TMS320C6472/TMS320TCI6486 DSP Shared-Memory Controller (SMC)

User's Guide

TEXAS INSTRUMENTS

Literature Number: SPRUEG5D

December 2006–Revised September 2010

List of Figures

List of Tables

Read This First

About This Manual

This document describes the shared-memory controller (SMC) in the TMS320C6472/TMS320TCI6486 digital signal processors (DSPs).

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the C6000™ devices and related support tools. Copies of these documents are available on the Internet. *Tip:* Enter the literature number in the search box provided at www.ti.com.

SPRU189 — *TMS320C6000 DSP CPU and Instruction Set Reference Guide.* Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C6000 digital signal processors (DSPs).

SPRU198 — *TMS320C6000 Programmer's Guide.* Describes ways to optimize C and assembly code for the TMS320C6000™ DSPs and includes application program examples.

SPRU301 — *TMS320C6000 Code Composer Studio Tutorial.* Introduces the Code Composer Studio™ integrated development environment and software tools.

SPRU321 — *Code Composer Studio Application Programming Interface Reference Guide.* Describes the Code Composer Studio™ application programming interface (API), which allows you to program custom plug-ins for Code Composer.

SPRU871 — *TMS320C64x+ Megamodule Reference Guide.* Describes the TMS320C64x+ digital signal processor (DSP) megamodule. Included is a discussion on the internal direct memory access (IDMA) controller, the interrupt controller, the power-down controller, memory protection, bandwidth management, and the memory and cache.

User's Guide
SPRUEG5D–December 2006–Revised September 2010

C6472/TCI6486 SMC

1 Introduction

1.1 Purpose of the Peripheral

The SMC is interfaced to the C64x+ megamodule through the L2 controller. SMC access performance is a very important consideration in determining the performance of the C64x+ megamodule that uses the SMC. The SL2 memory is used for both data and program accesses. Unlike local L2, SL2 sleep/wakeup is controlled by the SMC. The SMC operates at half the CPU's frequency (SYSCLK9).

1.2 Features

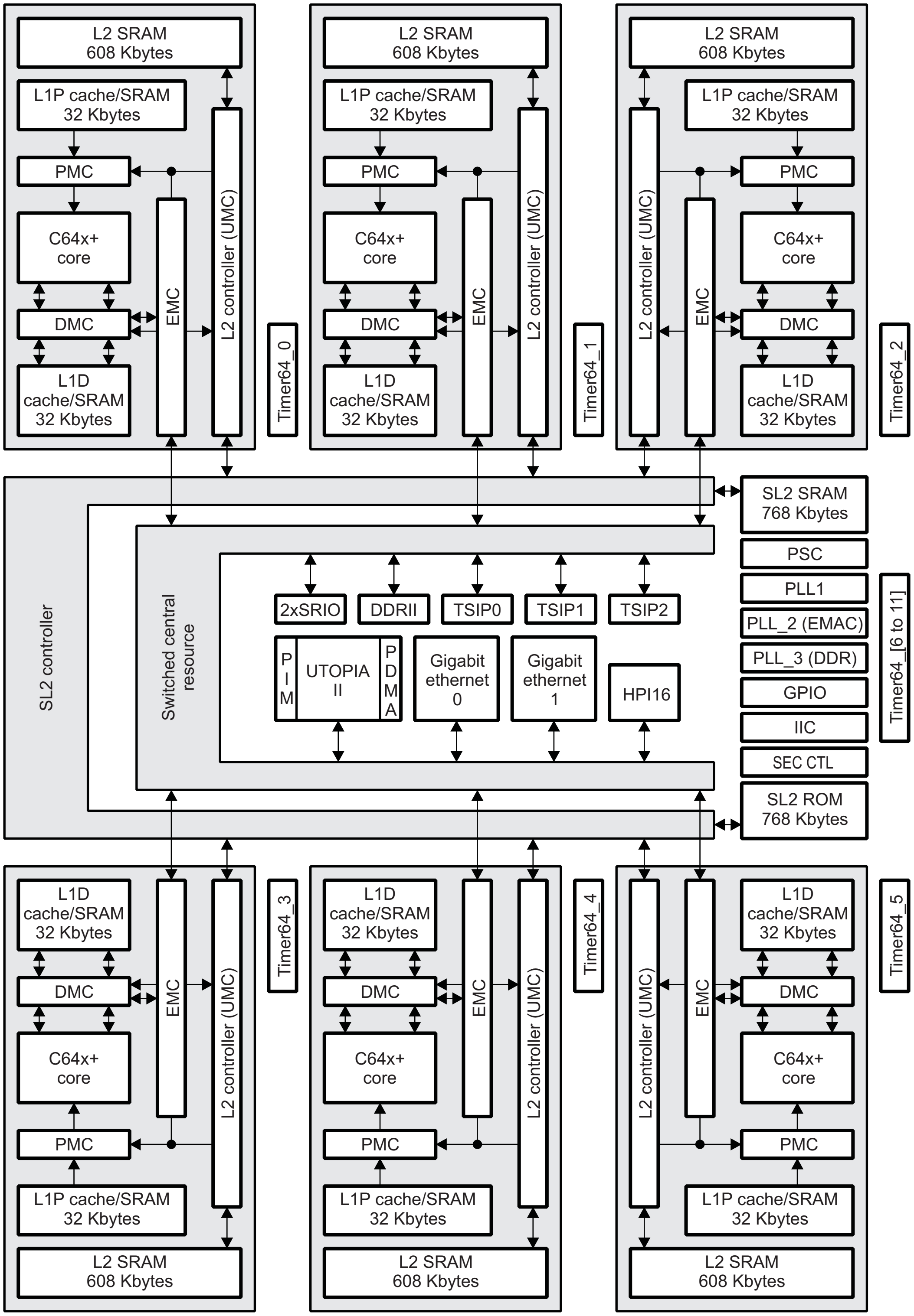
The SMC has the following features:

- Optimized for prefetchable memory reads
  - Speculative prefetching
  - 0 wait state prefetch hits at max CPU clock = 500 MHz
  - Selectable prefetchable and non-prefetchable memory region
- LRU-based arbitration per bank ARB
- Supports L2 controller to UMAP1 pipeline-Ack handshake-based control interface
- No memory protection support on SMC (present in L2 controller)
- Optimized for single streamer writes
- Atomic access support
- Power-down support

1.3 Functional Block Diagram

Figure 1 shows the block diagram for the C6472/TCI6486 base design. All the accesses to the SMC go through the L2 controller.

Figure 1. C6472/TCI6486 Block Diagram



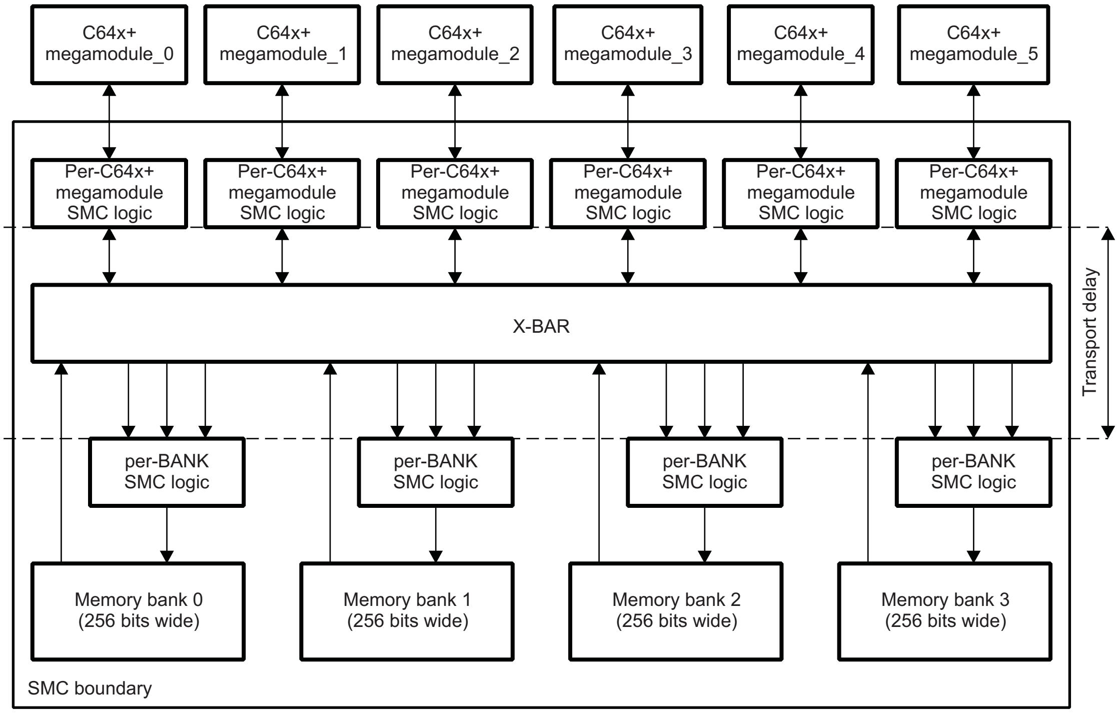
2 Shared-Memory Controller Architecture

SMC access performance is a very important consideration in determining the performance of C64x+ megamodule that uses the SMC. In order to allow accesses from multiple CPUs to and from the SL2 memory to occur as seamlessly as possible, the arbitration scheme within the SL2 controller is critical.

The SMC architecture implements the *speculative prefetch mechanism* to prefetch the future prefetchable SL2 read misses before the CPU asks for these requests to be serviced. Thus a pipeline builds up which hides the SMC latencies for future prefetchable SL2 read misses.

The SMC is divided in to two logics as shown in Figure 2, one of them is per-C64x+ megamodule SMC logic that physically sits near the C64x+ megamodule boundary and other is per-BANK SMC logic that is shared by all the C64x+ megamodules connected to the SMC.

Figure 2. Shared-Memory Block Diagram



2.1 *Per-C64x+ Megamodule SMC Logic*

Per-C64x+ megamodule SMC logic uses the prefetch mechanism corresponding to that C64x+ megamodule.

The SMC implements a prefetch mechanism, primarily to support sequential program accesses. Per-C64x+ megamodule SMC logic distinguishes these from requests arriving from the L2 controller by labeling the L2 controller requests as real commands and the prefetches as prefetch commands. Prefetch commands have no actual L2 controller command associated with them at the time the SMC generates them. Rather, the SMC generates these requests speculatively in the hope that it can service future L2 controller requests with the prefetched data.

2.1.1 Per-C64x+ Megamodule SMC Logic Overview

Per-C64x+ megamodule SMC logic includes:

- Prefetch address generator. For details, see Section 2.3.2.
- Prefetch buffer look-up/fetch. For details, see Section 2.3.4.
- SMC request controller. Whenever the per-c64x+ megamodule submits the request to the bank, it is registered in the corresponding per-c64x+ megamodule SMC request controller, until the corresponding per-BANK arbitration logic sends the acknowledgement back to indicate that "it won the per-BANK arbitration" and the data is going to appear in the next cycle. It also controls where data is landed (prefetch buffer or L2 controller).
- Data select logic. Controlled by the per-c6x+ megamodule SMC request controller, the data select logic routes the correct data to the correct destination.

Per-C64x+ megamodule SMC logic is capable of holding the request in its command register or sending the request to the corresponding per-BANK SMC logic for arbitration.

2.2 Per-BANK SMC Logic

Per-BANK SMC logic contains the following blocks:

- Arbitration logic
- One command register for each per-C64x+ megamodule SMC logic attached to the SMC
- One write buffer, which is shared by all the C64x+ megamodules attached to the SMC
- Per-BANK token controller for the ownership of the write buffers

The per-BANK SMC logic performs arbitration between the per-C64x+ megamodule commands registered in it and sends the acknowledgment back to the winner through the per-C64x+ megamodule SMC logic as well as to the next per-BANK SMC logic.

2.2.1 Per-Bank Arbitration Logic

Per-BANK arbitration unit arbitrates six requests coming from six requesters (L2 controllers) using LRU.

2.3 Prefetch and Non-Prefetch Accesses in SMC

The SL2 memory is used for both data and program accesses. Most data accesses are in the form of latency-tolerant DMAs, as well as isolated data cache read and write misses. Program cache misses are heavily pipelined, highly sequential. Both program and data cache read misses are latency sensitive, but program misses are more predictable. Thus, the SMC architecture focuses on servicing data accesses with consistent latency, and on aggressively optimizing program accesses with a prefetch mechanism.

