TMS320VC5510 DSP Instruction Cache Reference Guide

SPRU576D
June 2004

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This manual describes the features and operation of the instruction cache that is available on the TMS320VC5510 digital signal processors (DSPs) in the TMS320C55x™ (C55x™) DSP generation. This manual assumes the reader has some fundamental knowledge about cache operation.

Notational Conventions

This document uses the following conventions:

In most cases, hexadecimal numbers are shown with the suffix h. For example, the following number is a hexadecimal 40 (decimal 64):

40h

Similarly, binary numbers often are shown with the suffix b. For example, the following number is the decimal number 4 shown in binary form:

0100b

Related Documentation From Texas Instruments

The following documents describe the C55x devices and related support tools. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

TMS320VC5510 Fixed-Point Digital Signal Processor Data Manual (literature number SPRS076) describes the features of the TMS320VC5510 fixed-point DSP and provides signal descriptions, pinouts, electrical specifications, and timings for the device.

TMS320C55x Technical Overview (literature number SPRU393). This overview is an introduction to the TMS320C55x DSPs, the latest generation of fixed-point DSPs in the TMS320C5000™ DSP platform. Like the previous generations, this processor is optimized for high performance and low-power operation. This book describes the CPU architecture, low-power enhancements, and embedded emulation features.

TMS320C55x DSP CPU Reference Guide (literature number SPRU371) describes the architecture, registers, and operation of the CPU for the TMS320C55x DSPs.

TMS320C55x DSP Peripherals Overview Reference Guide (literature number SPRU317) introduces the peripherals, interfaces, and related hardware that are available on TMS320C55x DSPs.

TMS320C55x DSP Algebraic Instruction Set Reference Guide (literature number SPRU375) describes the TMS320C55x DSP algebraic instructions individually. Also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the mnemonic instruction set.

TMS320C55x DSP Mnemonic Instruction Set Reference Guide (literature number SPRU374) describes the TMS320C55x DSP mnemonic instructions individually. Also includes a summary of the instruction set, a list of the instruction opcodes, and a cross-reference to the algebraic instruction set.

TMS320C55x Optimizing C/C++ Compiler User's Guide (literature number SPRU281) describes the TMS320C55x C/C++ Compiler. This C/C++ compiler accepts ISO standard C and C++ source code and produces assembly language source code for TMS320C55x devices.

TMS320C55x Assembly Language Tools User's Guide (literature number SPRU280) describes the assembly language tools (assembler, linker, and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for TMS320C55x devices.

TMS320C55x DSP Programmer's Guide (literature number SPRU376) describes ways to optimize C and assembly code for the TMS320C55x DSPs and explains how to write code that uses special features and instructions of the DSPs.

Trademarks

TMS320C5000, TMS320C55x, and C55x are trademarks of Texas Instruments.

Other trademarks are the property of their respective owners.

Contents

Figures

Tables

This page is intentionally left blank.

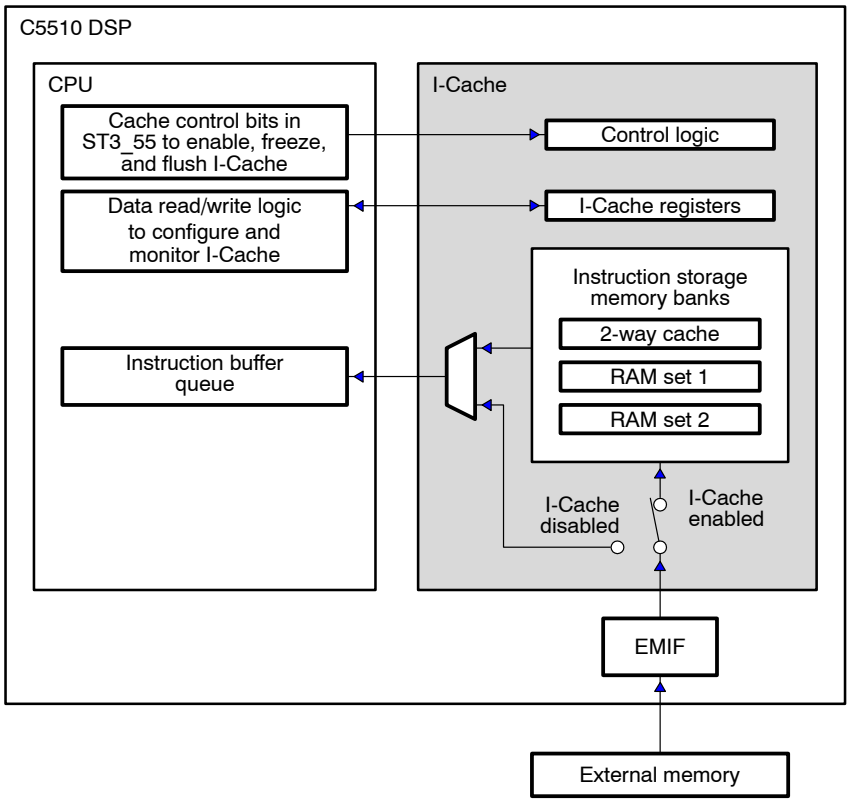
Instruction Cache

On the TMS320VC5510 digital signal processor (DSP), instructions can reside in internal memory or external memory. When instructions reside in external memory, the instruction cache (I-Cache) can improve the overall system performance by buffering the most recent instructions accessed by the CPU.

