TEXAS INSTRUMENTS

TMS320C62x/C67x

Technical Brief

1998

Digital Signal Processing Solutions

Texas Instruments

Printed in U.S.A., April 1998

SPRU197B

TMS320C62x/C67x Technical Brief

Literature Number: SPRU197B
April 1998

TEXAS INSTRUMENTS

Printed on Recycled Paper

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Preface

Read This First

About This Manual

This book is an introduction to the TMS320C6x generation digital signal processor (DSP) devices. This book describes the CPU architecture, peripherals, and development tools for the TMS320C6x DSPs. Unless otherwise specified, all references to the ’C6x refer to the TMS320C6x generation of DSPs, ’C62x refers to the TMS320C62x fixed-point DSPs in the ’C6x generation, and ’C67x refers to the TMS320C67x floating-point DSPs in the ’C6x generation.

How to Use This Manual

The following table summarizes the information in this technical brief:

If you are looking for information about:	Turn to these chapters:
Code generation tools	Chapter 5, *Development Support*
CPU architecture	Chapter 2, *CPU Architecture*
Development support tools	Chapter 5, *Development Support*
Direct-memory access (DMA)	Chapter 4, *Peripherals*
Evaluation tools	Chapter 5, *Development Support*
External memory interface	Chapter 3, *Memory* Chapter 4, *Peripherals*
Host-port interface	Chapter 4, *Peripherals*
Memory map	Chapter 3, *Memory*
Multichannel buffered serial port (McBSP)	Chapter 4, *Peripherals*
Peripherals	Chapter 4, *Peripherals*
Timers	Chapter 4, *Peripherals*

Related Documentation From Texas Instruments

The following books describe the TMS320C62x/C67x devices and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

TMS320C6201 Digital Signal Processor Data Sheet (literature number SPRS051) describes the features of the TMS320C6201 and provides pinouts, electrical specifications, and timings for the device.

TMS320C6701 Digital Signal Processor Data Sheet (literature number SPRS067) describes the features of the TMS320C6701 and provides pinouts, electrical specifications, and timings for the device.

TMS320C62x/C67x CPU and Instruction Set Reference Guide (literature number SPRU189) describes the 'C62x/C67x CPU architecture, instruction set, pipeline, and interrupts for these digital signal processors.

TMS320C6201/C6701 Peripherals Reference Guide (literature number SPRU190) describes common peripherals available on the TMS320C6201/C6701 digital signal processors. This book includes information on the internal data and program memories, the external memory interface (EMIF), the host port, multichannel buffered serial ports, direct memory access (DMA), clocking and phase-locked loop (PLL), and the power-down modes.

TMS320C62x/C67x Programmer's Guide (literature number SPRU198) describes ways to optimize C and assembly code for the TMS320C62x/C67x DSPs and includes application program examples.

TMS320C6x Assembly Language Tools User's Guide (literature number SPRU186) 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 the 'C6x generation of devices.

TMS320C6x Optimizing C Compiler User's Guide (literature number SPRU187) describes the 'C6x C compiler and the assembly optimizer. This C compiler accepts ANSI standard C source code and produces assembly language source code for the 'C6x generation of devices. The assembly optimizer helps you optimize your assembly code.

TMS320C6x C Source Debugger User's Guide (literature number SPRU188) tells you how to invoke the 'C6x simulator and emulator versions of the C source debugger interface. This book discusses various aspects of the debugger, including command entry, code execution, data management, breakpoints, profiling, and analysis.

TMS320 DSP Development Support Reference Guide (literature number SPRU011) describes the TMS320 family of digital signal processors and the tools that support these devices. Included are code-generation tools (compilers, assemblers, linkers, etc.) and system integration and debug tools (simulators, emulators, evaluation modules, etc.). Also covered are available documentation, seminars, the university program, and factory repair and exchange.

Trademarks

If You Need Assistance . . .

❑ World-Wide Web Sites

TI Online	http://www.ti.com
Semiconductor Product Information Center (PIC)	http://www.ti.com/sc/docs/pic/home.htm
DSP Solutions	http://www.ti.com/dsps
320 Hotline On-line™	http://www.ti.com/sc/docs/dsps/support.htm

❑ North America, South America, Central America

Product Information Center (PIC)	(972) 644-5580		
TI Literature Response Center U.S.A.	(800) 477-8924		
Software Registration/Upgrades	(214) 638-0333	Fax: (214) 638-7742	
U.S.A. Factory Repair/Hardware Upgrades	(281) 274-2285		
U.S. Technical Training Organization	(972) 644-5580		
DSP Hotline	(281) 274-2320	Fax: (281) 274-2324	Email: dsph@ti.com
DSP Modem BBS	(281) 274-2323		

DSP Internet BBS via anonymous ftp to ftp://ftp.ti.com/pub/tms320bbs

❑ Europe, Middle East, Africa

European Product Information Center (EPIC) Hotlines:

Multi-Language Support	+33 1 30 70 11 69	Fax: +33 1 30 70 10 32

Email: epic@ti.com

Deutsch	+49 8161 80 33 11	or +33 1 30 70 11 68
English		+33 1 30 70 11 65
Francais		+33 1 30 70 11 64
Italiano		+33 1 30 70 11 67