To support program accesses as well as data accesses, memories connected to the SMC are divided into two classes of storage — prefetchable and non-prefetchable. The SMC is optimized for program/data stored in the prefetchable memory region. The application must specify its prefetchable/non-prefetchable memory regions in SL2 using the SMC memory map register.

2.3.1 Process of Enabling Prefetchable Pages in SL2 Memory

SMC has an MMR register called SL2PFCPEN to enable the prefetching pages in SL2 memory. For details, see Section 5.1.

2.3.2 Per-C64x+ Megamodule Prefetch Address Generator Overview

The SMC prefetches data it predicts the L2 controller will request, before the L2 controller requests the data. This has two main benefits: First, it takes advantage of the gaps in the requests from all C64x+ megamodules, thereby making better use of the available bandwidth. Second, it reduces latency for requests that subsequently hit in the prefetch buffer by servicing those requests quickly from the prefetch buffer instead of fetching the data from the slow SL2 memories. The prefetch buffer essentially spreads out bandwidth peaks by making the same requests ahead of time.

To support this strategy, the SMC implements a *prefetch generator* for each CPU. The prefetch address generator watches the request streams from each C64x+ megamodule, and based on that request stream, decides when to issue prefetches.

A notion of prefetchable address space lies at the heart of this scheme. The SMC divides its address space into 32 equal-sized pages. These pages exactly match the page structure superimposed on the SMC by the C64x+ megamodule's error detection and correction (EDC) and memory protection features. Applications select what pages in the SMC support prefetch through a single 32-bit prefetchable page register.

The SMC logic examines the request stream arriving from a C64x+ megamodule. When it sees a read request for a prefetchable address, it remembers that address and uses it to generate a prefetch address. The prefetch address simply is one 256-bit word later than the last prefetchable read or successful prefetch this machine generated.

SMC supports 2 Mbyte SL2 memories. In order to distinguish between prefetchable and non-prefetchable address ranges, and trigger the prefetch mechanism; the SMC can divide SL2 memories equally into 32 pages (using the MS paging scheme).

2.3.3 Per-C64x+ Megamodule Prefetch Request Generation Mechanism Overview

To enable prefetch request generation, the SMC implements a bit (Prefetch_Enabled) per the C64x+ megamodule attached to the bit to tell the prefetch generation engine to start the prefetches. On SMC_RESET, this bit goes to 0 indicating prefetch is not enabled and all the per-C64x+ megamodule prefetch buffers default to invalid. Any read request from any per-C64x+ megamodule that results in prefetch-miss sets its Prefetch_Enabled bit to 1 telling its prefetch generation engine to start prefetching. The following conditions will clear the per-C64x+ megamodule Prefetch_Enabled bit to 0:

- Any write request that hits in the corresponding per-C64x+ megamodule prefetch buffer clears its Prefetch_Enabled bit to 0.
- Invalidating all C64x+ megamodules prefetch buffers using SL2PFCFLUSH register clears all per-C64x+ megamodule Prefetch_Enabled bits to 0.

The Per-C64x+ megamodule SMC logic waits for the gaps in the L2 controller's command sequence from the same C64x+ megamodule and it generates prefetch requests when the following conditions are true:

- The corresponding L2 controller has no new command for SMC.
- The speculative prefetching is enabled (Prefetch _Enabled bit is 1) in the per-C64x+ megamodule SMC logic.
- The prefetch buffer has a slot available for allocation.
- The per-C64x+ megamodule SMC logic has a valid prefetch address.

Since the prefetch request strides linearly through memory, hitting consecutive 256-bit wide banks, the generator is capable of submitting more than one request back-to-back as long as the above conditions are satisfied.

A prefetch request must arbitrate with all the other requests to the per-bank SMC logic it targets. If a real command arrives after the prefetch request has generated, then per-C64x+ megamodule prefetch look-up logic takes the appropriate actions based on the type of the real request (prefetch miss, prefetch hit/hit-wait or non-prefetchable memory access).

2.3.4 Per-C64x+ Megamodule Prefetch Buffer Look-Up/Fetch

Per-C64x+ megamodule prefetch buffer look-up/fetch logic determines whether the real-read requests (commands) coming from the corresponding C64x+ megamodule's L2 controller is hit, hit-wait, or miss in its prefetch buffers. Table 1 shows the different states of the prefetch buffer slots. When the address of the real-read request hits in the prefetch buffer, the prefetch buffer look-up logic looks at the state of the prefetch buffer slots to decide whether to service the request from the prefetch buffer or send the request to per-BANK SMC logic as a prefetch miss. After the prefetch look-up logic hit or miss result, the decision whether to start processing that real request is made by the per-C64x+ megamodule SMC request controller. The following sections explains the per-C64x+ megamodule prefetch buffer look-up/fetch actions when the request is prefetch hit, or prefetch hit-wait, or prefetch miss.

Table 1. Valid States of Each Prefetch Buffer Slot

States of Prefetchable Buffer Slots	Description
Invalid	The prefetch buffer slot is empty.
Valid (Waiting for Arbitration)	The ACK from the BANK ARB is not received. The prefetch buffer slot is valid.
Valid (Data landed)	Prefetch buffer slot is valid and Data is landed in the prefetch buffer.

2.3.4.1 Per-C64x+ Megamodule Prefetch Look-Up/Fetch Actions During Prefetch Hit

Prefetch hit means that the real command from the C64x+ megamodule L2 controller is hit in the corresponding per-C64x+ megamodule prefetch buffer and the data is loaded into that prefetch buffer slot. The request is serviced in the same cycle the request is received.

If no other read commands are in flight for this L2 controller as signaled by the per-C64x+ megamodule SMC request controller, per-C64x+ megamodule SMC logic can simply direct the data select logic to sample the prefetched data on the next cycle.

If the L2 controller has a read command in flight as signaled by per-C64x+ megamodule SMC request controller, the prefetch lookup logic delays the prefetch-hit to allow the previous read to complete. (In flight writes from the L2 controller do not delay prefetch hits.) This ensures that from the standpoint of the L2 controller, all L2 controller requests complete in order. When the prefetch look-up logic stalls a prefetch hit in this manner, the per-C64x+ megamodule SMC request controller prevents the SMC from considering new real/prefetch commands.

When there is a hit in the prefetch buffer slots, then that entry and all older entries get marked as invalid.

2.3.4.2 Per-C64x+ Megamodule Prefetch Look-Up/Fetch Actions During Prefetch Hit-Wait

Prefetch hit-wait means that the real command from the C64x+ megamodule L2 controller has hit in the prefetch buffer slot, but the data has not loaded in that prefetch buffer slot. The prefetch buffer slot, prefetch hit-wait state, as shown in Table 1, shows the prefetch status request.

The prefetch slot, prefetch hit-wait status is: Waiting for Arbitration.

The per-C64x+ megamodule SMC request controller controls the prefetch look-up logic to arrange for the L2 controller to read the prefetched hit-wait data, based on the SMC status.

When the request is prefetch hit-wait, if the prefetch hit-wait buffer slot state is waiting for ARB, then one of the following occurs:

- If the per-C64x+ megamodule SMC request controller indicates **no** real-read request in the SMC, then the corresponding per-C64x+ megamodule prefetch look-up/fetch logic performs two tasks for this single prefetch hit-wait real request:
  - Converts the prefetch request into a real request inside the SMC logic and invalidates that prefetch slot.
  - Sends a qualifier to the corresponding per-BANK SMC logic, to which the prefetch request was submitted, to update the priority of that prefetch request to real request unconditionally (that is, no dependencies on the ACK from previous per-bank SMC logic).
- If the per-C64x+ megamodule SMC logic indicates there **is** a real-read request in the SMC, then one of the following occurs:
  - If this request is in sequence with the previous real request, then the corresponding per-C64x+ megamodule prefetch look-up/fetch logic performs two tasks for this single prefetch hit-wait real request:
    - Converts the prefetch request into a real request inside the per C64x+ megamodule SMC request controller and invalidates that prefetch slot.
    - Sends a qualifier to the corresponding per-BANK SMC logic, to which the prefetch request was submitted, to update the priority of that prefetch request to a real request with the condition.
  - If this request is **not** in sequence with the previous real request, then hold that request in the per-C64x+ megamodule SMC request controller until the ARB ACK for all the previous requests are received. After receiving all the outstanding real acknowledgments, then send the request as prefetch-miss to the corresponding per-BANK SMC logic.

2.3.4.3 Per-C64x+ Megamodule Prefetch Look-Up/Fetch Actions During Prefetch Miss

A prefetch miss indicates that the real-read request does not exist in the prefetch buffer and you need a real memory read command.

When the real L2 controller request is prefetch miss, then on of the following occurs:

- If the corresponding per-C64x+ megamodule SMC request controller indicates that there is no real request in the SMC, then submit this prefetch miss request unconditionally (that is, no dependencies on the ACK from previous per-bank SMC logic).
- If the corresponding per-C64x+ megamodule SMC request controller indicates that there is/are real requests outstanding from this C64x+ megamodule in the SMC, then:
  - If this prefetch miss request is in sequence with the previous real request and no real request from the same C64x+ megamodule is pending in the targeted bank, then submit the request with the appropriate condition. (Conditions are stated in Section 2.3.3.)
  - If this prefetch miss request is not in sequence with the previous outstanding real request/s, then hold this request in the per-C64x+ megamodule SMC logic until the ACKs for all outstanding real requests are received by the per-C64x+ megamodule SMC logic (that is, in the per-C64x+ megamodule SMC logic).

If this real request is for prefetchable memory and miss in the prefetch buffer then mark all the prefetch buffer descriptors as invalid.

2.3.5 Summary

Table 2. Summary of Prefetch and Non-Prefetch Accesses

Type of Access	Explanation	Action
In sequence prefetch hit	The real-read request is hit in the prefetch buffer and in sequence (256-bit aligned address) with the previous real-read request.	Service the request from the prefetch buffer and invalid that slot in the prefetch buffer
Out of sequence prefetch hit	The real-read request is hit in the prefetch buffer but not in sequence (256-bit aligned address) with the previous real-read request.	Service the request from the prefetch buffer and invalid that slot as well as all old entries in the prefetch buffer.
In sequence prefetch hit-wait	The real-read request is hit-wait in the prefetch buffer and in sequence (256-bit aligned address) with the previous real-read request.	Convert the prefetch request to real request in the SMC logic, send an upgrade signal to the corresponding per- BANK SMC logic and invalidate the corresponding prefetch buffer slot.
Out of sequence prefetch hit-wait	The real-read request is hit-wait in the prefetch buffer but not in sequence (256-bit aligned address) with the previous real-read request.	Service the request as prefetch miss.
Prefetch miss	The request is to the prefetchable SL2 memory region and misses in the prefetch buffer.	The C64x+ megamodule prefetch buffers and submit the request to the targeted per-BANK SMC logic. See Section 2.3.4.3 for more information.
Non-prefetchable read	The request is to the non-prefetchable SL2 memory region.	Invalidate the corresponding per-C64x+ megamodule prefetch buffers, kill any outstanding prefetch requests in SMC from same C64x+ megamodule and submit the request to the corresponding per-BANK SMC logic.
Write miss	The write request misses in the prefetch buffer.	See Section 2.4.
Write hit	The write request hits in the prefetch buffer.	Invalidate the corresponding per-C64x+ megamodule prefetch buffers, kill any outstanding prefetch requests from the same C64x+ megamodule and perform the tasks specified in Section 2.4.

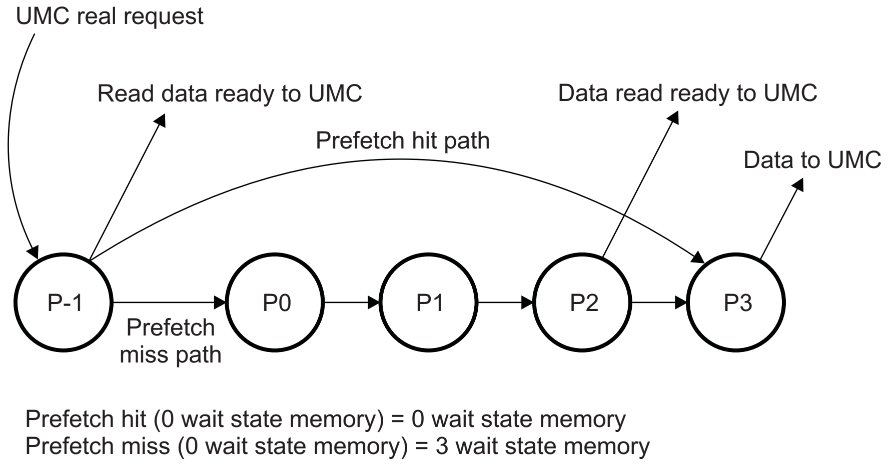
2.3.6 Invalidating 6 per-C64x+ Megamodule SMC Prefetch buffers in SMC

In order to invalidate the per-C64x+ megamodule prefetch buffers of six C64x+ megamodules connected to SMC, there is the single-bit SL2PFCFLUSH register. For details, see Section 5.2.