Figure 1 shows how the I-Cache fits into the TMS320VC5510 DSP system. CPU status register ST3_55 contains three cache control bits for enabling, freezing, and flushing the I-Cache (see section 3 on page 18). To configure the I-Cache and check its status, the CPU accesses a set of registers in the I-Cache (see section 7 on page 25). For storing instructions, the I-Cache contains:

- One 2-way cache. The 2-way cache uses 2-way set associative mapping and holds up to 16K bytes: 512 sets, two lines per set, four 32-bit words per line. In the 2-way cache, each line is identified by a unique tag.
- Two RAM sets (1 and 2). These two banks of RAM are available to hold blocks of code. Each RAM set holds up to 4K bytes: 256 lines, four 32-bit words per line. Each RAM set uses a single tag to identify a continuous range of memory addresses that is represented in the RAM set. Before enabling the I-Cache, configure the I-Cache to use zero, one, or both RAM sets.

Figure 1. Conceptual Block Diagram of the I-Cache in the DSP System



1 Introduction to the I-Cache

When the C5510 CPU requests instructions, it requests 32 bits at a time. To initiate an instruction fetch, the CPU sends a fetch request and a fetch address to the I-Cache.

If the I-Cache is enabled, it handles the fetch request as follows: If the requested word is in one of the data arrays (a *hit*), the I-Cache delivers the word to the CPU. If the requested word is not in the I-Cache (a *miss*), the I-Cache uses the external memory interface (EMIF) to fetch the 4-word external memory block that contains the requested word. As soon as the requested word arrives in the I-Cache, it is delivered to the CPU. Timing information for I-Cache hits and misses can be found in section 5 on page 22.

If the I-Cache is disabled, the data arrays are not checked. Instead, the fetch request and fetch address are passed to the EMIF. Once fetched by the EMIF, the requested 32-bit word is passed directly to the CPU.

Note:

The I-Cache does not automatically maintain coherency. If you write to a location in program memory, the corresponding line in the I-Cache is not updated. To regain coherency you must flush the I-Cache as described in section 3.2 (page 18).

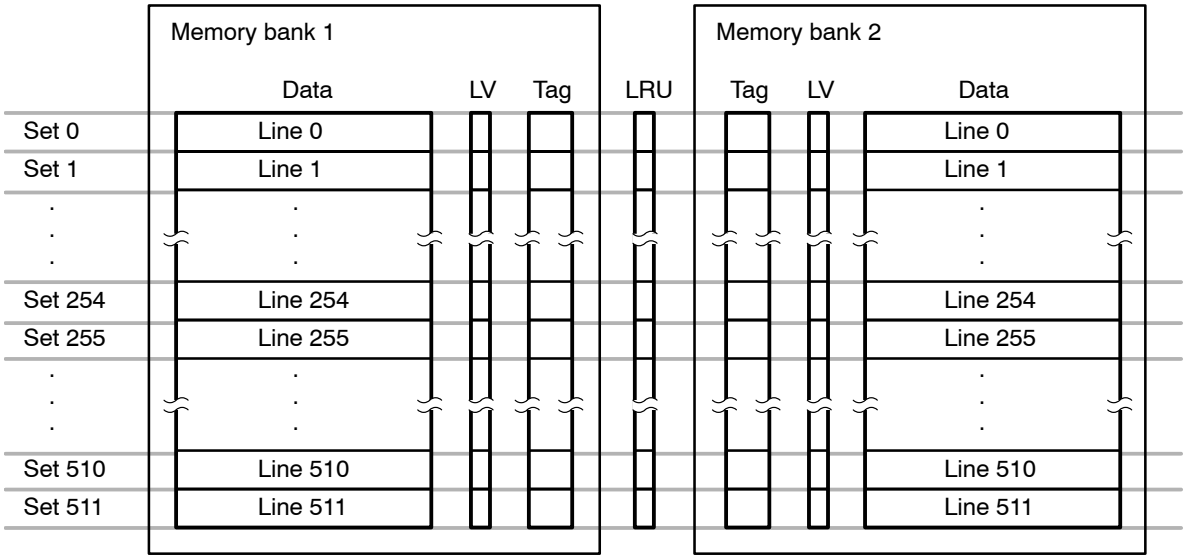
1.1 2-Way Cache

As shown in Figure 2, the 2-way cache has two memory banks. Each memory bank has the same parts:

- Data array. Each data array contains 512 lines (0 through 511) that the I-Cache can fill one by one in response to misses in the 2-way cache.
- Line valid (LV) bit array. Each line has a line valid bit. Once a line has been loaded, its line valid bit is set. Whenever the I-Cache is flushed, all 512 line valid bits are cleared, invalidating all the lines. For more information on flushing the I-Cache, see section 3.2 on page 18.
- Tag array. Each line has a tag field. When the I-Cache receives a 24-bit fetch address from the CPU, the I-Cache interprets bits 23–13 as a tag. When a line gets filled, the associated tag is stored in the tag field for that line.