EPIC Modem BBS	+33 1 30 70 11 99	
European Factory Repair	+33 4 93 22 25 40	
Europe Customer Training Helpline		Fax: +49 81 61 80 40 10

❑ Asia-Pacific

Literature Response Center	+852 2 956 7288	Fax: +852 2 956 2200
Hong Kong DSP Hotline	+852 2 956 7268	Fax: +852 2 956 1002
Korea DSP Hotline	+82 2 551 2804	Fax: +82 2 551 2828
Korea DSP Modem BBS	+82 2 551 2914	
Singapore DSP Hotline		Fax: +65 390 7179
Taiwan DSP Hotline	+886 2 377 1450	Fax: +886 2 377 2718
Taiwan DSP Modem BBS	+886 2 376 2592	

Taiwan DSP Internet BBS via anonymous ftp to ftp://dsp.ee.tit.edu.tw/pub/TI/

❑ Japan

Product Information Center	+0120-81-0026 (in Japan)	Fax: +0120-81-0036 (in Japan)
	+03-3457-0972 or (INTL) 813-3457-0972	Fax: +03-3457-1259 or (INTL) 813-3457-1259
DSP Hotline	+03-3769-8735 or (INTL) 813-3769-8735	Fax: +03-3457-7071 or (INTL) 813-3457-7071
DSP BBS via Nifty-Serve	Type “Go TIASP”	

❑ Documentation

When making suggestions or reporting errors in documentation, please include the following information that is on the title page: the full title of the book, the publication date, and the literature number.

Mail: Texas Instruments Incorporated
Technical Documentation Services, MS 702
P.O. Box 1443
Houston, Texas 77251-1443

Email: dsph@ti.com

Note: When calling a Literature Response Center to order documentation, please specify the literature number of the book.

Contents

Figures

Tables

Chapter 1

Introduction

The TMS320C6x generation of digital signal processors (DSPs) is part of the TMS320 family of DSPs. The TMS320C62x (’C62x) devices are fixed-point DSPs in the TMS320C6x generation. The TMS320C67x (’C67x) devices are floating-point DSPs in the TMS320C6x generation.

The TMS320C62x and TMS320C67x are code compatible and both feature the VelociTI™ architecture. The VelociTI architecture is a high-performance, advanced, very-long-instruction-word (VLIW) architecture developed by Texas Instruments, making these DSPs excellent choices for multichannel and multifunction applications. VelociTI, together with the development tool set and evaluation tools, provides faster development time and higher performance for embedded DSP applications through increased instruction-level parallelism.

Topic **Page**

1.1 The TMS320 Family of Digital Signal Processors

The TMS320 family consists of 16-bit and 32-bit fixed- and floating-point devices. These DSPs possess the operational flexibility of high-speed controllers and the numerical capability of array processors. The following characteristics make this family the ideal choice for a wide range of processing applications:

- Very flexible instruction set
- Inherent operational flexibility
- High-speed performance
- Innovative, parallel architectural design
- Cost-effectiveness

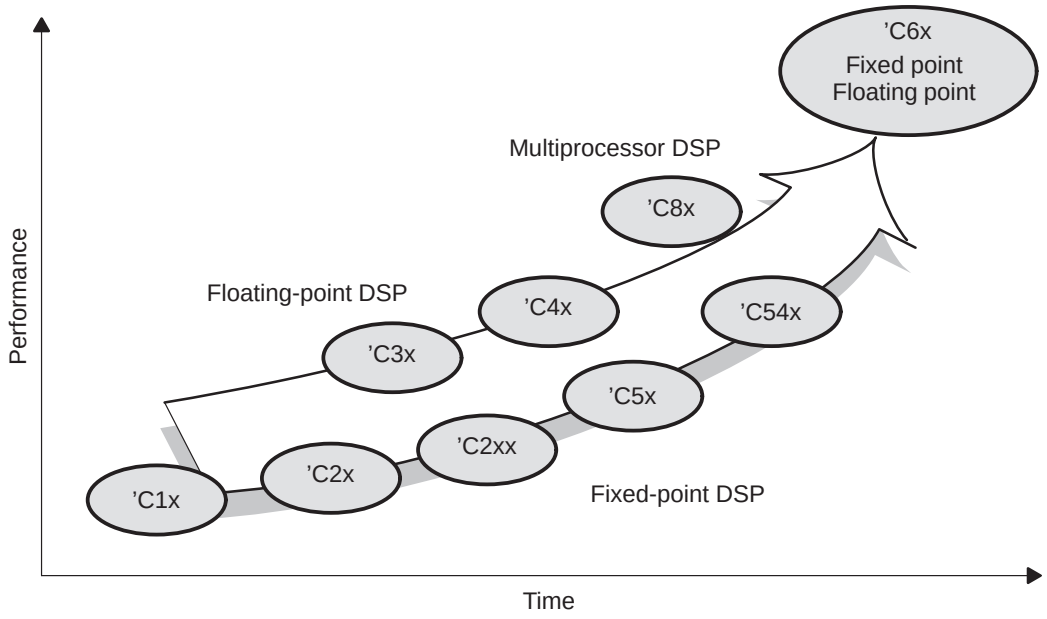
1.1.1 History, Development, and Advantages of TMS320 DSPs

In 1982, Texas Instruments introduced the TMS32010 — the first fixed-point DSP in the TMS320 family. Before the end of the year, the *Electronic Products* magazine awarded the TMS32010 the title “Product of the Year”. The TMS32010 became the model for future TMS320 generations.