2.4 Handling Read Requests to SMC From Six C64x+ Megamodules

The control interface between the UMC and SMC is pipeline-Ack based. The UMC registers the request to the SMC in the beginning of the P-1 pipeline stage. Pipeline P0 is the transport delay. Pipeline P1 is the arbitration stage. Pipeline P2 is the memory access. During pipeline stage P3, the data gets routed to the destination: either CPU or prefetch buffer. These stages are shown in Figure 3.

Figure 3. SMC Pipeline Stages for Reads



P-1 read pipeline stage (shared by UMC pipeline stage):

- At this stage the corresponding per-C64x+ megamodule SMC compares the address of the incoming read request with the content of the prefetch buffers. If the address hits in the per-C64x+ megamodule SMC then the read request gets serviced from the prefetch buffer as a 0 wait state memory access. If the address does not match the request, it is marked as a miss and serviced from the share L2 memory as a 3-wait-state memory. If the request is a prefetch miss, then it resets the prefetch mechanism of that C64x+ megamodule in its per-C64x+ megamodule SMC.
- If there is no read request, then the per-C64x+ megamodule SMC generates the prefetch requests to fill the prefetch buffers.

P0 read pipeline stage:

- This pipeline stage is the transport delay from the per-C64x+ megamodule SMC to the per-BANK SMC.

P1 read pipeline stage:

- At this pipeline stage the per-BANK SMC does the arbitration of the all the read requests to it from different C64x+ megamodules (interfaced to SMC).

P2 read pipeline stage:

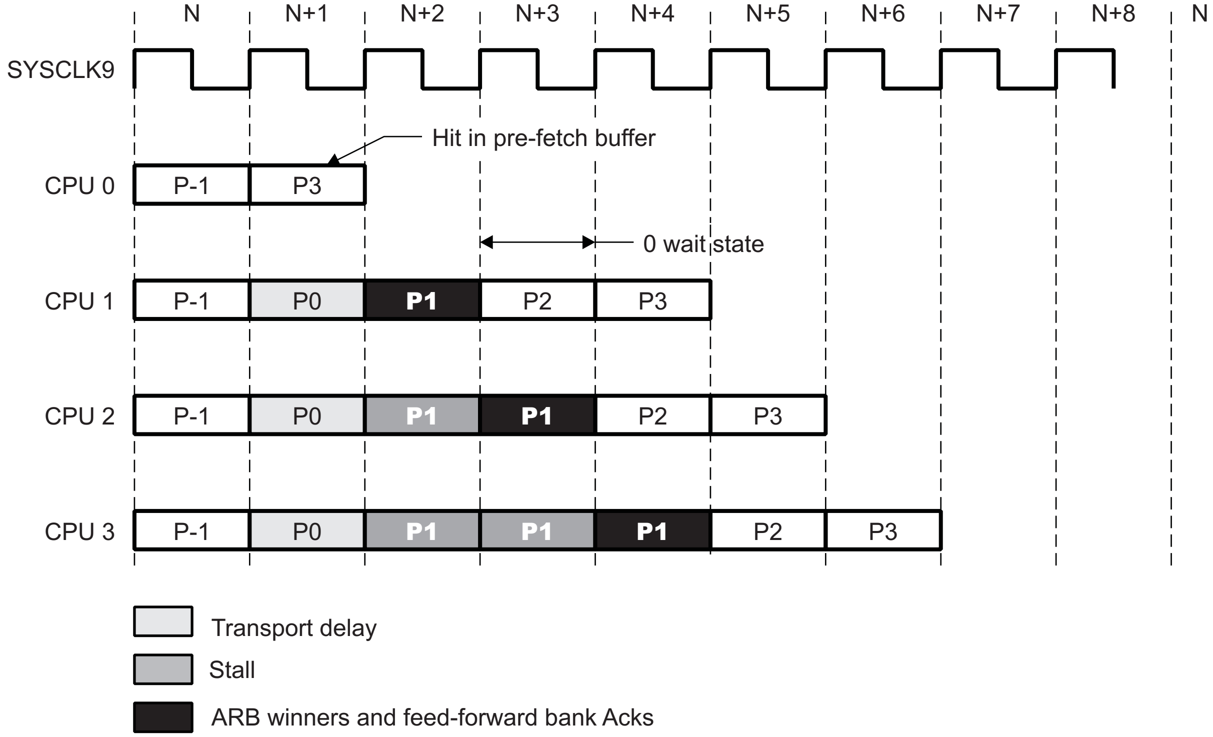
- This pipeline stage represents the cycle at which a read request is presented to 0 wait state shared L2 memory.

P3 read pipeline stage:

- This pipeline represents the transport delay for the data to travel from shared L2 memory to the C64x+ megamodule that generated the read request.

Figure 4 is based on the pipeline stages (P-1, P0, P1, P2 and P3). The UMC registered the request to the SMC in the beginning of P-1 pipeline stage. Figure 4 shows CPU0 to CPU3 accessing BANK0, CPU0 prefetch hit, and CPU1 to CPU3 prefetch miss to Bank0.

Figure 4. CPU 1 to CPU 3 Prefetch Miss and CPU 0 Prefetch Hit to Bank 0



2.5 Handling Write Requests to SMC from Six C64x+ Megamodules

For write requests, you need sufficient total buffering so that the following conditions are met:

- A single C64x+ megamodule should be able to stream writes at full bandwidth if there is no other activity in SMC. This is necessary for chips that use the SMC for data.
- A single CPU streaming write into SMC should not adversely impact reads from other CPUs, especially reads that hit prefetch buffers.

To fulfill the above two requirements, the token controller scheme was implemented for handling writes from C64x+ megamodules. In this scheme, storage of one write buffer in per-C64x+ megamodule SMC logic and one write buffer in per-BANK SMC logic is used. The write buffer in per-BANK SMC logic is shared by six C64x+ megamodules connected to the SMC. Token-based protocol is used to decide the owner of the shared per-BANK SMC write buffer.

When there is a write from any C64x+ megamodule, its corresponding per-C64x+ megamodule logic registers write into its write buffer and send a signal to the targeted per-BANK SMC logic for the token for the write buffer to register the write request into the targeted per-BANK SMC logic. When per-BANK SMC logic receives a request for one or more tokens, it picks one of the requests using its own arbitration scheme (write LRU) and sends the token acknowledgment to the corresponding per-C64x+ megamodule SMC logic. In addition, it feeds forward a token request to the next sequential per-BANK SMC for the same C64x+ megamodule. When there is no write request, or an out-of-sequence write request from the C64x+ megamodule that triggered the feed-forward token request, then the feed-forward requests should be terminated. Once per-C64x+ megamodule logic receives the token for the write buffer, it forwards the request to the targeted per-BANK SMC logic for arbitration. In per-BANK SMC arbitration logic, a real-write request has higher priority over a real-read request from any per-C64x+ megamodule, thus the token controller can keep the token valid for only one SYSCLK9 cycle. A C64x+ megamodule that is granted access to the per-BANK SMC logic write buffer uses it only for one cycle. The write completes in one cycle, since the write is given highest priority by the per-BANK SMC arbitration logic over reads.

Full bandwidth should be guaranteed only for writes that are in sequence and come in consecutive cycles. Full bandwidth for out-of-sequence writes or gaps between write requests even though they are in sequence is not required.

Update the write LRU only when the token is used (the write buffer gets written) and do not update the read LRUs. For design simplification, write LRU gets updated once the token is granted.

Any streams of writes have a startup overhead, but once it goes, it can keep going.

In the C6472/TCI6486 device, prioritizing write over reads is allowed. To lock two C64x+ megamodules completely from SMC, there need to be remaining four streamers in sequence and that to be perfect forward strides of 256-bit, which is highly unlikely.

For simplicity, token requests to the targeted per-BANK SMC logic are submitted when no outstanding real-read request(s) from the same C64x+ megamodule exists in the SMC.

2.6 Coherency Between C64x+ Megamodules Connected to SMC

The coherency within the C64x+ megamodule and its per-C64x+ megamodule SMC logic inside SMC needs to be maintained. SMC does not support coherency among the C64x+ megamodules and the per-C64x+ megamodule SMC logic inside each megamodule.

To maintain the coherency within the C64x+ megamodule and its per-C64x+ megamodule SMC logic, per-C64x+ megamodule prefetch look-up is performed on the write requests as well. If the write request's address hits in its corresponding per-C64x+ megamodule prefetch buffer, then invalidate all the entries of that prefetch buffer only.

Software needs to handle the coherency among the C64x+ megamodules using locks/interrupts. To maintain the coherency among the per-C64x+ megamodule SMC logics, there are software-programmable memory-mapped registers to invalidate all the per-C64x+ megamodule prefetch buffers, as explained in Section 2.3.6.

2.7 Resets and Clocks

SMC must get the CPU/2 and CPU/3 clocks from the CPUs PLL controller. On getting reset signal, SMC should reset all the logic and registers to default. SMC gets one reset signal (smc_reset) from the CPUs PLL controller. It is the job of the chip integrator to ORed the following resets and sends only one reset to SMC.

- Power on reset (hard)
- Warm reset (using external pin)
- All other global resets including global emulation if any C64x+ megamodule's local reset/s should not go to SMC.

2.8 Shared L2 Memory Bank

The SL2 memory appears as level-2 memory in the local address space of each attached C64x+ megamodule. This shared L2 RAM memory cannot be configured as cache. The shared L2 memory is cacheable by any of the L1 Data and Program Caches.

2.8.1 SL2 Memory Aliasing

The SMC is aliased for the reserved/non-existent SL2 memory location to avoid any lock-ups between the L2 controller to SMC pipeline-Ack based interface. Reads to reserved/non-existent SL2 memory return the data from the aliased address to avoid any lock-up situation. Writes to reserved/non-existent SL2 memory result in memory corruption in the aliased location unless memory protection is used in the L2 controller.

2.8.2 Shared L2 Memory Map

The SMC provides access to one contiguous memory (256K bytes, 512K bytes, 1M bytes or 2M bytes) and does not distinguish between different memories (RAM or ROM) attached to it. SMC does aliasing for the reserved/non-existent SL2 memory locations to avoid any lock-ups between SMC and L2 controller pipeline-Ack based interface.

2.8.3 Endianness

The same program image can run from the same ROM in little endian or big endian. Any endian dependencies arise only from how the software is written. In other words, PMC does nothing for endianness but DMC needs endianness.

The SMC does not have information about the device endianness. It is left to the ROM software programmer to do the following:

- Duplicate the table (data) in both endianness.
- Make the software (code) bi-endian.

3 Atomic-Access Monitor Overview

The atomic-access variables are needed for reliable, high-performance synchronization between multiple processors.

In the C6472/TCI6486 device, there is an atomic-access monitor for every per-BANK SMC logic in the SMC to avoid any contention issues when checking up to six C64x+ megamodules concurrent accesses against the monitor. Accesses to distinct banks, though, are guaranteed to never hit the same monitor address. Atomic access should go only to non-prefetchable address spaces.

The shared-memory controller supports the load-link (LL), store-link (SL), and a commit-link (CMTL) operations.

3.1 C64x+ Megamodule Atomic Instructions

The shared-memory controller supports the load-link (LL), store-link (SL), and a commit-link (CMTL) operation. These instructions are explained in detail in this section.

3.1.1 LL Instruction

When any of the C64x+ megamodules wants to lock an address (that is, tell the atomic-access monitor to monitor this address location for any other stores) and load a word from that address, it executes LL. The LL instruction is a read access to SMC and it goes to the targeted atomic-access monitor as a request to monitor that address.

When the monitor sees an LL instruction from any of the C64x+ megamodules, the monitor logs the requester C64x+ megamodule id in the CPU_ID field, logs the address in LinkAdr field, set the LinkV flag to 1 and set the LinkdtV flag to 0, irrespective of the state of the monitor.

Whatever the atomic-access monitor does with the given LL instruction, it does not affect the LL read operation (i.e., loading a word from memory) and giving the read data to the requesting C64x+ megamodule.

3.1.2 SL Instruction

SL is a write request but only to the targeted atomic-access monitor (not to the memory). The monitor decides whether to accept the SL write request or not.

When the monitor sees an SL instruction, it checks LinkV, CPU_ID and LinkAdr.