Across the two memory banks, every two lines with the same number belong to one set. For example, line 0 of memory bank 1 and line 0 of memory bank 2 belong to set 0. When the I-Cache receives a fetch address, the I-Cache finds the set number in bits 12–4. If the I-Cache must replace one of the lines in the set, it uses a least-recently used (LRU) algorithm: The line replaced is the one that has been unused for the longest time. Each set has an LRU bit that is toggled to indicate which line should be replaced.

Figure 2. 2-Way Cache



1.2 RAM Sets 1 and 2

As shown in Figure 3, RAM set 1 and RAM set 2 each have the following parts:

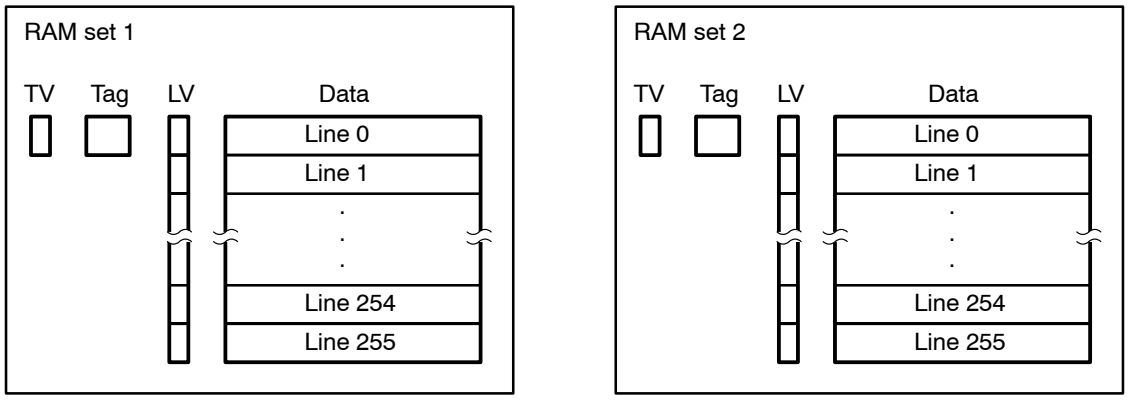
- Data array. Each data array contains 256 lines (0 through 255).
- Line valid (LV) bit array. Each line has a line valid bit. When a line has been loaded, its line valid bit is set. Whenever the I-Cache is flushed, all 256 line valid bits are cleared, invalidating all the lines. For more information on flushing the I-Cache, see section 3.2 on page 18.
- Tag field. The RAM set has one 12-bit tag field that indicates which range of external memory addresses are mapped to the RAM set. To select a tag for RAM set n (1 or 2), write to RAM-set tag register n. When you write to the tag register, the I-Cache immediately fills the RAM set with all the 32-bit words in the address range specified by the tag. As each line is loaded, the associated line valid bit is set.

- Tag valid (TV) bit. The RAM set has one tag valid bit. Just before filling the RAM set, the I-Cache clears the tag valid bit. When the filling is complete, the I-Cache sets the tag valid bit. For RAM set n (1 or 2), the tag valid bit is reflected in RAM-set control register n.

Note:

Once the I-Cache begins to fill a RAM set, the I-Cache will not service CPU instruction-fetch requests until the RAM-set fill operation is complete. For optimal code performance, it is recommended that instruction fetches not be made from external memory during RAM-set fill operations.

Figure 3. RAM Sets 1 and 2



2 I-Cache Operation

When the C5510 CPU requests instructions, it requests 32 bits at a time. With each request, the CPU sends a fetch address that indicates where to read the 32 bit requested word. When a fetch request arrives, the I-Cache performs an instruction presence check; that is, it determines whether the requested word is available in the 2-way cache and/or any RAM sets included in your I-Cache configuration.

Because the 2-way cache and RAM-set architectures are different, the I-Cache interprets the fetch address differently when searching the 2-way cache and when searching the RAM-set. Section 2.1 explains the differences.

Section 2.2 describes the steps of the instruction presence check and explains the factors that determine whether the I-Cache fetches the requested word from a RAM set, from the 2-way cache, or from external memory. Whenever possible, the I-Cache gets the requested word from a RAM set. If the requested word is in a RAM set but not in the 2-way cache, the word is fetched from the RAM set and the 2-way cache is not loaded with that word.

2.1 How the I-Cache Uses the Fetch Address

Figure 4 and Table 1 describe how the I-Cache uses the fetch address for the 2-way cache. Figure 5 and Table 2 describe the same for a RAM set.

Figure 4. Fetch Address Fields for the 2-Way Cache

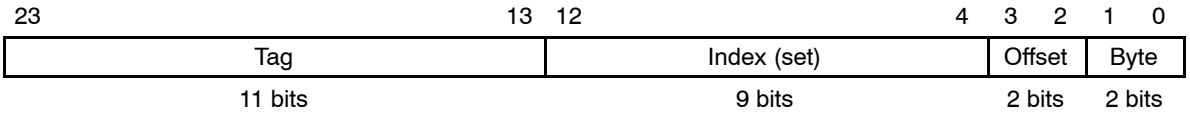


Table 1. Fetch Address Field Descriptions for the 2-Way Cache

Bit	Field	Description
23–13	Tag	Whenever a line of the 2-way cache is loaded from external memory, the tag portion of the fetch address is stored with the line (in the tag array). During an instruction presence check, the I-Cache uses the Index field to find the addressed set and then compares both tags in the set with the tag portion of the fetch address.
12–4	Index	This 9-bit value references one of the 512 sets of the 2-way cache. As shown in Figure 2 (page 12), each set has two lines.