Today, the TMS320 family consists of nine generations: the ’C1x, ’C2x, ’C2xx, ’C5x, and ’C54x are fixed-point, the ’C3x and ’C4x are floating-point, and the ’C8x is a multiprocessor. Now there is a new generation of DSPs, the TMS320C6x generation, with performance and features that are reflective of Texas Instruments commitment to lead the world in DSP solutions.

Each generation of TMS320 devices uses a core central processing unit (CPU) that is combined with a variety of on-chip memory and peripheral configurations. These various configurations satisfy a wide range of needs in the worldwide electronics market. When memory and peripherals are integrated with a CPU into one chip, the overall system cost is greatly reduced, and circuit board space is reduced. Figure 1–1 shows the progression of the TMS320 family of devices.

Figure 1–1. The TMS320 Family of Digital Signal Processors (DSPs)



1.1.2 Typical Applications for the TMS320 Family

The TMS320 family of DSPs offers adaptable approaches to traditional signal-processing problems, such as vocoding, filtering, and error coding. Furthermore, the TMS320 family supports complex applications that often require multiple operations to be performed simultaneously. Table 1–1 lists many of the typical applications of the TMS320 family.

Table 1–1. Typical Applications for the TMS320 Family of Digital Signal Processors (DSPs)

Automotive	Consumer	Control
Adaptive ride control	Digital radios/TVs	Disk drive control
Antiskid brakes	Educational toys	Engine control
Cellular telephones	Music synthesizers	Laser printer control
Digital radios	Pagers	Motor control
Engine control	Power tools	Robotics control
Global positioning	Radar detectors	Servo control
Navigation	Solid-state answering machines	
Vibration analysis		
Voice commands		
General-Purpose	**Graphics/Imaging**	**Industrial**
Adaptive filtering	3-D computing	Numeric control
Convolution	Animation/digital maps	Power-line monitoring
Correlation	Homomorphic processing	Robotics
Digital filtering	Image compression/transmission	Security access
Fast Fourier transforms	Image enhancement	
Hilbert transforms	Pattern recognition	
Waveform generation	Robot vision	
Windowing	Workstations	
Instrumentation	**Medical**	**Military**
Digital filtering	Diagnostic equipment	Image processing
Function generation	Fetal monitoring	Missile guidance
Pattern matching	Hearing aids	Navigation
Phase-locked loops	Patient monitoring	Radar processing
Seismic processing	Prosthetics	Radio frequency modems
Spectrum analysis	Ultrasound equipment	Secure communications
Transient analysis		Sonar processing
Telecommunications		**Voice/Speech**
1200- to 56 600-bps modems	Faxing	Speaker verification
Adaptive equalizers	Future terminals	Speech enhancement
ADPCM transcoders	Line repeaters	Speech recognition
Base stations	Personal communications systems (PCS)	Speech synthesis
Cellular telephones	Personal digital assistants (PDA)	Speech vocoding
Channel multiplexing	Speaker phones	Text-to-speech
Data encryption	Spread spectrum communications	Voice mail
Digital PBXs	Digital subscriber loop (xDSL)	
Digital speech interpolation (DSI)	Video conferencing	
DTMF encoding/decoding	X.25 packet switching	
Echo cancellation		

1.2 Introduction to the TMS320C6x Generation of Digital Signal Processors

With performance of up to 1600 million instructions per second (MIPS) and a complete set of development tools, the TMS320C6x DSPs offer cost-effective solutions to high-performance DSP programming challenges. The TMS320C6x development tools include a new C compiler, an assembly optimizer that simplifies programming and scheduling, and a Windows™ debugger interface.

The TMS320C6x DSPs give system architects unlimited possibilities to differentiate their products. High performance, ease of use, and affordable pricing make the TMS320C6x generation the ideal solution for multichannel, multifunction applications, such as:

- Pooled modems
- Wireless local loop base stations
- Beam-forming base stations
- Remote access servers (RAS)
- Digital subscriber loop (DSL) systems
- Cable modems
- Multichannel telephony systems
- Virtual reality 3-D graphics
- Speech recognition
- Audio
- Radar
- Atmospheric modeling
- Finite element analysis
- Imaging (example: fingerprint recognition, ultrasound, and MRI)

The TMS320C6x generation is also an ideal solution for exciting new applications; for example:

- Personalized home security with face and hand/fingerprint recognition
- Advanced cruise control with global positioning systems (GPS) navigation and accident avoidance
- Remote medical diagnostics

1.3 Features and Options of the TMS320C62x/C67x Devices

The ’C62x devices operate at 200 MHz (5-ns cycle time). The ’C67x devices operate at 167 MHz (6-ns cycle time). Both DSPs execute up to eight 32-bit instructions every cycle. The device’s core CPU consists of 32 general-purpose registers of 32-bit word length and eight functional units:

- Two multipliers
- Six ALUs

The ’C62x/C67x have a complete set of optimized development tools, including an efficient C compiler, an assembly optimizer for simplified assembly-language programming and scheduling, and a Windows-based debugger interface for visibility into source code execution characteristics. A hardware emulation board, compatible with the TI XDS510™ emulator interface, is also available. This tool complies with IEEE Standard 1149.1–1990, IEEE Standard Test Access Port and Boundary-Scan Architecture.