Table 3. SL Instruction

LinkV [1]	CPU_ID [2]	LinkAdr [3]	LinkdtV [4]	Action
0	0	0	X	Discard the SL request
0	0	1	X	Discard the SL request
0	1	0	X	Discard the SL request
0	1	1	X	Discard the SL request
1	0	0	X	Discard the SL request
1	0	1	X	Discard the SL request
1	1	0	X	Set LinkV to 0
1	1	1	0	Store the 32-bit data from incoming 256-bit SL data into LinkData and set LinkdtV flag to 1.
1	1	1	1	Invalidate the Link (LinkV = 0)

[1] 0 = Link invalid; 1 = Link valid

[2] 0 = SL C64x+ megamodule_ID does not match; 1 = SL C64x+ megamodule_ID matches;

[3] 0 = SL address does not match; 1 = SL address matches

[4] 0 = LinkdtV is not set; 1 = LinkdtV is set

3.1.3 CMTL Instruction

CMTL is a read request to read the LinkV flag from the targeted atomic-access monitor and trigger a write request from the monitor to the memory location.

When the monitor sees the CMTL instruction, it checks LinkV, CPU_ID, LinkAdr, and LinkdtV.

Table 4. CMTL Instruction

Link V [(1)]	CPU_ID [(2)]	LinkAdr [(3)]	LinkdtV [(4)]	Action
0	0	0	0	Discard the request and return “0”
0	0	0	1	Discard the request and return “0”
0	0	1	0	Discard the request and return “0”
0	0	1	1	Discard the request and return “0”
0	1	0	0	Discard the request and return “0”
0	1	0	1	Discard the request and return “0”
0	1	1	0	Discard the request and return “0”
0	1	1	1	Discard the request and return “0”
1	0	0	0	Discard the request and return “0”
1	0	0	1	Discard the request and return “0”
1	0	1	0	Discard the request and return “0”
1	0	1	1	Discard the request and return “0”
1	1	0	0	Set LinkV to 0 and return “0”
1	1	0	1	Set LinkV to 0 and return “0”
1	1	1	0	Set LinkV to 0 and return “0”
1	1	1	1	Set LinkV to 0, trigger write to write 32-bit LinkData into LinkAdr address and return LinkdtV flag (1) on each of the eight 32-bit lanes back to the requesting C64x+ megamodule's UMC.

[(1)] 0 = Link invalid; 1 = Link valid
[(2)] 0 = CMTL C64x+ megamodule_ID does not match; 1 = CMTL C64x+ megamodule_ID matches;
[(3)] 0 = CMTL address does not match; 1 = CMTL address matches
[(4)] 0 = Invalid; 1 = Valid

The short-lived behavior of atomic-linked variables is shown in Example 1 by implementing a shared-counter synchronization operation. Note that only the .D2 unit can perform a synchronization primitive. The .D1 unit may be used for load, store, arithmetic or logical operations when the .D2 unit is performing a synchronization primitive.

Example 1. Shared Counter

```
shared_ctr:                ; here with counter address in B8
                           ; and increment value in B9
         LL *B8, B4        ; load-linked, lock location
         NOP 4
         ADD  B4,B9, B4    ; compute the incremented value
         SL  B4, *B8       ; new value to store back
         CMTL *B8, B0      ; commit the store
         NOP 4
[!B0]    B shared_ctr      ; commit failed so try again
```

4 Shared-Memory Power-Down Support

The SMC provides support for dynamically powering down portions or all of the SL2 memory attached to it while a program is active. Programs can put pages of SL2 memory to sleep and subsequently wake them manually or by accessing them.

4.1 Shared-Memory Dynamic Power-Down Overview

SMC divides SL2 memory into four physical pages that can be powered independently. SMC provides an interface, SL2PDWAKE/SL2PDSLEEP, to allow programs to manually command pages to wake or sleep. This interface should be replicated for the number of C64x+ megamodules connected to SMC.

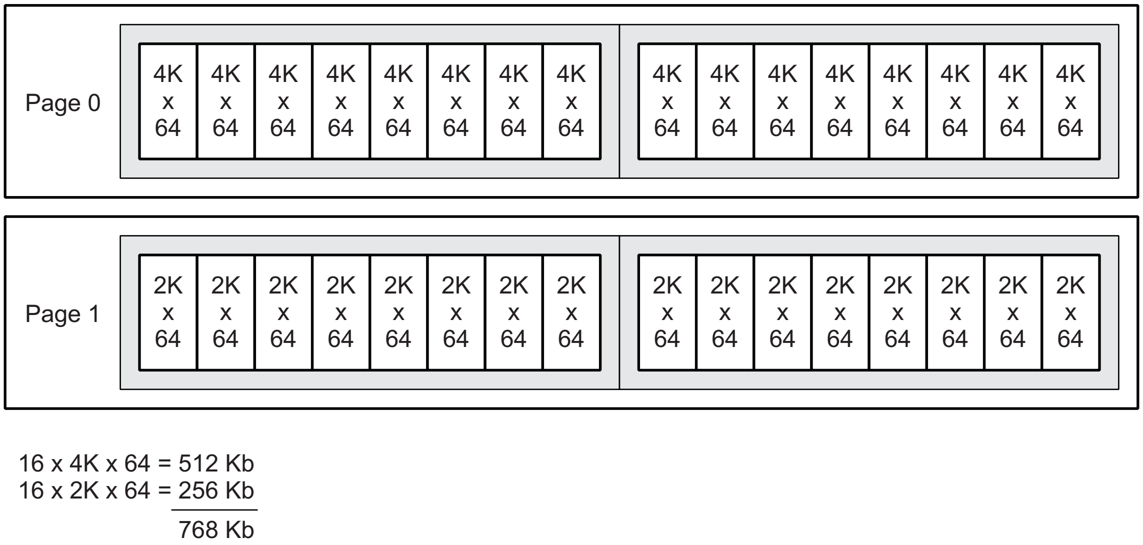
The following are the SMC responsibilities for the SL2 RAM connected to it:

- Merged awake/asleep tracking. SMC looks at the logical AND of all sleep inputs for each page from each C64x+ megamodule and use that to determine what pages may actually sleep. If any C64x+ megamodule connected to SMC wants a page awake, SMC keeps the page awake.
- Page wakeup scheduling. SMC must schedule page-wakeups in SL2 RAM among each other and with the chip-level global power-down and sleep controller (GPSC).
- Access stalling. SMC must stall accesses to sleeping physical pages while they are being awakened.

4.2 Shared-Memory Page-Oriented Dynamic Power-Down

SMC divides SL2 RAMs address space into 4 physical pages. The size of each physical page depends on the number of physical banks in SL2 RAM, and the width of the physical memory pages. In the C6472/TCI6486 device, approximately half of the SL2 memory is ROM and no power-down support is required for the SL2 ROMs. Thus, as shown in Figure 5, SMC supports power down for two physical pages (that is, RAM) in the C6472/TCI6486 device with the size of the first physical page as 64-bit × 4KB × 16 and the size of the second physical page as 64-bit × 2KB × 16.

Figure 5. C6472/TCI6486 SL2 RAM Power-Down Page Configurations



4.3 Shared-Memory Page Wake-Up Timing Considerations

SL2 page wakeups can induce power-rail transients. For this reason, there are two important parameters to consider in the memory wake-up process:

- Staggering the wakeup of physical pages. The number of pages that can safely be awakened in parallel is a physical design constraint. The SMC relies on the chip-level global power-down/sleep controller (GPSC) to stagger the wakeups between the physical pages by an appropriate number of cycles (which is programmed in GPSC). The SMC and GPSC interface is out of scope of this document and is described in a separate document.
- The number of cycles that must pass between individual physical page wakeups. The SMC does not track this count for the physical SL2 pages connected to it but it waits for the SL2 memory wrapper to signal the completion of the RAM wakeup through RAMRDY signals. The L2 controller's wake-to-access count for the physical pages connected to SMC is not used even though the SMC is connected to the L2 controller (system integrator needs to program this count to 0). In other words, the L2 controller always thinks that the SMC physical pages are awake and starts sampling data_ready. SMC wakes the page(s) in response to the following requests:
  - Program manually requests to wakeup the page(s) through the SL2PDWAKE/SL2PDSLEEP interface.
  - A C64x+ megamodule accesses the sleeping/power-down SMC physical page(s). In this access the SMC needs to delay the data_ready acknowledgement to the L2 controller as well as deassert command_ready until the page(s) is awake.

4.4 Manual Sleep Control

Programs can give permission to the SMC to put the physical page(s) to sleep, or command the page(s) to wake. When SMC has the permission from all the C64x+ megamodules connected to it, then it puts the page(s) to sleep. This allows the program to manually control what portions of SL2 memory are powered up or sleeping.

5 SMC Memory-Mapped Registers

Table 5 lists the memory-mapped registers for the SMC. For the memory address of these registers, see the *TMS320TCI6486 Communications Infrastructure Digital Signal Processor* data manual (SPRS300) or the *TMS320C6472 Fixed-Point Digital Signal Processor* data manual (SPRS612).

Table 5. SMC Memory Mapped Registers

Offset	Acronym	Register Name	See
0x0	SL2PFCPEN	SMC Prefetchable Page Enable Register	Section 5.1
0x4	SL2PFCFLUSH	SMC Prefetchable Buffer Flush Register	Section 5.2
0x8	SL2MPFSR	SMC Memory Protection Fault Status Register	Section 5.3
0xC	SL2MPFAR	SMC Memory Protection Fault Address Register	Section 5.4
0x10	SL2PDSTAT	SMC Power-Down Status Register	Section 5.5
0x100 - 0x10C	SL2AMLSTAT0-3	SMC Atomic-Access Monitor Status 0-3 Register	Section 5.6
0x120 - 0x012C	SL2AMLDATA0-3	SMC Atomic-Access Data 0-3 Register	Section 5.7
0x200 - 0x214	SL2PDSLEEP0-5	SMC Power-Down Sleep 0-5 Register	Section 5.8
0x220 - 0x234	SL2PDWAKE0-5	SMC Power-Down Wake-Up 0-5 Register	Section 5.9

5.1 SMC Prefetchable Page Enable Register

The SMC prefetchable page enable register (SL2PFCPEN) is used to indicate prefetchable/non-prefetchable SL2 memory storages (32 pages).

The SL2PFCPEN register splits the SL2 memories into 32 equal-size pages. To enable the prefetchable memory pages for the SL2 memories inside the SMC, the C64x+ megamodule needs to write 1s to the corresponding bit location in the SL2PFCPEN register.

C64x+ megamodule(s) can write to the SL2PFCPEN register only in secure supervisor mode or supervisor mode, but the register can be read in all modes (user as well as supervisor mode). If C64x+ megamodule(s) tries to write this MMR in secure user or user mode then the SMC generates an exception back to that C64x+ megamodule as well as to other C64x+ megamodules connected to it. The SMC also needs to record the latest exception status inside it. The exception status is described in Section 6.

On SMC_RESET, the SL2PFCPEN register bits values default to 0s (all SL2 memory is non-prefetchable memory).

Figure 6. SMC Prefetchable Page Enable Register (SL2PFCPEN)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAGE 31	PAGE 30	PAGE 29	PAGE 28	PAGE 27	PAGE 26	PAGE 25	PAGE 24	PAGE 23	PAGE 22	PAGE 21	PAGE 20	PAGE 19	PAGE 18	PAGE 17	PAGE 16
R/W-0															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAGE 15	PAGE 14	PAGE 13	PQGE 12	PAGE 11	PAGE 10	PAGE 9	PAGE 8	PAGE 7	PAGE 6	PAGE 5	PAGE 4	PAGE 3	PAGE 2	PAGE 1	PAGE 0
R/W-0															

LEGEND: R = Read, W = Write, n = value at reset

Table 6. SMC Prefetchable Page Enable Register (SL2PFCPEN) Field Descriptions

Bit	Field	EnumValue [(1)]	Description
31	PAGE31		PAGE31 Prefetch Enable
		0	Disable
		1	Enable
30	PAGE30		PAGE30 Prefetch Enable
29	PAGE29		PAGE29 Prefetch Enable
28	PAGE28		PAGE28 Prefetch Enable
27	PAGE27		PAGE27 Prefetch Enable
26	PAGE26		PAGE26 Prefetch Enable
25	PAGE25		PAGE25 Prefetch Enable
24	PAGE24		PAGE24 Prefetch Enable
23	PAGE23		PAGE23 Prefetch Enable
22	PAGE22		PAGE22 Prefetch Enable
21	PAGE21		PAGE21 Prefetch Enable
20	PAGE20		PAGE20 Prefetch Enable
19	PAGE19		PAGE19 Prefetch Enable
18	PAGE18		PAGE18 Prefetch Enable
17	PAGE17		PAGE17 Prefetch Enable
16	PAGE16		PAGE16 Prefetch Enable
15	PAGE15		PAGE15 Prefetch Enable
14	PAGE14		PAGE14 Prefetch Enable
13	PAGE13		PAGE13 Prefetch Enable
12	PAGE12		PAGE12 Prefetch Enable
11	PAGE11		PAGE11 Prefetch Enable

[(1)] The possible EnumValues for all bits are the same as described for the PAGE31 bit.