Table 1. Fetch Address Field Descriptions for the 2-Way Cache (Continued)

Bit	Field	Description
3–2	Offset	When the I-Cache must read a 32-bit word from one of the lines of the 2-way cache, the offset field indicates which of the four 32-bit words in the line should be read.
1–0	Byte	This field is not used by the I-Cache but is the part of the fetch address that indicates the specific byte being addressed.

Figure 5. Fetch Address Fields for a RAM Set

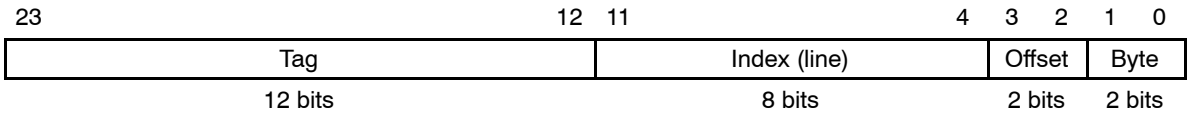


Table 2. Fetch Address Field Descriptions for a RAM Set

Bit	Field	Description
23–12	Tag	During an instruction presence check, the I-Cache compares the tag portion of the fetch address with the tag defined in the RAM-set tag register.
11–4	Index	This 8-bit value references one of the 256 lines of the RAM set.
3–2	Offset	When the I-Cache must read a 32-bit word from one of the lines of the RAM set, the offset field indicates which of the four 32-bit words in the line should be read.
1–0	Byte	This field is not used by the I-Cache but is the part of the fetch address that indicates the specific byte being addressed.

2.2 Instruction Presence Check and the Corresponding I-Cache Response

As mentioned earlier, when a fetch request arrives, the I-Cache performs an instruction presence check to determine whether the 32-bit requested word is available in the I-Cache. During the instruction presence check, the I-Cache performs these two operations on both the 2-way cache and the RAM sets:

1) Compares the tag portion of the fetch address with the tag in the data array at the location referenced by the Index portion of the fetch address.

2) Checks the line valid bit at the referenced location, to determine whether the line associated with the tag is valid.

If the tag comparison fails and/or the line valid bit is 0, this qualifies as a *miss*. If the instruction presence check finds a tag match and the line valid bit is 1, this qualifies as a *hit*. Table 3 summarizes the possible presence check cases (1 through 6) and the corresponding I-Cache responses. Whenever a line in the I-Cache must be loaded from external memory (cases 1, 2, and 5), the I-Cache uses the line load process described in section 2.3.

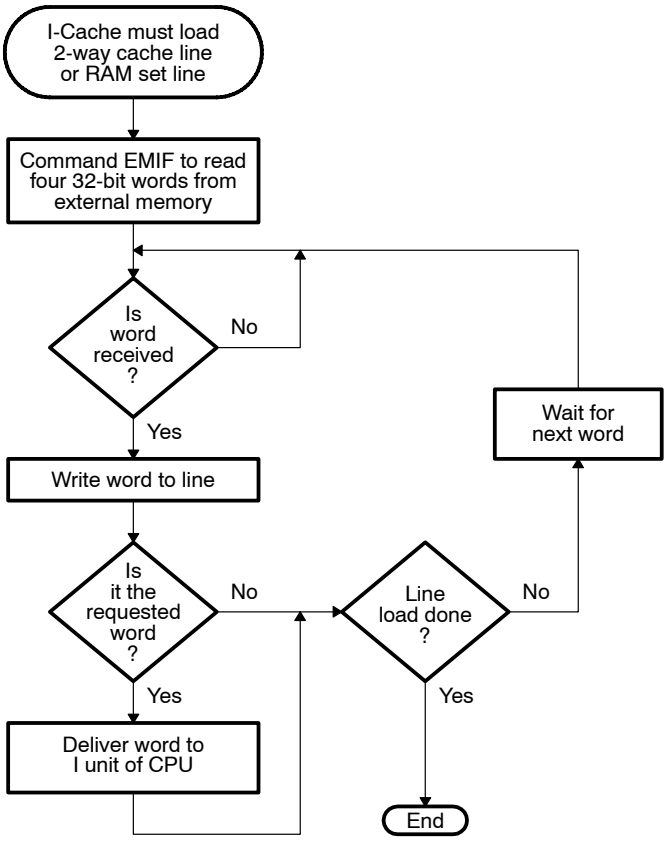
Table 3. Instruction Presence Check and I-Cache Response