Features of the ’C62x/C67x include:

- Advanced VLIW CPU with eight functional units, including two multipliers and six arithmetic units
  - Executes up to eight instructions per cycle for up to ten times the performance of typical DSPs
  - Allows designers to develop highly effective RISC-like code for fast development time
- Instruction packing
  - Gives code size equivalence for eight instructions executed serially or in parallel
  - Reduces code size, program fetches, and power consumption.
- All instructions execute conditionally.
  - Reduces costly branching
  - Increases parallelism for higher sustained performance
- Code executes as programmed on independent functional units.
  - Industry’s most efficient C compiler on DSP benchmark suite
  - Industry’s first assembly optimizer for fast development and improved parallelization
- 8/16/32-bit data support, providing efficient memory support for a variety of applications
- 40-bit arithmetic options add extra precision for vocoders and other computationally intensive applications

- Saturation and normalization provide support for key arithmetic operations.
- Field manipulation and instruction extract, set, clear, and bit counting support common operation found in control and data manipulation applications.

The ’C67x has the following features:

- Peak 1336 MIPS at 167 MHz
- Peak 1G FLOPS at 167 MHz for single-precision operations
- Peak 250M FLOPS at 167 MHz for double-precision operations
- Peak 688M FLOPS at 167 MHz for multiply and accumulate operations
- Hardware support for single precision (32-bit) and double precision (64-bit) IEEE floating-point operations.
- 32 $\times$ 32-bit integer multiply with 32- or 64-bit result.

A variety of memory and peripheral options are available for the ’C62x/C67x:

- Large on-chip RAM for fast algorithm execution
- 32-bit external memory interface supports SDRAM, SBSRAM, SRAM, and other asynchronous memories, for a broad range of external memory requirements and maximum system performance
- 16-bit host port fot access to ’C62x/C67x memory and peripherals
- Multichannel DMA controller
- Multichannel serial port(s)
- 32-bit timer(s)

Chapter 2

CPU Architecture

The VelociTI architecture makes the ’C6x the first off-the-shelf DSP to use an enhancement of traditional VLIW to achieve high performance through increased instruction-level parallelism. A traditional VLIW architecture consists of multiple execution units running in parallel that perform multiple instructions during a single clock cycle. Parallelism is the key to extremely high performance and takes these next-generation DSPs well beyond the performance capabilities of traditional superscalar designs. VelociTI is a highly deterministic architecture, with few restrictions on how or when instructions are fetched, executed, or stored. This architectural flexibility is key to the breakthrough efficiency levels of the ’C6x compiler.

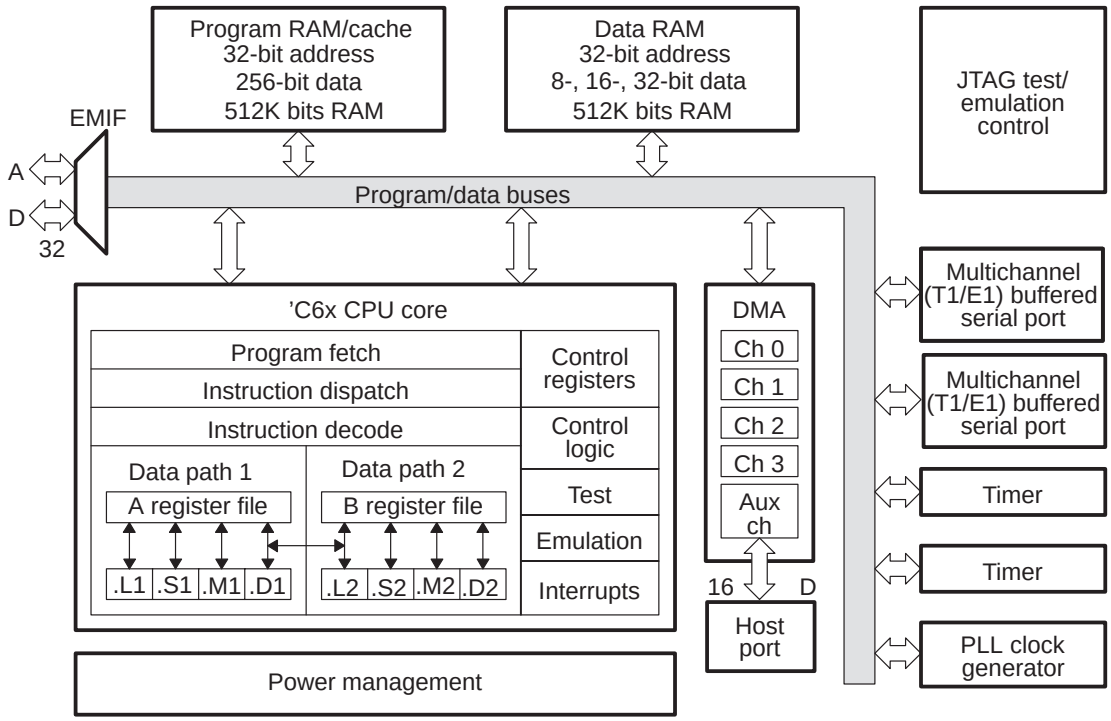
Topic **Page**

2.1 TMS320C62x/C67x Block Diagram

The ’C62x/C67x processor consists of three main parts: CPU (or the *core*), peripherals, and memory. Eight functional units operate in parallel, with two similar sets of the basic four functional units. The units communicate using a cross path between two register files, each of which contains 16 32-bit registers. Program parallelism is defined at compile time because there is no data dependency checking done in hardware during run time. The 256-bit-wide program memory fetches eight 32-bit instructions every single cycle.