Table 6. SMC Prefetchable Page Enable Register (SL2PFCPEN) Field Descriptions (continued)

Bit	Field	EnumValue [(1)]	Description
10	PAGE10		PAGE10 Prefetch Enable
9	PAGE9		PAGE9 Prefetch Enable
8	PAGE8		PAGE8 Prefetch Enable
7	PAGE7		PAGE7 Prefetch Enable
6	PAGE6		PAGE6 Prefetch Enable
5	PAGE5		PAGE5 Prefetch Enable
4	PAGE4		PAGE4 Prefetch Enable
3	PAGE3		PAGE3 Prefetch Enable
2	PAGE2		PAGE2 Prefetch Enable
1	PAGE1		PAGE1 Prefetch Enable
0	PAGE0		PAGE0 Prefetch Enable

5.2 SMC Prefetchable Buffer Flush Register

SMC prefetchable buffer flush register (SL2PFCFLUSH) is a read/write register. Any C64x+ megamodule connected to the SMC can write to SL2PFCFLUSH when in secure supervisor mode to invalidate the register as well as other C64x+ megamodules prefetch buffer. C64x+ megamodule(s) can write to SL2PFCFLUSH only in secure supervisor mode or supervisor mode, but can read SL2PFCFLUSH in all modes (user as well as supervisor mode). If a C64x+ megamodule(s) tries to write to this MMR in secure user or user mode then the SMC generates an exception back to that C64x+ megamodule as well as to other C64x+ megamodules connected to it. The SMC also needs to record the latest exception status inside it. The exception status is described in Section 6.

To invalidate all the per-C64x+ megamodule prefetch buffers of all the C64x+ megamodules connected to SMC, any C64x+ megamodule (in secure supervisor or supervisor mode) can write 1 to the FLUSH bit of SL2PFCFLUSH register. This write only triggers the SMC prefetch buffer flush logic but should not latch into the SL2PFCFLUSH register. A read of the SL2PFCFLUSH register should return 0s.

Figure 7. SMC Prefetchable Buffer Flush Register (SL2PFCFLUSH)

31 16
Reserved
R-0

15 1	0
Reserved	FLUSH
R-0	R/W-0

LEGEND: R = Read, W = Write, n = value at reset

Table 7. SMC Prefetchable Buffer Flush Register (SL2PFCFLUSH) Field Descriptions

Bit	Field	EnumValue	Description
31-1	Reserved		Reserved
0	FLUSH		Prefetch Buffer Flush
		0	No effect
		1	Flush the buffer

5.3 SMC Memory Protection Fault Status Register

If any of the C64x+ megamodules connected to SMC writes to the writable registers in a mode other than supervisor mode (secure/unsecure) then the SMC sends an exception back to that C64x+ megamodule as well as to other C64x+ megamodules attached to the SMC, as shown in Figure 15. The SMC also reports the accessing C64x+ megamodule_ID into SL2MPFSR.Faulted_CPU_ID, and the accessing mode into SL2MPFSR.MODE.

The C64x+ megamodule software must handle exceptions and take appropriate action. The C64x+ megamodule software also needs to clear the fault status and fault address by writing 1 to the SL2MPFSR.CLEAR bit field. A non-supervisor mode write returns a memory protection error to the configuration bus interface, which gets propagated (as write status) to the initiating master (C64x+ megamodule).

Figure 8. SMC Memory Protection Fault Status Register (SL2MPFSR)

31–16
Reserved
R, +0000 0000 0000 0000 0000 0000 0000

15–5	4–2	1	0
Reserved	CPU_ ID	MODE	CLEAR
R, +0000 0000 0000 0000 0000 0000 0000	R +111	R +0	W +0

LEGEND: R = Read, W = Write, n = value at reset

Table 8. SMC Memory Protection Fault Status Register (SL2MPFSR) Field Descriptions

Bit	Field	EnumValue	Description
31-5	Reserved		Reserved
4-2	CPU_ID		Accessing megamodule_id
1	MODE		Mode Selection
		0	Secure
		1	Nonsecure
0	CLEAR		Clear fault status and address
		0	No effect
		1	Clear

5.4 SMC Memory Protection Fault Address Register

If any of the C64x+ megamodules connected to the SMC writes to the writable registers in a mode other than supervisor mode (secure/unsecure) then the SMC sends an exception back to that C64x+ megamodule as well as to other C64x+ megamodules attached to the SMC, as shown in Figure 18. The SMC also reports the fault access address into SL2MPFAR.

The C64x+ megamodule software handles these exceptions and takes the appropriate action.

Figure 9. SMC Memory Protection Fault Address Register (SL2MPFAR)

31	16
FAULT ADDRESS	
R, +0000 0000 0000 0000 0000 0000 0000 0000	

15	0
FAULT ADDRESS	
R, +0000 0000 0000 0000 0000 0000 0000 0000	

LEGEND: R = Read, W = Write, n = value at reset

Table 9. SMC Memory Protection Fault Address Register (SL2MPFAR) Field Descriptions

Bit	Field	EnumValue	Description
31-0	ADDR		Address

5.5 SMC Power-Down Status Register

The SMC power-down status register reports status information about the SMC's power-down logic.

The SMC provides one memory mapped register—SL2PDSTAT—that reports the current sleep-status of each SL2 memory page. This register allows the application running on C64x+ megamodules connected to the SMC to determine which pages the C64x+ megamodules have placed to sleep.

This register reports sleep mode status for both physical pages in SL2. 0s in the P3 and P2 fields in Table 10 indicate the SL2 page 1 and page 0 are in sleep mode (default) and 1s indicate SL2 page 3 and page 2 are awake.

Figure 10. SMC Power-Down Status Register (SL2PDSTAT)

31–16
Reserved
R, +0000 0000 0000 0000 0000 0000 0000 0000

15–4	3	2	1–0
Reserved	P3	P2	Reserved
R, +0000 0000 0000 0000 0000 0000 0000 0000	R, +0	R, +0	R, +0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 10. SMC Power-Down Status Register (SL2PDSTAT) Field Descriptions

Bit	Field	EnumValue	Description
31-4	Reserved		Reserved
3	P3		Physical Page 1 status
		0	Sleep mode
		1	Awake
2	P2		Physical Page 0 status
		0	Sleep mode
		1	Awake
1-0	Reserved		Reserved

5.6 SMC Atomic-Access Monitor Status Register

The SMC atomic-access monitor status register (SL2AMLSTAT0-3) is a 32-bit MMR. There is an SL2AMLSTAT0-3 register for every atomic access monitor in the SMC for link address and link status to reflect the atomic-access register.

Figure 11. SMC Atomic-Access Monitor Status 0-3 Register (SL2AMLSTAT0-3)

<table>
<tr><td>31</td><td>24</td><td>23</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">ADDR</td></tr>
<tr><td colspan="2">R, +0000 0000 0000 0000 0000</td><td colspan="2">R, +0000 0000 0000 0000 0000</td></tr>
</table>

<table>
<tr><td>15</td><td>5</td><td>4</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="2">ADDR</td><td colspan="2">CPU_ID</td><td>LINKV</td><td>LINKTV</td></tr>
<tr><td colspan="2">R, +0000 0000 0000 0000 0000</td><td colspan="2">R +000</td><td>R +0</td><td>R +0</td></tr>
</table>

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 11. SMC Atomic-Access Monitor Status 0-3 Register (SL2AMLSTAT0-3) Field Descriptions

Bit	Field	EnumValue	Description
31-24	Reserved		Reserved
23-5	ADDR		ADDR IN SMC
4-2	CPUID		CPU ID
1	LINKV		ATOMIC LINK
		0	Valid
		1	Not valid
0	LINKTV		LINK DATA
		0	Not valid
		1	Valid

5.7 SMC Atomic-Access Data Register

The SMC atomic-access monitor data register (SL2AMLDATA0-3) is a 32-bit MMR. There is an SL2AMLDATA0-3 register for every atomic access monitor in the SMC for link data.

Figure 12. SMC Atomic-Access Data 0-3 Register (SL2AMLDATA0-3)

31	0
LINKDATA	
R, +0000 0000 0000 0000 0000 0000 0000 0000	

LEGEND: R = Read, W = Write, n = value at reset

Table 12. SMC Atomic-Access Data 0-3 Register (SL2AMLDATA0-3) Field Descriptions

Bit	Field	EnumValue	Description
31-0	LINKDATA		Link Data

5.8 SMC Power-Down Sleep Register

C64x+ megamodule(s) can write to SL2PDSLEEP0-5 registers only in secure supervisor mode or supervisor mode, and a read returns 0. If a C64x+ megamodule(s) tries to write these MMRs in secure user or user mode then the SMC generates an exception back to that C64x+ megamodule as well as to other C64x+ megamodules connected to it. The SMC also needs to record the latest exception status inside it. The exception status is described in Section 6. No exceptions are generated for the reads in any mode to these registers.

To give permission to put physical pages of SL2 memory to sleep, a C64x+ megamodule write 1s to the page-sleep bits in the appropriate SL2PDSLEEP0-5 register in SMC. For instance, C64x+ megamodule_0 and C64x+ megamodule_1 decide to put SL2 page 1 to sleep then C64x+ megamodule_0 writes 0x00000002 (bit 1 set to 1) in its corresponding SL2PDSLEEP0 register and C64x+ megamodule_1 writes 0x00000002 (bit 1 set to 1) in its corresponding SL2PDSLEEP1 register. If all C64x+ megamodules give permission to put SL2 page(s) to sleep, the SMC ANDs all the permissions to decide whether to sleep the page(s) or not.

Figure 13. SMC Power-Down Sleep 0-5 Register (SL2PDSLEEP0-5)

31	16
Reserved	
R, +00 0000 0000 0000 0000 0000 0000 00	

15 4	3	2	1 0
Reserved	P3	P2	Reserved
R, +00 0000 0000 0000 0000 0000 0000 00	R/W, +0	R/W, +0	R, +0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 13. SMC Power-Down Sleep 0-5 Register (SL2PDSLEEP0-5) Field Descriptions

Bit	Field	EnumValue	Description
31-4	Reserved		Reserved
3	P3		Physical Page 1 sleep mode
		0	Disable
		1	Enable
2	P2		Physical Page 0 sleep mode
		0	Disable
		1	Enable
1-0	Reserved		Reserved

5.9 SMC Power-Down Wake-Up Register

C64x+ megamodule(s) can write to SL2PDWAKE0-5 registers only in secure supervisor mode or supervisor mode and a read returns 0. If a C64x+ megamodule(s) tries to write these MMRs in secure user or user mode then SMC generates an exception back to that C64x+ megamodule as well as to other C64x+ megamodules connected to it. The SMC also needs to record the latest exception status inside it. The exception status is described in Section 6. No exceptions are generated for the reads in any mode to these registers.