Case	2-Way Cache	RAM Sets	Presence	I-Cache Response
1	Miss	Miss (no tag match)	False	2-way cache line loaded from external memory, requested 32-bit word delivered to CPU
2	Miss	Miss but tag match	False	RAM set line loaded from external memory, requested 32-bit word delivered to CPU
3	Miss	Hit	True	Requested 32-bit word taken directly from RAM set; 2-way cache line not loaded
4	Hit	Miss (no tag match)	True	Requested 32-bit word taken directly from 2-way cache
5	Hit	Miss but tag match	True	RAM set line loaded from external memory, requested 32-bit word delivered to CPU
6	Hit	Hit	True	Requested 32-bit word taken directly from RAM set

2.3 Line Load Process

When an instruction presence check results in a fetch from the external memory, the 4-word external memory block that contains the requested word is fetched and loaded into a line in the I-Cache. This line load process is illustrated in Figure 6. The I-Cache uses the external memory interface (EMIF) to fetch the 4-word block that contains the requested word. These four 32-bit words are written to the line in the I-Cache one word at a time. The I-Cache delivers the requested word to the CPU as soon as the word arrives in the data array, even if the rest of the line is still being loaded. When the entire line is loaded in the data array, the corresponding tag is written to the tag array and the line valid bit is set to validate the line.

Figure 6. Flow Chart of the Line Load Process



3 CPU Bits for Controlling the I-Cache

Control of the I-Cache is maintained not only through the I-Cache registers but also through three bits located in status register ST3_55 of the CPU. These bits are highlighted in Figure 7. For more details about ST3_55, see the *TMS320C55x DSP CPU Reference Guide* (SPRU371).

Figure 7. CAFRZ, CAEN, and CACLR Bits in ST3_55

15	14	13	12	11			8
CAFRZ	**CAEN**	**CACLR**	HINT	Reserved			
R/W−0	R/W−0	R/W−0					

7	6	5	4	3	2	1	0
CBERR	MPNMC	SATA	Reserved		CLKOFF	SMUL	SST

3.1 CAEN Bit to Enable or Disable the I-Cache

To enable the I-Cache, set the cache enable (CAEN) bit of ST3_55. To disable the I-Cache, clear the CAEN bit. When disabled, the lines of the I-Cache data arrays are not checked; instead, the I-Cache forwards instruction-fetch requests directly to the external memory interface (EMIF).

For proper operation of the I-Cache, configure the I-Cache before enabling it and disable the I-Cache before making any changes to its configuration. The procedures for configuring and enabling the I-Cache are in section 4 on page 20.

A DSP reset forces CAEN = 0 (I-Cache disabled).

3.2 CACLR Bit to Flush the I-Cache

The flush operation is defined as the invalidation of all of the lines in the I-Cache.

To flush the I-Cache, write 1 to the cache clear (CACLR) bit of ST3_55. In response, all the line valid bits of the 2-way cache and of the RAM sets are cleared. In addition, the tag valid bit of each RAM set is cleared. The CACLR bit remains 1 until the flush process is complete, at which time CACLR is automatically reset to 0.

A DSP reset forces CACLR = 0 (no flush in process).

3.3 CAFRZ Bit to Freeze the Contents of the I-Cache

When you write 1 to the cache freeze (CAFRZ) bit of ST3_55, the contents of the I-Cache are locked. Instruction words that were cached prior to the freeze are still accessible in the case of an I-Cache hit, but the data arrays are not updated in response to an I-Cache miss. To re-enable updates, write 0 to CAFRZ.

A DSP reset forces CAFRZ = 0 (I-Cache not frozen).

Note:

When the I-Cache is frozen (CAFRZ = 1), each I-Cache miss still causes a 4-word (16-byte) fetch cycle in the EMIF. It is recommended that you profile your code to minimize the number of misses during an I-Cache freeze.

4 Configuring and Enabling the I-Cache

This section gives the procedures for preparing and enabling the I-Cache for the three I-Cache configurations:

- ❑ 2-way cache and no RAM sets
- ❑ 2-way cache and one RAM set
- ❑ 2-way cache and two RAM sets

The I-Cache registers mentioned in this section are described in section 7 (page 25). The cache enable (CAEN) bit that is used to enable and disable the I-Cache is described in section 3 (page 18).

Notes:

1) Write to the control registers (ICGC, ICWC, ICRC1, and ICRC2) **only** when the I-Cache is **disabled** (CAEN = 0 in ST3_55).

2) Write to the RAM-set tag registers (ICRTAG1 and ICRTAG2) **only** when the I-Cache is **enabled** (after making CAEN = 1 in ST3_55, wait for IEN = 1 in ICST).

To configure with 2-way cache and no RAM sets:

1) Write to the appropriate control registers:
   - ❑ Write CBFFh to ICGC to indicate no RAM sets.
   - ❑ Write 000Fh to ICWC to initialize the logic for the 2-way cache.
2) Set the cache enable bit (CAEN) bit of CPU status register ST3_55 to send an enable request to the I-Cache.
3) Poll the I-Cache-enabled (IEN) bit of ICST until IEN = 1. (The I-Cache is not instantaneously enabled.)