Figure 2–1 is the block diagram for the TMS320C62x/C67x devices. The ’C62x/C67x devices come with program memory, which, on some devices, can be used as a program cache. The devices also have varying sizes of data memory. Peripherals such as a a direct memory access (DMA) controller, power-down logic, and external memory interface (EMIF) usually come with the CPU, while peripherals such as serial ports, host ports, and timers are only available on certain devices. Check the data sheet for your device to determine the specific peripheral configurations you have.

Figure 2–1. TMS320C62x/C67x Block Diagram



2.2 Central Processing Unit (CPU)

The ’C62x/C67x CPU, in Figure 2–1, is common to all the ’C62x/C67x devices. The CPU contains:

- Program fetch unit
- Instruction dispatch unit
- Instruction decode unit
- 32 32-bit registers
- Two data paths, each with four functional units
- Control registers
- Control logic
- Test, emulation, and interrupt logic

The CPU has two data paths (A and B) in which processing occurs. Each data path has four functional units (.L, .S, .M, and .D) and a register file containing 16 32-bit registers. The functional units execute logic, shifting, multiply, and data address operations. All instructions except loads and stores operate on the registers. The two data-addressing units (.D1 and .D2) are exclusively responsible for all data transfers between the register files and memory.

The four functional units of a data path have a single data bus connected to registers on the other side of the CPU so that the units can exchange data with the register file on the opposite side. Register access across the CPU supports one read and write operation per cycle.

The two sets of functional units include the following items:

- Two multipliers
- Six arithmetic logic units (ALUs)
- Two register files, each containing 16 32-bit registers

Each functional unit is controlled by a 32-bit instruction. The instruction fetch, instruction dispatch, and instruction decode blocks can deliver up to eight 32-bit instructions from the program memory to the functional units every cycle. The control register file provides methods to configure and control various aspects of processor operation. Access to the control registers is provided from data-path B.

The VLIW processing flow begins when a 256-bit-wide instruction fetch packet (IFP) is fetched from the internal program memory. The instructions linked together for simultaneous execution (up to eight instructions) form an execute packet. For more details on the processing, see the data sheet for your particular device.

2.3 CPU Data Paths

Figure 2–2 shows the ’C62x CPU data paths and Figure 2–3 shows the ’C67x CPU data paths, which consist of:

- Two general-purpose register files (A and B)
- Eight functional units (.L1, .L2, .S1, .S2, .M1, .M2, .D1, and .D2)
- Two load-from-memory paths (LD1 and LD2)
- Four load-from-memory paths(LD1, Ld
- Two store-to-memory paths (ST1 and ST2)
- Two register file cross paths (1X and 2X)
- Two data address paths (DA1 and DA2)

2.3.1 General-Purpose Register Files

There are two general-purpose register files (A and B) in the ’C62x/C67x data paths. Each of these files contains 16 32-bit registers (A0–A15 for file A and B0–B15 for file B). The general-purpose registers can be used for data or data-address pointers. Registers A1, A2, B0, B1, and B2 can be used for condition registers. Registers A4–A7 and B4–B7 can be used for circular addressing.

The general-purpose register files support 32- and 40-bit fixed-point data. 32-bit data can be contained in any general-purpose register. 40-bit data is contained across two registers; the 32 LSBs of the data are placed in an even register and the remaining eight MSBs are placed in the eight LSBs of the next upper register (which is always an odd register). The ’C67x also uses these register pairs to hold 64-bit double- precision floating-point values.