Figure 14. SMC Power-Down Wake-Up 0-5 Register (SL2PDWAKE0-5)

31 16
Reserved
R, +00 0000 0000 0000 0000 0000 0000 00

15 4	3	2	1 0
Reserved	P3	P2	Reserved
R, +00 0000 0000 0000 0000 0000 0000 00	R/W, +0	R/W, +0	R, +0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

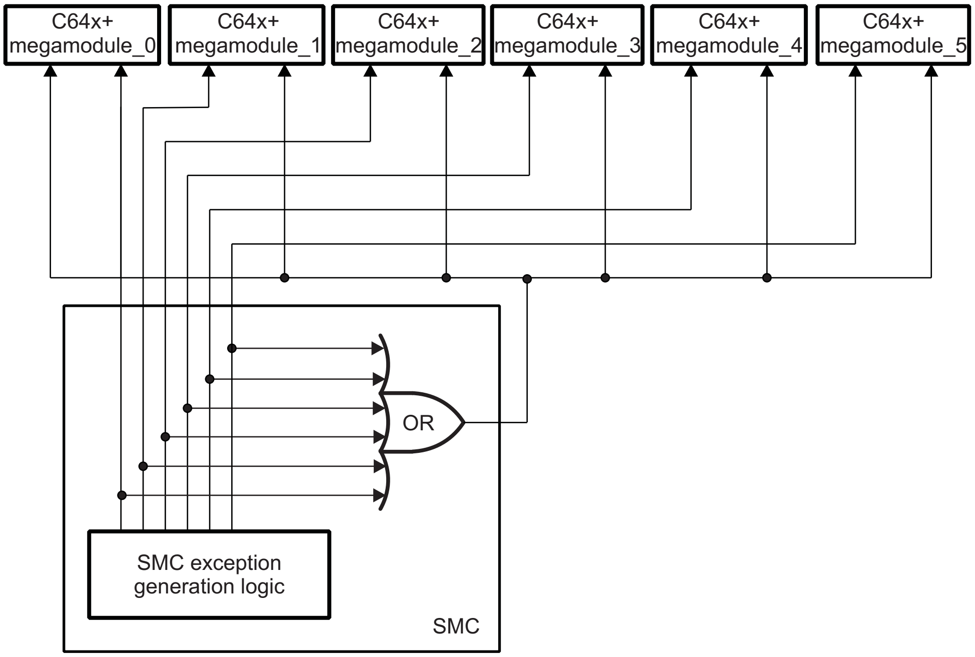
Table 14. SMC Power-Down Wake-Up 0-5 Register (SL2PDWAKE0-5) Field Descriptions

Bit	Field	EnumValue	Description
31-4	Reserved		Reserved
3	P3		Physical Page 3 WAKE
		0	Disable
		1	Enable
2	P2		Physical Page 2 WAKE
		0	Disable
		1	Enable
1-0	Reserved		Reserved

6 SMC Exception to the C64x+ Megamodules

There are two exceptions that go to each C64x+ megamodule (connected to the SMC) from the SMC, as shown in Figure 15, for six C64x+ megamodules connected to the SMC. Only one common exception goes to all six C64x+ megamodules.

Figure 15. SMC Exception Generation Logic for Six C64x+ Megamodules Connected to it

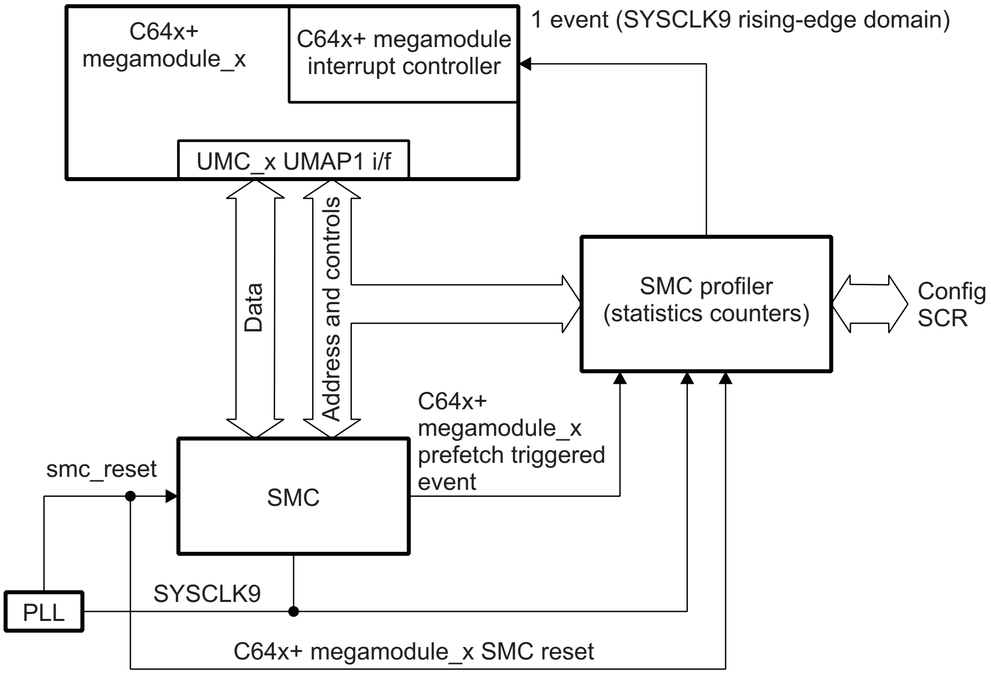


7 SMC Profiler

In multiple C64x+ megamodule devices that use shared-memory, like the C6472/TCI6486 device, the performance statistics of a single C64x+ megamodule in the shared-memory is required to optimize the device performance in the system.

The logic to calculate the performance of a C64x+ megamodule in shared-memory can be implemented in the shared-memory controller (SMC) but on the cost of SMC design and verification complexity. Alternatively, the UMAP1 pipelined-Ack interface between the C64x+ megamodule and the SMC gives us a flexibility of implementing an external SMC Profiler per C64x+ megamodule, which can be used to monitor the UMAP1 interface signals and generate the required read performance statistics for a given C64x+ megamodule to which the profiler is connected. Figure 16 illustrates how the SMC profiler coordinates with the SMC and C64x+ Megamodules.

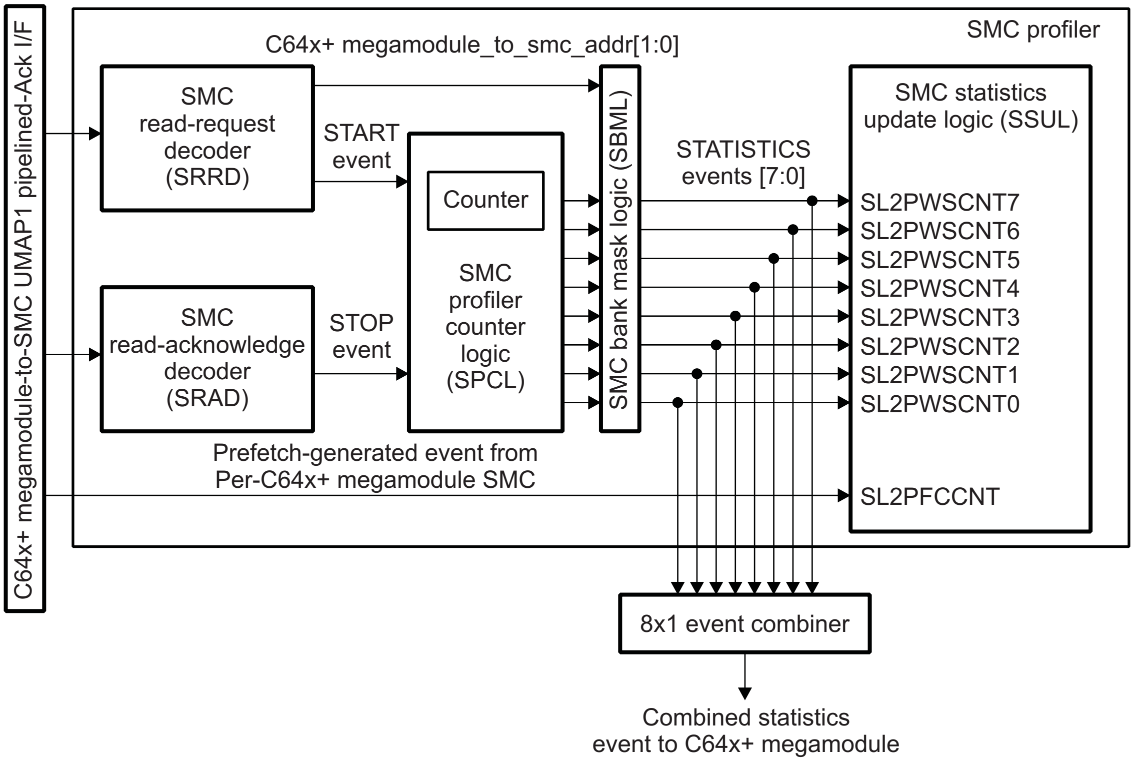
Figure 16. SMC Profiler


C64x+ megamodule_x
C64x+ megamodule interrupt controller
1 event (SYSCLK9 rising-edge domain)
UMC_x UMAP1 i/f
Data
Address and controls
SMC profiler (statistics counters)
Config SCR
C64x+ megamodule_x prefetch triggered event
smc_reset
SMC
PLL
SYSCLK9
C64x+ megamodule_x SMC reset

7.1 Details of SMC Profiler

SMC profiler logic monitors the UMAP1 pipelined-Ack interface signals and generates and stores the statistics metadata of all the SMC read requests into its memory-mapped registers. The profiler logic also sends a combined SMC statistics event to the corresponding C64x+ megamodule's AET logic.

Figure 17. SMC Profiler Block Diagram


C64x+ megamodule-to-SMC UMAP1 pipelined-Ack I/F
SMC profiler
C64x+ megamodule_to_smc_addr[1:0]
SMC read-request decoder (SRRD)
START event
SMC read-acknowledge decoder (SRAD)
STOP event
Counter
SMC profiler counter logic (SPCL)
SMC bank mask logic (SBML)
STATISTICS events [7:0]
SMC statistics update logic (SSUL)
SL2PWSCNT7
SL2PWSCNT6
SL2PWSCNT5
SL2PWSCNT4
SL2PWSCNT3
SL2PWSCNT2
SL2PWSCNT1
SL2PWSCNT0
Prefetch-generated event from Per-C64x+ megamodule SMC
SL2PFCCNT
8x1 event combiner
Combined statistics event to C64x+ megamodule

As shown in Figure 17, the SMC profiler contains five blocks:

- SMC Read Request Decoder (SRRD)
- SMC Read Acknowledge Decoder (SRAD)
- SMC Profiler Counter Logic (SPCL)
- SMC BANK Mask Logic (SBML)
- SMC Statistics Update Logic (SSUL)

7.1.1 SMC Read Request Decoder (SRRD)

This decoder snoops the required L2 controller's UMAP1 pipelined-Ack signals for any read request to SMC from C64x+ megamodule and keeps the history of the previous read acknowledgement.

SRRD has to make sure to decode only those read request which are accepted by SMC. SRRD should decode stalled read request at C64x+ megamodule boundary as No read request and should not trigger its logic.

SRRD decodes and records the read request as well as it also looks for the previous read request acknowledge from SMC to L2 controller on the UMAP1 interface. If there is a read request decoded and the previous request acknowledges has been submitted, it sends a START event to the internal SMC profiler counter logic (see Section 7.1.3) otherwise it waits for the previous read acknowledgement.

The L2 controller can have four outstanding read requests inside the SMC. SRRD needs to record all the read requests and sends the START event to the SPCL for the oldest read request recorded on getting the acknowledgement for the previous read request. The previous read request acknowledgement gating of the START event guarantees that the wait-states are not counted more than one time if SMC has more than one read request outstanding. During submitting the START event SRRD should clear the previous acknowledgement history for the new acknowledgement that will come for this submitted START event.

7.1.2 SMC Read Acknowledge Decoder (SRAD)

This decoder snoops the required UMAP1 signals for the read acknowledgement from SMC back to C64x+ megamodule. Once the read acknowledge is identified, the SRAD sends an STOP event to the SMC Profiler Counter Logic (Section 7.1.3) to stop the appropriate counter.