To configure with 2-way cache and one RAM set:

1) Write to the appropriate control registers:
   - ❑ Write CE1Fh to ICGC to indicate one RAM set.
   - ❑ Write 000Fh to ICWC to initialize the logic for the 2-way cache.
   - ❑ Write 000Fh to ICRC1 to initialize the logic for RAM set 1.
2) Set the cache enable bit (CAEN) bit of CPU status register ST3_55 to send an enable request to the I-Cache.
3) Poll the I-Cache-enabled (IEN) bit of ICST until IEN = 1. (The I-Cache is not instantaneously enabled.)

4) Write the desired tag to ICRTAG1. When you write to the tag register, the tag is used to immediately fill RAM set 1 from external memory.

   While the I-Cache is enabled, you can write to the tag register at any time to change the RAM-set address range. Each time you load the tag register, RAM set 1 is immediately filled from the selected address range.

5) To monitor the RAM-set filling, poll the tag-valid bit: When R1TVALID = 1 in ICRC1, the I-Cache has finished filling RAM set 1.

To configure with 2-way cache and two RAM sets:

1) Write to the appropriate control registers:

   - ❏ Write CFFFh to ICGC to indicate two RAM sets.
   - ❏ Write 000Fh to ICWC to initialize the logic for the 2-way cache.
   - ❏ Write 000Fh to ICRC1 to initialize the logic for RAM set 1.
   - ❏ Write 000Fh to ICRC2 to initialize the logic for RAM set 2.

2) Set the cache enable bit (CAEN) bit of CPU status register ST3_55 to send an enable request to the I-Cache.

3) Poll the I-Cache-enabled (IEN) bit of ICST until IEN = 1, indicating that the I-Cache is enabled. (The I-Cache is not instantaneously enabled.)

4) Write to the RAM-set tag registers:

   - ❏ Write the desired tag to ICRTAG1. When you write to the tag register, the tag is used to immediately fill RAM set 1 from external memory.
   - ❏ Write the desired tag to ICRTAG2. When you write to the tag register, the tag is used to immediately fill RAM set 2 from external memory.

   While the I-Cache is enabled, you can write to a tag register at any time to change the address range as necessary. Each time you load a tag register, the corresponding RAM set is immediately filled from the selected address range.

5) To monitor the RAM-set filling, poll the tag-valid bits:

   - ❏ When R1TVALID = 1 in ICRC1, the I-Cache is done filling RAM set 1.
   - ❏ When R2TVALID = 1 in ICRC2, the I-Cache is done filling RAM set 2.

5 Timing Considerations

As the I-Cache fetches and returns 32-bit words requested by the CPU, two key time periods affect the speed of the I-Cache:

- ❏ Hit time
- ❏ Miss penalty

5.1 Hit Time

The *hit time* is the time required for the I-Cache to deliver the 32-bit requested word to the CPU in the case of a hit (when the word is present in the I-Cache). The hit time is either 1 or 2 CPU clock cycles:

- ❏ An initial request (a request that follows a period of inactivity) has a hit time of 2 cycles.
- ❏ Subsequent requests have a hit time of 1 cycle if:
  - ■ The requests are consecutive (no inactivity in between) **and**
  - ■ The requests are to sequential addresses
- ❏ Subsequent requests have a hit time of 2 cycles if:
  - ■ The requests are not consecutive **or**
  - ■ The requests are to nonsequential addresses

5.2 Miss Penalty

The *miss penalty* is the time required for the I-Cache to deliver the 32-bit requested word to the CPU in the case of a miss (when the word must be fetched from external memory). In response to a miss, the I-Cache requests four words from the external memory interface (EMIF) to load the appropriate line.

The miss penalty due to an initial request to the EMIF is:

1) Four cycles for the I-Cache to receive the fetch request, detect an I-Cache miss, and forward the fetch request to the EMIF.

2) X cycles for the EMIF to get the requested word to the I-Cache, where X depends on factors such as:

   - ❑ The initial access latency of the type of external memory that is used
   - ❑ The position of the requested word in the I-Cache line. For example, if the requested word is the third word of the line, two words are fetched before the requested word.
   - ❑ Whether the four words are fetched in a burst access (if synchronous memory is used)

3) Three cycles for the I-Cache to get the requested 32-bit word to the instruction fetch unit (I unit) of the CPU.

Subsequent requests can incur a smaller miss penalty if the external memory is synchronous. After accessing the first word from synchronous memory, the EMIF can return each of the remaining words in a single cycle.

The I-Cache includes a feature that reduces miss penalties overall. As mentioned earlier, the I-Cache gives the requested word to the CPU as soon as it arrives in the I-Cache line, rather than after the whole line is loaded.

6 Power, Emulation, and Reset Considerations

6.1 Idle Mode for Reducing Power Consumed

If you want to temporarily halt the I-Cache to reduce power, you can place it in its idle mode:

1) Select the idle mode for the I-Cache by making CACHEI = 1 in the idle configuration register (ICR) of the DSP.

2) Execute the IDLE instruction.

When the I-Cache is in its idle mode or is disabled, instruction-fetch requests are handled by the external memory interface (EMIF).

To wake the I-Cache from its idle mode:

1) Deselect the idle mode by making CACHEI = 0 in ICR.

2) Execute the IDLE instruction.