Figure 2–2. TMS320C62x CPU Data Paths

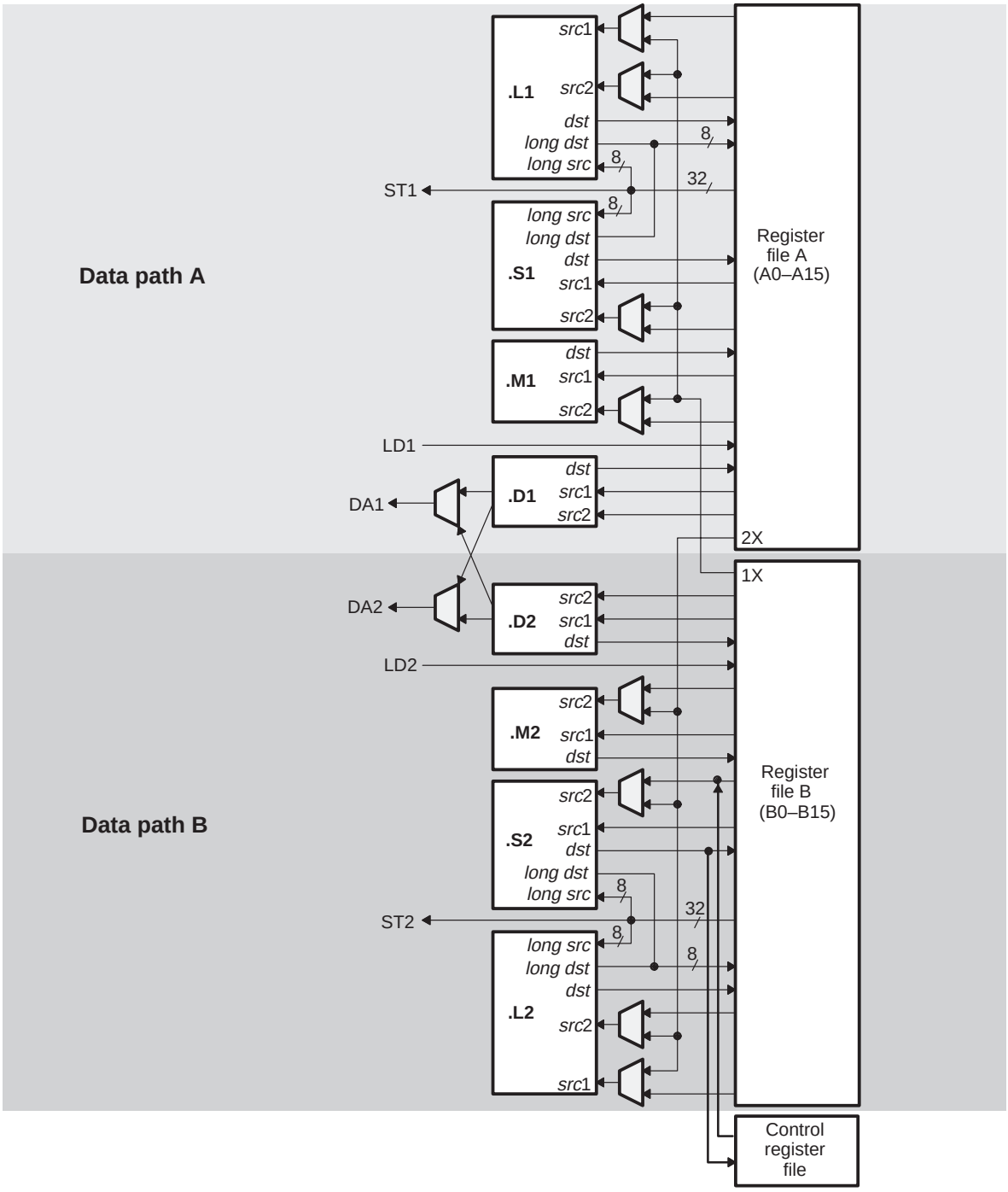
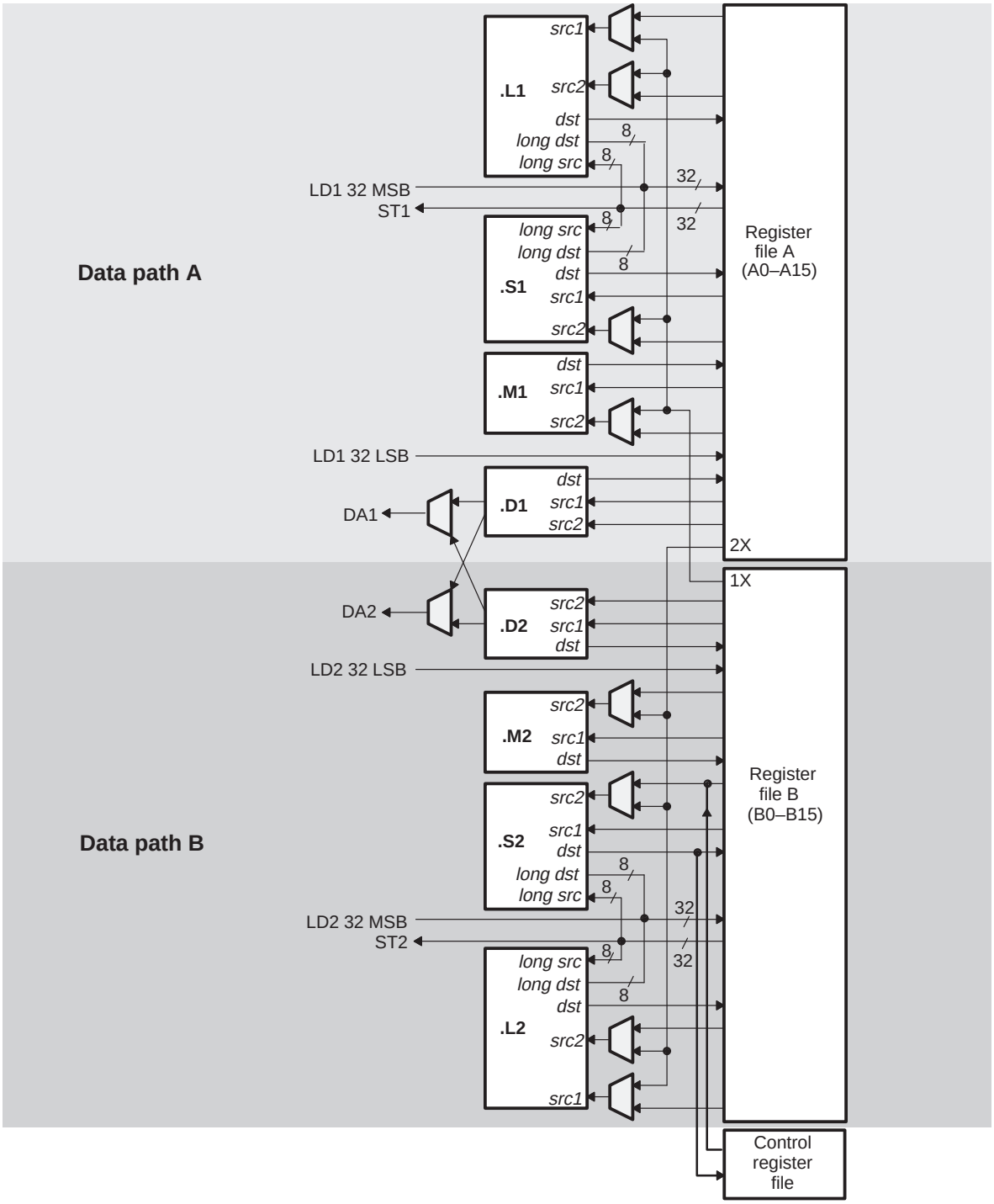