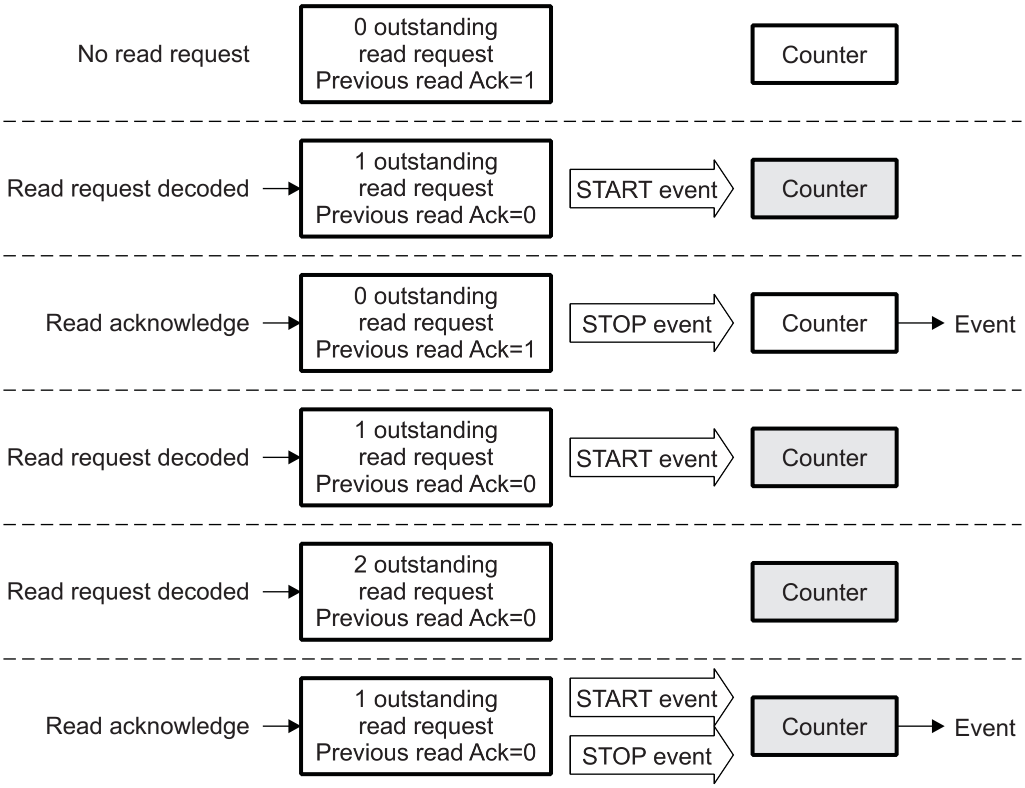
7.1.3 SMC Profiler Counter Logic (SPCL)

On getting the START event from the SRRD, SPCL triggers its internal counters (max count value =7) to count and stops them on getting the STOP event from SRAD.

The purpose of SPCL is to count the time duration of a SMC read request in terms of SMC clock cycles (SYSCLK9) which C64x+ megamodule will experience. Since one of the inputs to trigger START event in the SRRD is the acknowledgement of the previous read request, there will not be more than one START event in the SPCL at any given time and thus only one counter can be used inside SPCL module.

Figure 18 shows the operation of SPCL. On getting START triggered event the SPCL counter starts counting. When START decoded and START triggered happens at the same cycle the counter starts counting from the next cycle. But when START triggered event happens alone then the counter should starts counting after one cycle as you do not want to double count the data cycle of the previous access.

Figure 18. SPCL State Machine


No read request
0 outstanding read request Previous read Ack=1
Counter
Read request decoded
1 outstanding read request Previous read Ack=0
START event
Counter
Read acknowledge
0 outstanding read request Previous read Ack=1
STOP event
Counter
Event
Read request decoded
1 outstanding read request Previous read Ack=0
START event
Counter
Read request decoded
2 outstanding read request Previous read Ack=0
Counter
Read acknowledge
1 outstanding read request Previous read Ack=0
START event
STOP event
Counter
Event

The maximum required value of counter/s is 7. When they reach the maximum value, the count remains at the maximum value. SPCL uses the count value of the counter to select which STATISTICS event to drive as shown in Table 15.

Table 15. Mapping of Statistics Event in SPCL

Count Value	Statistics Event	Description
0	STATISTIC event[0]	Read request serviced in 0 wait state
1	STATISTIC event[1]	Read request serviced in 0 wait state
2	STATISTIC event[2]	Read request serviced in 0 wait state
3	STATISTIC event[3]	Read request serviced in 0 wait state
4	STATISTIC event[4]	Read request serviced in 0 wait state
5	STATISTIC event[5]	Read request serviced in 0 wait state
6	STATISTIC event[6]	Read request serviced in 0 wait state
7	STATISTIC event[7]	Read request serviced in 0 or more wait state

7.1.4 SMC BANK Mask Logic (SBML)

This logic sets the required UMAP1 signals and masks the read request for a particular bank or all banks to calculate the total bank conflicts for a given C64x+ megamodule in one or more banks inside SMC.

In order to give you programming flexibility to measure the C64x+ megamodule performance on a particular bank of SMC, the SL2BNKMSK register (Figure 19) inside the SMC BANK mask logic (SBML) can be used.

SBML decodes the STATISTICS event and its address from SPCL using the programmed SL2BNKMSK register to decide whether to filter this request or let it go to the SSUL unit. The description of SL2BNKMSK is given in Figure 21 and Table 18.

7.1.5 SMC Statistics Update Logic (SSUL)

This logic updates the SMC profiler statistics registers by taking events from SBML and SMC. This logic also generates the appropriate events for the corresponding C64x+ megamodule's AET.

The functionality of the SSUL unit is to record the STATISTICS events from SBML into the corresponding statistics register (SL2PWSCNTx).

SSUL also takes a prefetch-generated event directly from SMC and records it in the SL2PFCCNT register.

SSUL also sends these STATISTICS events to the 8 to 1 event combiner. The single event output of the 8 to 1 event combiner is routed to C64x+ megamodule for doing AET tracing.

The events which SSUL sends to the 8 to 1 event combiner are given in Table 16.

Table 16. SSUL Events to Event Selector Logic

Statistics Event	Description
WS0_EVENT	Indicates the SMC read request was 0 wait state read request
WS1_EVENT	Indicates the SMC read request was 1wait state read request
WS2_EVENT	Indicates the SMC read request was 2 wait state read request
WS3_EVENT	Indicates the SMC read request was 3 wait state read request
WS4_EVENT	Indicates the SMC read request was 4 wait state read request
WS5_EVENT	Indicates the SMC read request was 5 wait state read request
WS6_EVENT	Indicates the SMC read request was 6 wait state read request
WS7_EVENT	Indicates the SMC read request was 7 wait state read request

7.2 SMC Profiler Usage

Figure 19 and Figure 20 are timing diagrams that show how the SMC profiler should gather the corresponding C64x+ megamodule statistics in the SMC.

This terminology is used in the timing diagrams:

Start(x) decoded— Read request decoded by SRRD logic.

Start triggered— "Start (x) decoded event" is triggered by SRRD logic when one of the following occurs:

- No outstanding "Start (x-1)" events in SPCL
- The read acknowledge indicated by Stop (x-1) from SRAD is received.

Stop (x)— Read acknowledgement is decoded by SRAD logic.

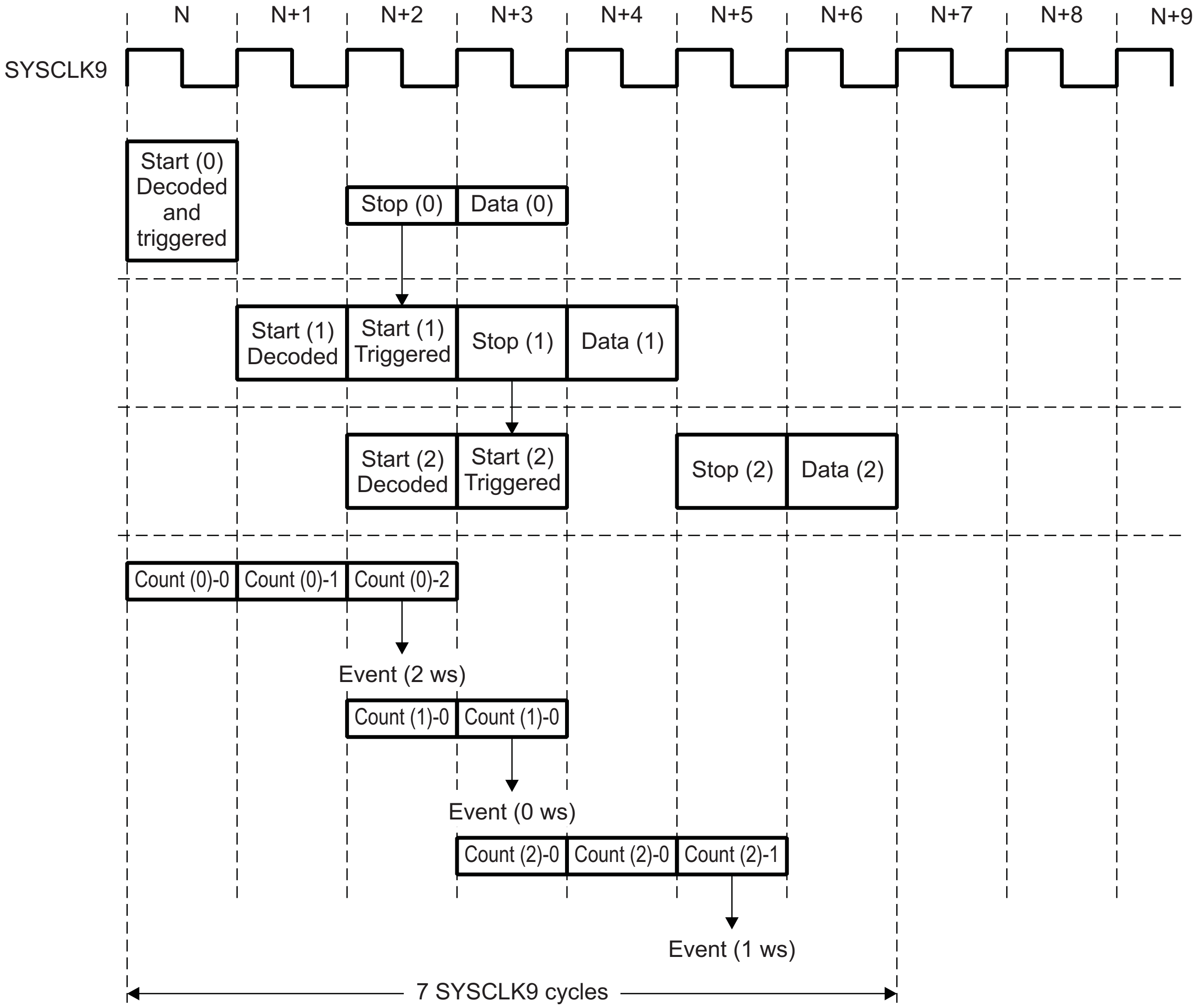
Count(x)— The SPCL counter value

Events(x)— Statistics event number "x" generated by SPCL for SSUL.

This section describes three back-to-back requests from CPU0 to the SMC. Figure 19 shows two hit waits and one miss request. Figure 20 shows three prefetch hit requests.

Figure 19. Profiling Three Back-to-Back Requests From CPU_0 to SMC (2 Hit Waits and 1 Miss)

- Cycle N: Request (0) comes to SMC from CPU_0 (Prefetch hit wait)
- Cycle N+1: Request (1) comes to SMC ROM CPU_0 (Prefetch hit wait)
- Cycle N+2: Request (2) comes to SMC from CPU_0 (Prefetch miss)



- Events:
  - 2 ws event (@ N+2)
  - 0 ws event (@ N+3)
  - 1 ws event (@ N+5)

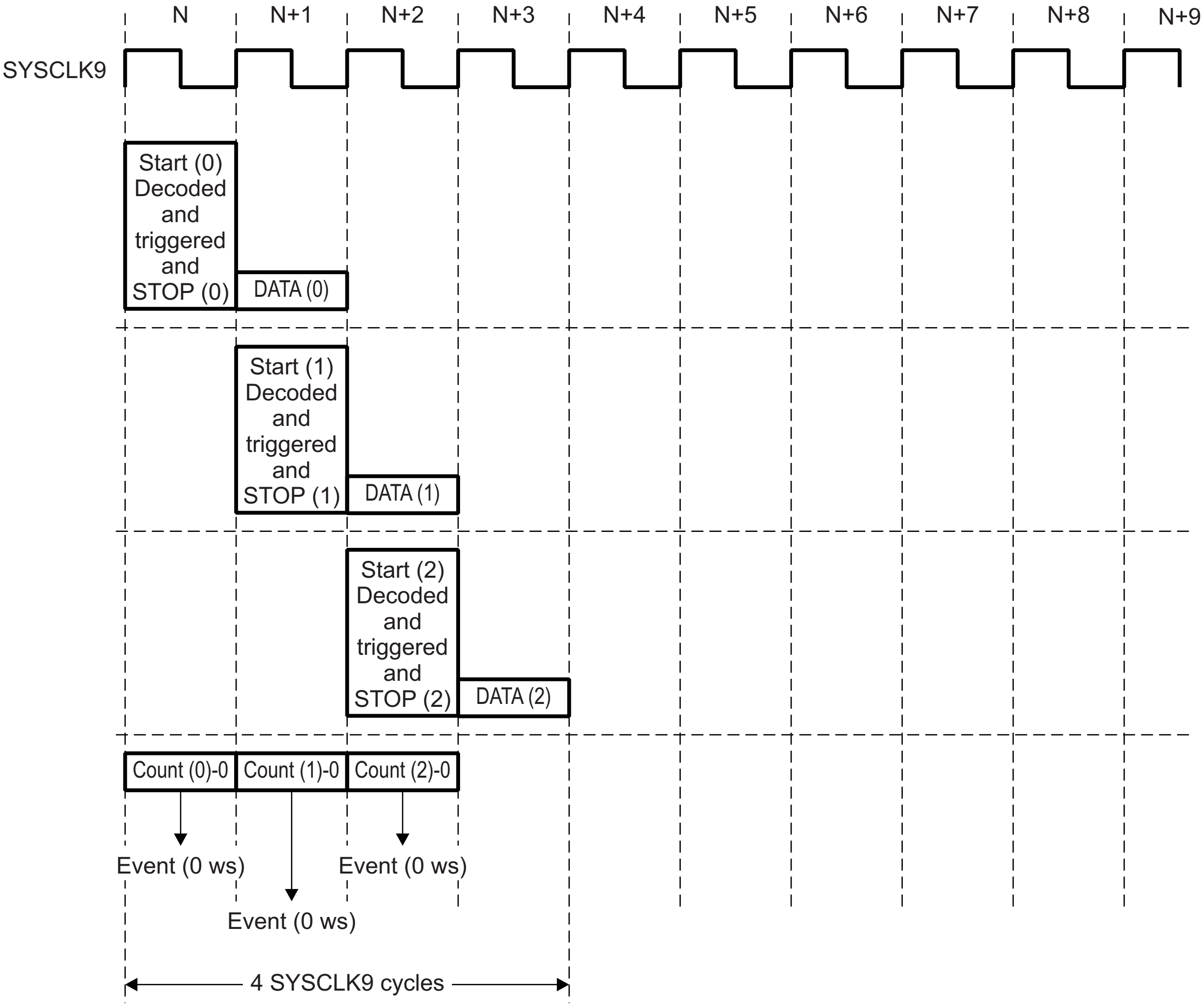
- C64x+ megamodule profile:
  (ignoring PMC/DMC + UMC latencies):
  - C64x+ megamodule sees 6 SYSCLK9 cycle delays to fetch 3 256-bit fetch packets from SMC