6.2 Emulator Access

The software emulator can read the contents of the I-Cache during the debug mode. The contents of the I-Cache are *not* modified by emulator read operations.

6.3 Effect of Setting a Software Breakpoint

During emulation, If you set or remove a software breakpoint at an instruction, the corresponding line in the I-Cache is automatically invalidated.

6.4 Reconfiguration Required After a DSP Reset

After a DSP reset, the I-Cache is not automatically reconfigured for use. Make sure that your initialization code configures the I-Cache as described in section 4 (page 20) after every reset.

7 I-Cache Registers

Control of the I-Cache is maintained through a set of registers within the I-Cache. These registers are accessible at addresses in the I/O space of the DSP. For the addresses, see the TMS320VC5510 Data Manual (literature number SPRS076).

Table 4. Summary of the I-Cache Registers

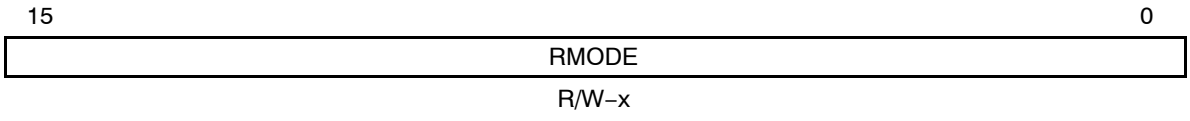
Name	Description	See ...
ICGC	Global control register. Use this register to select the number of active RAM sets.	Page 25
ICWC	Way control register. Use this register to initialize the logic for the 2-way cache.	Page 26
ICRC1	RAM set 1 control register. Use this register to initialize the logic for RAM set 1 and to check the corresponding tag-valid flag.	Page 27
ICRC2	RAM set 2 control register. Use this register to initialize the logic for RAM set 2 and to check the corresponding tag-valid flag.	Page 27
ICRTAG1	RAM set 1 tag register. Use the register to define the 12-bit tag for RAM set 1.	Page 28
ICRTAG2	RAM set 2 tag register. Use this register to define the 12-bit tag for RAM set 2.	Page 28
ICST	Status register. Use this register to verify that the I-Cache is enabled before you write to either of the RAM set tag registers.	Page 29

7.1 I-Cache Global Control Register (ICGC)

The I-Cache supports one 2-way cache and zero, one, or two RAM sets. Before enabling the I-Cache, use the global control register (ICGC) to select the desired RAM-set mode.

The values shown for RMODE in Table 5 are the only valid values that you can write to ICGC. Do not write other values to this register.

Figure 8. I-Cache Global Control Register (ICGC)



Legend: R = Read; W = Write; -x = Value after reset is not defined

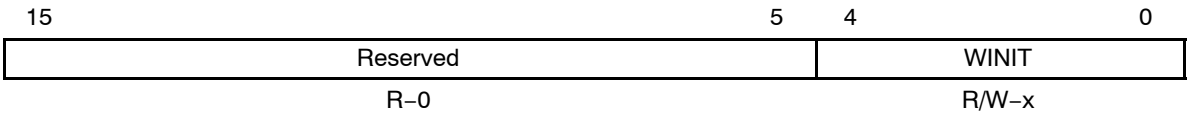
Table 5. I-Cache Global Control Register (ICGC) Bits

Bit	Field	Value	Description
15–0	RMODE		RAM-set mode bits
		CBFFh	No RAM sets
		CE1Fh	1 RAM set. Only RAM set 1 is available.
		CFFFh	2 RAM sets. RAM set 1 and RAM set 2 are available.

7.2 I-Cache Way Control Register (ICWC)

You must initialize the logic for the 2-way cache before you enable the I-Cache. To perform the initialization, write 000Fh to the way control register (ICWC). Do not write any value other than 0Fh to the WINIT field of ICWC.

Figure 9. I-Cache Way Control Register (ICWC)



Legend: R = Read; W = Write; -*n* = Value after reset; -x = Value after reset is not defined

Table 6. I-Cache Way Control Register (ICWC) Bits

Bit	Field	Value	Description
15–5	Reserved		These read-only bits are not used.
4–0	WINIT	0Fh	Way initialization bits. Make WINIT = 0Fh to initialize the logic for the 2-way cache.

7.3 I-Cache RAM Set Control Registers (ICRC1 and ICRC2)

Each RAM set control register contains an initialization field and a tag-valid bit.

Initialization field (RxINIT). If you have selected one or two RAM sets with the global control register, you must initialize the logic for RAM set 1 before you enable the I-Cache. If you have selected two RAM sets with the global control register, you must also initialize the logic for RAM set 2. To perform the initialization for each RAM set, write 000Fh to its RAM set control register. Do not write any value other than 11b to R1INIT and R2INIT.

Tag-valid bit (RxTVALID). When the I-Cache completes the process of filling RAM set 1, the I-Cache sets R1TVALID. You can poll this bit to determine when the RAM set is ready.

