store.

4.4 CACHE OPERATION

The cache and prefetch cache module implements a fully associative 16-line cache. Each line consists of 128 bits (16 bytes). The cache and prefetch cache module only request 16-byte aligned instruction data from the PFM. If the CPU requested address is not aligned to a 16-byte boundary, the module will align the address by dropping address bits<3:0>. When configured only as a cache, the module loads multiple instructions into a line on a miss. It uses the pseudo LRU algorithm to select which line receives the new set of instructions. The cache controller uses the Wait states state values from PFMWS (CHECON<2:0>) to determine how long it must wait for a Flash access when it detects a miss. On a hit, the cache returns data in zero Wait states. If the code is 100% linear, the Cache-Only mode will provide instructions back to the CPU with Wait states only on the first instruction of a cache line. For 32-bit linear code, Wait states are seen every four instructions. For 16-bit linear code, Wait states occur only once for every eight instructions executed.

4.5 CACHE CONFIGURATIONS

The CHECON register controls the configurations available for instruction and data caching of PFM. Two parameters control the allocation of cache lines to specific features.

The DCSZ (CHECON<9:8>) field controls the number of lines allocated to program data caching. Table 4-3 shows the cache line relationship for values of DCSZ (CHECON<9:8>). The data caching capability is for read-only data, e.g., constants, parameters, table data, etc., that are not modified.

Table 4-3: Program Data Cache

DCSZ<1:0>	Lines Allocated to Program Data
`00`	None
`01`	Cache Line Number 15
`10`	Cache Lines Number 14 and 15
`11`	Cache Lines Number 12 through 15

The PREFEN (CHECON<5:4>) field controls predictive prefetching, which allows the cache controller to speculatively fetch the next 16-byte aligned set of instructions.

4.5.1 Line Locking

Each line in the cache can be locked to hold its contents. A line is locked if both LVALID (CHETAG<3>) = `1` and LLOCK (CHETAG<2>) = `1`. If LVALID = `0` and LLOCK = `1`, the cache controller issues a preload request (see Section **4.5.3 "Preload Behavior"**). Locking cache lines may reduce the performance of general program flow. However, if one or two function calls consume a significant percent of overall processing, locking their addresses can provide improved performance.

Though any number of lines can be locked, the cache works more efficiently when locking either 1 or 4 lines. If locking 4 lines, choose those lines in which the line numbers, when divide by 4, have the same quotient. This locks an entire LRU group which benefits the LRU algorithm. For example, lines 8, 9, A, and B each have a quotient of 2 when divided by 4.

4.5.2 Address Mask

Cache lines 10 and 11 allow masking of the CPU address, and the tag address, to force a match on corresponding bits. The LMASK (CHEMSK<15:5>) field is set up to complement the interrupt vector spacing field in the CPU. This feature allows boot code to lock the first four instructions of a vector in the cache. If all vectors contain identical instructions in their first four locations, then setting the LMASK (CHEMSK<15:5>) to match the vector spacing, and the LTAG (CHETAG<23:4>) to match the vector base address, causes all the vector addresses to hit the cache. The cache responds with zero Wait states and immediately initiates a fetch of the next set of four instructions for the requesting vector if prefetch cache is enabled.

Using LMASK (CHEMSK<15:5>) is restricted to aligned address ranges. Its size allows for a maximum range of 32 KB and a minimum spacing of 32 B. Using the two lines in conjunction provides the ability to have different ranges and different spacing.

Setting up the address mask such that more than one line will match an address causes undefined results. Therefore, it is highly recommended that masking is set up before entering cacheable code.

4.5.3 Preload Behavior

Application code can direct the cache controller to preform a preload of a cache line and lock it with instructions or data from the Flash. The preload function uses the CHEACC.CHEIDX register field to select the cache line into which the load is directed. Setting CHEACC.CHEWEN to ‘1’ enables writes to the CHETAG register.

Writing LVALID (CHETAG<3>) = 0 and LLOCK (CHETAG<2>) = 1 causes a preload request to the cache controller. The controller acknowledges the request in the cycle after the write and, if possible, stops any outstanding Flash access, and stalls any CPU load from the cache or Flash.

When the controller has finished or stalled the previous transaction, it initiates a Flash read to fetch the instructions, or data, requested using the address in LTAG (CHETAG<23:4>). After the programmed number of Wait states, as defined by PFMWS (CHECON<2:0>), the controller updates the data array with the values read from Flash. On the update, it sets LVALID (CHETAG<3>) = 1. The LRU state of the line is not affected.

Once the controller finishes updating the cache, it allows CPU requests to complete. If this request misses the cache, the controller initiates a Flash read, which incurs the full Flash access time.

4.5.4 Bypass Behavior

Processor accesses in which cache coherency attributes indicate uncacheable addresses bypass the cache. In bypass, the module accesses the PFM for every instruction, incurring the Flash access time as defined by PFMWS (CHECON<2:0>).