- “N” wait-states represents “N+1” SYSCLK9 cycles

- SMC profile:
  - Delay = 3’1+1’1+2’1 = 6 SYSCLK9 cycles

Figure 20. Profiling Three Back-to-Back Requests From CPU_0 to SMC (All Are Prefetch Hits)

- Cycle N: Request (0) comes to SMC from CPU_0 (Prefetch hit)
- Cycle N+1: Request (1) comes to SMC ROM CPU_0 (Prefetch-hit)
- Cycle N+2: Request (2) comes to SMC from CPU_0 (Prefetch-hit)



- Events:
  - 0 ws event (@ N)
  - 0 ws event (@ N+1)
  - 0 ws event (@ N+2)

- C64x+ megamodule profile:
  (ignoring PMC/DMC + UMC latencies):
  - C64x+ megamodule sees 3 SYSCLK9 cycle delays to fetch 3 256-bit fetch packets from SMC

- “N” wait-states represents “N+1” SYSCLK9 cycles

- SMC profile:
  - Delay = 3*1 = 3 SYSCLK9 cycles

7.3 SMC Profiler Register Summary

Table 17 lists the memory mapped registers for the SMC profiler. For the memory address of these registers, see the *TMS320TCI6486 Communications Infrastructure Digital Signal Processor* data manual (SPRS300) or the *TMS320C6472 Fixed-Point Digital Signal Processor* data manual (SPRS612).

Table 17. SMC Profiler Memory Mapped Registers

Offset	Acronym	Register Name	See
0x0	SL2BNKMSK	SMC Bank Mask Register	Section 7.3.1
0x4 - 0x20	SL2PWSCNT0-7	SMC Wait State Counter 0-7 Register	Section 7.3.2
0x24	SL2PFCCNT	SMC Prefetch Counter Register	Section 7.3.3
0x28	SL2PCMD	SMC Profiler Command Register	Section 7.3.4
0x2C	SL2PSTAT	SMC Profiler Counter Saturation Status Register	Section 7.3.5
0x30	SL2STATMASK	SMC Profiler Mask Status Register	Section 7.3.6

7.3.1 SMC Bank Mask Register

In order to give you software flexibility to measure the C64x+ megamodule performance on a particular bank of the SMC, the SMC bank mask register (SL2BNKMSK) inside the SMC BANK mask logic can be used. SBML logic decodes the STATISTICS event and its address from SPCL using the programmed SL2BNKMSK register to decide whether to filter this request or let it go to the SSUL unit. The SL2BNKMSK register is shown in Figure 21 and described in Table 18.

Figure 21. SMC Bank Mask Register (SL2BNKMSK)

31 16
Reserved
R, +0000 0000 0000 0000 0000 0000 0000

15 4	3	2	1	0
Reserved	BANK3	BANK2	BANK1	BANK0
R, +0000 0000 0000 0000 0000 0000 000	R/W, +1	R/W, +1	R/W, +1	R/W, +1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 18. SMC Bank Mask Register (SL2BNKMSK) Field Descriptions

Bit	Field	EnumValue	Description
31-4	Reserved		Reserved
3	BANK3		Statistics events for BANK3
		0	Reject
		1	Accept
2	BANK2		Statistics events for BANK2
		0	Reject
		1	Accept
1	BANK1		Statistics events for BANK1
		0	Reject
		1	Accept
0	BANK0		Statistics events for BANK0
		0	Reject
		1	Accept

7.3.2 SMC Wait-State Counter Register

The SSUL unit records the STATISTICS events from SBML into the corresponding SMC wait state counter register (SL2PWSCNT0-7). The SL2PWSCNT0-7 register is shown in Figure 22 and described in Table 19.

Figure 22. SMC Wait-State Counter 0-7 Register (SL2PWSCNT0-7)

31 … 0
WSCNT
R, +0000 0000 0000 0000 0000 0000 0000 0000

LEGEND: R = Read, W = Write, n = value at reset

Table 19. SMC Wait-State Counter 0-7 Register (SL2PWSCNT0-7) Field Description

Bit	Field	EnumValue	Description
31-0	WSCNT		32-bit counters for n wait state read accesses

7.3.3 SMC Prefetch Counter Register

The SSUL takes an event directly from the SMC called smc_to_smc_trace_prefetch_req and records it in the SMC prefetch counter register (SL2PFCCNT). The SL2PFCCNT register is shown in Figure 23 and described in Table 20.

Figure 23. SMC Prefetch Counter Register (SL2PFCCNT)

31 … 16
PFCCNT
R, +0000 0000 0000 0000 0000 0000 0000 0000

15 … 0
PFCCNT
R, +0000 0000 0000 0000 0000 0000 0000 0000

LEGEND: R = Read, W = Write, n = value at reset

Table 20. SMC Prefetch Counter Register (SL2PFCCNT) Field Description

Bit	Field	EnumValue	Description
31-0	PFCCNT		32-bit prefetch counter

7.3.4 SMC Profiler Command Register (SL2PCMD)

The SMC profiler command register (SL2PCMD) clears counters and enables profiing. Writing 1 to CLEAR (bit 0) clears/resets all the SL2PWSCNT registers and the SL2PFCCNT register. Writing 1 to ENPROFILE (bit 1) enables the profiler to gather statistics. By default, the SMC profiler is disabled. The SL2PCMD register is shown in Figure 24 and described in Table 21.

Figure 24. SMC Profiler Command Register (SL2PCMD)

31–16
Reserved
R, +00 0000 0000 0000 0000 0000 0000 0000

15–2	1	0
Reserved	ENPROFILE	CLEAR
R, +00 0000 0000 0000 0000 0000 0000 0000	R/W, +0	R/W, +0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 21. SMC Profiler Command Register (SL2PCMD) Field Descriptions

Bit	Field	EnumValue	Description
31-2	Reserved		Reserved
1	ENPROFILE		Profile status
		0	Disable
		1	Enable
0	CLEAR		Status of counters
		0	No effect
		1	Clears all the counters

7.3.5 SMC Profiler Counter Saturation Status Register

The SMC profiler counter saturation status register (SL2PSTAT) is read only. It provides the saturation status of all the counters. It provides the status of the SL2PWSCNT and SL2PFCCNT registers. The SL2PSTAT register is shown in Figure 25 and described in Table 22.

Figure 25. SMC Profiler Counter Saturation Status Register (SL2PSTAT)

31–16
Reserved
R, +000 0000 0000 0000 0000 0000

15–9	8	7	6	5	4	3	2	1	0
Reserved	PSAT	SAT7	SAT6	SAT5	SAT4	SAT3	SAT2	SAT1	SAT0
R, +000 0000 0000 0000 0000 0000	R, +0	R, +0	R, +0	R, +0	R, +0	R, +0	R, +0	R, +0	R, +0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 22. SMC Profiler Counter Saturation Status Register (SL2PSTAT) Field Descriptions

Bit	Field	EnumValue	Description
31-9	Reserved		Reserved
8	PSAT		SL2PFCCNT counter saturation
		0	Counter is not saturated
		1	Saturates the SL2PFCCNT counter
7	SAT7		SL2PWSCNT7 counter saturation
		0	Counter is not saturated
		1	Saturates the SL2PWSCNT7 counter
6	SAT6		SL2PWSCNT6 counter saturation
		0	Counter is not saturated
		1	Saturates the SL2PWSCNT6 counter
5	SAT5		SL2PWSCNT5 counter saturation
		0	Counter is not saturated
		1	Saturates the SL2PWSCNT5 counter
4	SAT4		SL2PWSCNT4 counter saturation
		0	Counter is not saturated
		1	Saturates the SL2PWSCNT4 counter
3	SAT3		SL2PWSCNT3 counter saturation
		0	Counter is not saturated
		1	Saturates the SL2PWSCNT3 counter
2	SAT2		SL2PWSCNT2 counter saturation
		0	Counter is not saturated
		1	Saturates the SL2PWSCNT2 counter
1	SAT1		SL2PWSCNT1 counter saturation
		0	Counter is not saturated
		1	Saturates the SL2PWSCNT1 counter
0	SAT0		SL2PWSCNT0 counter saturation
		0	Counter is not saturated
		1	Saturates the SL2PWSCNT0 counter

7.3.6 SMC Profiler Mask Status Register

The SMC profiler combines all eight SMC STATISTICS events and sends a single event (SMC profiler event) to the C64x+ megamodule. The C64x+ megamodule can use this event to trigger various debugging tasks using AET. The SMC profiler mask status memory-mapped register (SL2STATMASK) for masking the events for the event combiner is shown in Figure 26 and described in Table 23.

Figure 26. SMC Profiler Mask Status Register (SL2STATMASK)

31 16
Reserved
R, +0s

15 8	7	6	5	4	3	2	1	0
Reserved	MASK_ WS7	MASK_ WS6	MASK_ WS5	MASK_ WS4	MASK_ WS3	MASK_ WS2	MASK_ WS1	MASK_ WS0
R, +0s	R/W, +0	R/W, +0	R/W, +0	R/W, +0	R/W, +0	R/W, +0	R/W, +0	R/W, +0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 23. SMC Profiler Mask Status Register (SL2STATMASK) Field Descriptions

Bit	Field	EnumValue	Description
31-8	Reserved		Reserved
7	MASK_WS7		Use WS7_EVENT to generate SMC profile event to C64x+ megamodule
		0	Do not combine
		1	Combine
6	MASK_WS6		Use WS6_EVENT to generate SMC profile event to C64x+ megamodule
		0	Do not combine
		1	Combine
5	MASK_WS5		Use WS5_EVENT to generate SMC profile event to C64x+ megamodule
		0	Do not combine
		1	Combine
4	MASK_WS4		Use WS4_EVENT to generate SMC profile event to C64x+ megamodule
		0	Do not combine
		1	Combine
3	MASK_WS3		Use WS3_EVENT to generate SMC profile event to C64x+ megamodule
		0	Do not combine
		1	Combine
2	MASK_WS2		Use WS2_EVENT to generate SMC profile event to C64x+ megamodule
		0	Do not combine
		1	Combine
1	MASK_WS1		Use WS1_EVENT to generate SMC profile event to C64x+ megamodule
		0	Do not combine
		1	Combine
0	MASK_WS0		Use WS0_EVENT to generate SMC profile event to C64x+ megamodule
		0	Do not combine
		1	Combine

Appendix A Revision History

This revision history highlights the technical changes made to the document in this revision.

Table 24. C6472/TCI6486 SMC Revision History

See	Additions/Modifications/Deletions
Section 3.1.3	Modified paragraph before example and Example 1, Shared Counter

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 TI 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. Information of third parties may be subject to additional restrictions.

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.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

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
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps

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