Figure 10. I-Cache RAM Set Control Registers (ICRC1 and ICRC2)

ICRC1

15	14	2	1	0
R1TVALID	Reserved		R1INIT	
R−0	R−0003h		R/W−x	

ICRC2

15	14	2	1	0
R2TVALID	Reserved		R2INIT	
R−0	R−0003h		R/W−x	

Legend: R = Read; W = Write; -*n* = Value after reset; -x = Value after reset is not defined

Table 7. I-Cache RAM Set 1 Control Register (ICRC1) Bits

Bit	Field	Value	Description
15	R1TVALID		RAM set 1 tag-valid bit. Check this bit to determine when the I-Cache has completed the process of filling RAM set 1.
		0	The fill is not started or is not complete.
		1	The fill is complete.
14−2	Reserved		These read-only bits are not used.
1−0	R1INIT	11b	RAM set 1 initialization bits. Make R1INIT = 11b to initialize the logic for the RAM set 1.

Table 8. I-Cache RAM Set 2 Control Register (ICRC2) Bits

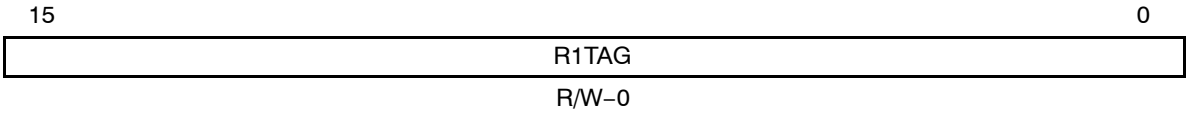
Bit	Field	Value	Description
15	R2TVALID		RAM set 2 tag-valid bit. When the I-Cache completes the process of filling RAM set 2, the I-Cache sets R2TVALID.
		0	The fill is not started or is not complete.
		1	The fill is complete.
14–2	Reserved		These read-only bits are not used.
1–0	R2INIT	11b	RAM set 2 initialization bits. Make R2INIT = 11b to initialize the logic for the RAM set 2.

7.4 I-Cache RAM Set Tag Registers (ICRTAG1 and ICRTAG2)

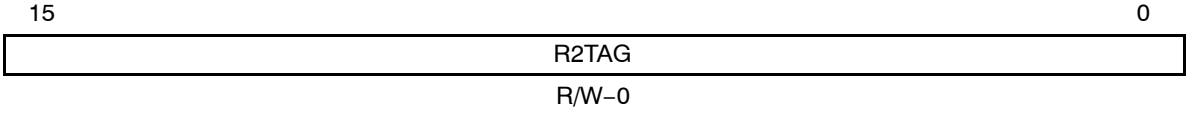
For each active RAM set (selected with the global control register), you must give the I-Cache a 12-bit tag that defines the range of addresses assigned to that RAM set. Load the tag into the appropriate RAM set tag register. Write a value with zeros in bits 15–12 and the tag in bits 11–0.

Figure 11. I-Cache RAM Set Tag Registers (ICRTAG1 and ICRTAG2)

ICRTAG1



ICRTAG2



Legend: R = Read; W = Write; -*n* = Value after reset

Table 9. I-Cache RAM Set 1 Tag Register (ICRTAG1) Bits

Bit	Field	Value	Description
15–0	R1TAG	0000h–0FFFh	RAM set 1 tag bits. Write a value with zeros in bits 15–12 and the tag in bits 11–0. This register is only applicable if you have selected one or two RAM sets with the global control register.

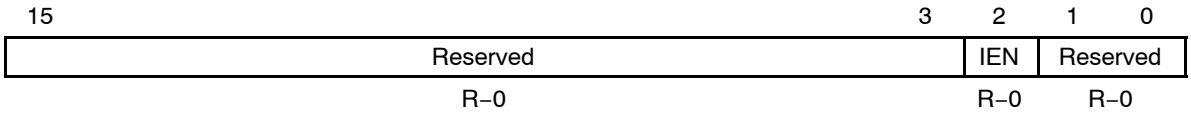
Table 10. I-Cache RAM Set 2 Tag Register (ICRTAG2) Bits

Bit	Field	Value	Description
15–0	R2TAG	0000h–0FFFh	RAM set 2 tag bits. Write a value with zeros in bits 15–12 and the tag in bits 11–0. This register is only applicable if you have selected two RAM sets with the global control register.

7.5 I-Cache Status Register (ICST)

The status register contains the IEN bit to indicate when the I-Cache is enabled. When you send an enable request to the I-Cache (CAEN = 1 in the CPU status register ST3_55), poll for IEN = 1 before writing to either of the RAM set tag registers.

Figure 12. I-Cache Status Register (ICST)



Legend: R = Read; -*n* = Value after reset

Table 11. I-Cache Status Register (ICST) Bits

Bit	Field	Value	Description
15–3	Reserved		These read-only bits are not used.
2	IEN		I-Cache-enabled bit. When you send an enable request to the I-Cache, poll for IEN = 1 before writing to either of the RAM set tag registers.
		0	The I-Cache is disabled.
		1	The I-Cache is enabled.
1–0	Reserved		These read-only bits are not used.

This page is intentionally left blank.

Revision History

This document was revised to SPRU576D from SPRU576C, which was released in August 2003.

Scope: This document has been reviewed for technical accuracy; the technical content is up-to-date as of the specified release date.

This page is intentionally left blank.

Index

L

M

N

O

P

R

S

T

W