Preliminary

4.5.5 Predictive Prefetch Cache Behavior

When configured for predictive prefetch cache on cacheable addresses, the module predicts the next line address and returns it into the pseudo LRU line of the cache. If enabled, the prefetch cache function starts predicting based on the first CPU instruction fetch. When the first line is placed in the cache, the module simply increments the address to the next 16-byte aligned address and starts a Flash access. When running linear code (i.e. no jumps), the Flash returns the next set of instructions into the prefetch cache buffer on or before all instructions can be executed from the previous line.

If, at any time during a predicted Flash access, a new CPU address does not match the predicted one, the Flash access will be changed to the correct address. This behavior does not cause the CPU access to take any longer than it does without prediction.

If an access that misses the cache hits the prefetch cache buffer, the instructions are placed in the pseudo LRU line, along with its address tag. The pseudo LRU value is marked as the most recently used line, and other lines are updated accordingly. If an access misses both the cache and the prefetch cache buffer, the access passes to the Flash, and those returning instructions are placed in the pseudo LRU line.

When configured for predictive prefetch cache on non-cacheable addresses, the controller only uses the prefetch cache buffer. The LRU cache line is not updated for hits or fills, so the cache remains intact. For linear code, enabling predictive prefetch cache for non-cacheable addresses allows the CPU to fetch instructions in zero Wait states.

It is not useful to use non-cacheable predictive prefetching when accesses to the Flash are set for zero Wait states. The controller holds prefetched instructions on the output of the Flash for up to 3 clock cycles (while the CPU is fetching from the buffer). This consumes more power, without any benefit, for zero-Wait-state Flash accesses.

Predictive data prefetching is not supported. However, a data access in the middle of a predictive instruction fetch causes the cache controller to stop the Flash access for the instruction fetch, and to start the data load from Flash. The predictive prefetch cache does not resume, but instead, waits for another instruction fetch. At which time, it either fills the buffer because of a miss, or starts a prefetch cache because of a hit.

4.5.6 Cache Replacement Policy

The cache controller uses a pseudo-LRU replacement policy for cache line fills that are caused by a read miss. The policy allows any line in the last quarter of least recently used lines to be replaced. Enabling locking and data caching affect the line to be replaced, but not the actual value of the pseudo-LRU.

4.6 COHERENCY SUPPORT

It is not possible to execute out of cache while programming the Flash memory. The Flash controller stalls the cache during the programming sequence. Therefore, user code that initiates a programming sequence should not be located in a cacheable address region.

During a programming operation, the prefetch cache is flushed by invalidating either all, or some of the cache lines.

If CHECOH (CHECON<16>) is set, every cache line is invalidated and unlocked during a Flash program memory write operation. The cache tags and masks are also cleared for all lines.

If CHECOH is not set, only lines that are not locked are forced invalid. Lines that are locked are retained.

4.7 EFFECTS OF RESET

4.7.1 On Reset

- All cache lines are invalidated
- All cache lines revert to instruction
- All cache lines are unlocked
- The LRU order is sequential, with line 0 being the least recently used
- All mask bits are cleared
- All registers revert to their Reset state

4.7.2 After Reset

- The module operates as per the values in the CHECON register
- The cache obeys the core’s cache coherency attributes

4.8 DESIGN TIPS

Even while running at clock frequencies allowing for zero-Wait-state operation, the cache function proves useful as a power-saving technique. Accesses to the Flash memory consume more power than accesses to the cache.

4.9 OPERATION IN POWER-SAVING MODES

Note: In this manual, a distinction is made between a power mode as it is used in a specific module, and a power mode as it is used by the device, e.g., Sleep mode of the Comparator and SLEEP mode of the CPU. To indicate which type of power mode is intended, uppercase and lowercase letters (Sleep, Idle, Debug) signify a module power mode, and all uppercase letters (SLEEP, IDLE, DEBUG) signify a device power mode.

4.9.1 SLEEP Mode

When the device enters SLEEP mode, the prefetch cache is disabled and placed into a low-power state where no clocking occurs in the prefetch cache module.

4.9.2 IDLE Mode

When the device enters IDLE mode, the cache and prefetch cache clock source remains functional and the CPU stops executing code. Any outstanding prefetch cache completes before the module stops its clock via automatic clock gating.

4.9.3 DEBUG Mode

The behavior of the prefetch cache is unaltered by DEBUG mode. Care must be taken to make sure the cache remains coherent during DEBUG mode execution when using software breakpoints. If a debugger places a software break instruction in the cache, the line should be locked before returning control to the application. When a locked software breakpoint is removed, the line should be unlocked and invalidated, causing the original instructions to be reloaded from the PFM upon execution.

4.10 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 PIC32MX device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the prefetch cache module are:

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 PIC32MX family of devices.

4.11 REVISION HISTORY

Revision A (October 2007)

This is the initial released version of this document.

Revision B (October 2007)

Updated document to remove Confidential status.

Revision C (April 2008)

Revised status to Preliminary; Revise U-0 to r-x.

Revision D (June 2008)

Change Reserved bits from “Maintain as” to “Write”.