Figure 2–3. TMS320C67x CPU and Data Paths



2.3.2 Functional Units

The eight functional units in the ’C62x/C67x data paths can be divided into two groups of four; each functional unit in one data path is almost identical to the corresponding unit in the other data path. The functional units are described in Table 2–1.

Most data lines in the CPU support 32-bit operands, and some support long (40-bit) operands. Each functional unit has its own 32-bit write port into a general-purpose register file. All units ending in 1 (for example, .L1) write to register file A and all units ending in 2 write to register file B. Each functional unit has two 32-bit read ports for source operands *src1* and *src2*. Four units (.L1, .L2, .S1, and .S2) have an extra 8-bit-wide port for 40-bit long writes as well as an 8-bit input for 40-bit long reads. Because each unit has its own 32-bit write port, all eight units can be used in parallel every cycle.

Table 2–1. Functional Units and Operations Performed

Functional Unit	Fixed-Point Operations	Floating-Point Operations
.L unit (.L1,.L2)	32/40-bit arithmetic and compare operations Leftmost 1 or 0 bit counting for 32 bits Normalization count for 32 and 40 bits 32-bit logical operations	Arithmetic operations. Conversion operations: DP $\rightarrow$ SP, INT $\rightarrow$ DP, INT $\rightarrow$ SP
.S unit (.S1, .S2)	32-bit arithmetic operations 32/40-bit shifts and 32-bit bit-field operations 32-bit logical operations Branches Constant generation Register transfers to/from the control register file (.S2 only)	Compare reciprocal and reciprocal square-root operations. Absolute value operations. SP to DP conversion operations.
.M unit (.M1, .M2)	16 $\times$ 16 bit multiply operations	32 $\times$ 32 bit multiply operations. Floating-point multiply operations.
.D unit (.D1, .D2)	32-bit add, subtract, linear and circular address calculation Loads and stores with a 5-bit constant offset Loads and stores with 15-bit constant offset (.D2 only)	Load double word with 5-bit constant offset.

2.3.3 TMS320C62x/C67x Control Register Files

One unit (.S2) can read from and write to the control register file, shown in Figure 2–2 and Figure 2–3. Table 2–2 lists the control registers contained in the control register file and describes each. Each control register is accessed by the MVC instruction.

Table 2–2. Control Registers

Register		
Abbreviation	**Name**	**Description**
AMR	Addressing mode register	Specifies whether to use linear or circular addressing for each of eight registers; also contains sizes for circular addressing
CSR	Control status register	Contains the global interrupt enable bit, cache control bits, and other miscellaneous control and status bits
IFR	Interrupt flag register	Displays status of interrupts
ISR	Interrupt set register	Allows you to set pending interrupts manually
ICR	Interrupt clear register	Allows you to clear pending interrupts manually
IER	Interrupt enable register	Allows enabling/disabling of individual interrupts
ISTP	Interrupt service table pointer	Points to the beginning of the interrupt service table
IRP	Interrupt return pointer	Contains the address to be used to return from a maskable interrupt
NRP	Nonmaskable interrupt return pointer	Contains the address to be used to return from a nonmaskable interrupt
PCE1	Program counter, E1 phase	Contains the address of the fetch packet that contains the execute packet in the E1 pipeline stage

2.3.4 TMS320C67x Control Register File Extensions

The ’C67x has three additional configuration registers to support floating point operations. The registers specify the desired floating-point rounding mode for the .L and .M units. They also contain bit fields to warn if *src1* and *src2* are NaN (not a number) or denormal numbers, and if the result overflows, underflows, is inexact, infinite, or invalid. There are also fields to warn if a divide by 0 was performed, or if a compare was attempted with a NaN source.

Table 2–3. TMS320C67x Control Register File Extensions

Register		
Abbreviation	**Name**	**Description**
FADCR	Floating-point adder configuration register	Specifies underflow mode, rounding mode, NaNs and other exceptions for the .L unit.
FAUCR	Floating-point auxiliary configuration register	Specifies underflow mode, rounding mode, NaNs and other exceptions for the .S unit.
FMCR	Floating-point multiplier configuration register	Specifies underflow mode, rounding mode, NaNs and other exceptions for the .M unit.

2.3.5 Register File Cross Paths

Each functional unit reads directly from and writes directly to the register file within its own data path. That is, the .L1, .S1, .D1, and .M1 units write to register file A and the .L2, .S2, .D2, and .M2 units write to register file B. The register files are connected to the opposite-side register file’s functional units via the 1X and 2X cross paths. These cross paths allow functional units from one data path to access a 32-bit operand from the opposite side’s register file. The 1X cross path allows data path A’s functional units to read their source from register file B and the 2X cross path allows data path B’s functional units to read their source from register file A.

Six of the functional units have access to the opposite side’s register file via a cross path. The .M1, .M2, .S1, and .S2 units’ *src2* inputs are multiplex-selectable between the cross path and the same side register file. The .L1 and .L2 units’ *src1* and *src2* inputs are also multiplex-selectable between the cross path and the same side register file.

Only two cross paths, 1X and 2X, exist in the ’C62x/C67x CPUs. This limits one source read from each data path’s opposite register file per cycle, or two cross-path source reads per cycle.

2.3.6 Memory, Load, and Store Paths

There are two 32-bit paths for loading data from memory to the register file: LD1 for register file A, and LD2 for register file B. The 'C67x also has a second 32-bit load path for both register files A and B, which allows the LDDW instruction to simultaneously load two 32-bit registers into side A and two 32-bit registers into side B. There are also two 32-bit paths, ST1 and ST2, for storing register values to memory from each register file. The store paths are shared with the .L and .S long read paths.

2.3.7 Data-Address Paths

The data-address paths (DA1 and DA2) shown in Figure 2–2 and Figure 2–3 coming out of the .D units allow data addresses generated from one register file to support loads and stores to memory from the other register file. However, loads and stores executed in parallel must load to and from the same register file or both use a crosspath to the opposite register.

2.4 Mapping Between Instructions and Functional Units

Table 2–4 shows the mapping between instructions and functional units and Table 2–5 shows the mapping between functional units and instructions for the TMS320C62x/C67x fixed-point instructions.

Table 2–4. Fixed-Point Instruction to Functional Unit Mapping

.L Unit	.M Unit	.S Unit		.D Unit	
ABS	MPY	ADD	SET	ADD	STB (15-bit offset)‡
ADD	MPYU	ADDU	SHL	ADDU	STH (15-bit offset)‡
ADDU	MPYUS	ADDK	SHR	ADDAB	STW (15-bit offset)‡
AND	MPYSU	ADD2	SHRU	ADDAH	SUB
CMPEQ	MPYH	AND	SHRL	ADDAW	SUBAB
CMPGT	MPYHU	B disp	SUB	LDB	SUBAH
CMPGTU	MPYHUS	B IRP†	SUBU	LDBU	SUBAW
CMPLT	MPYHSU	B NRP†	SUB2	LDH	ZERO
CMPLTU	MPYHL	B reg	XOR	LDHU	
LMBD	MPYHLU	CLR	ZERO	LDW	
MV	MPYHULS	EXT		LDB (15-bit offset)‡	
NEG	MPYHSLU	EXTU		LDBU (15-bit offset)‡	
NORM	MPYLH	MV		LDH (15-bit offset)‡	
NOT	MPYLHU	MVC†		LDHU (15-bit offset)‡	
OR	MPYLUHS	MVK		LDW (15-bit offset)‡	
SADD	MPYLSHU	MVKH		MV	
SAT	SMPY	MVKLH		STB	
SSUB	SMPYHL	NEG		STH	
SUB	SMPYLH	NOT		STW	
SUBU	SMPYH	OR			
SUBC					
XOR					
ZERO					

† S2 only
‡ D2 only

Table 2–5. Functional Unit to Fixed-Point Instruction Mapping

	Functional Units			
Instruction	**.L Unit**	**.M Unit**	**.S Unit**	**.D Unit**
ABS	✓			
ADD	✓		✓	✓
ADDU	✓		✓	✓
ADDAB				✓
ADDAH				✓
ADDAW				✓
ADDK			✓	
ADD2			✓	
AND	✓		✓	
B			✓	
B IRP			✓†	
B NRP			✓†	
B reg			✓†	
CLR			✓	
CMPEQ	✓			
CMPGT	✓			
CMPGTU	✓			
CMPLT	✓			
CMPLTU	✓			
EXT			✓	
EXTU			✓	
IDLE				
LDB mem				✓
LDBU mem				✓
LDH mem				✓
LDHU mem				✓

† S2 only
‡ D2 only

Table 2–5. Functional Unit to Fixed-Point Instruction Mapping (Continued)

	Functional Units			
Instruction	**.L Unit**	**.M Unit**	**.S Unit**	**.D Unit**
LDW mem				✓
LDB mem (15-bit offset)				✓‡
LDBU mem (15-bit offset)				✓‡
LDH mem (15-bit offset)				✓‡
LDHU mem (15-bit offset)				✓‡
LDW mem (15-bit offset)				✓‡
LMBD	✓			
MPY		✓		
MPYU		✓		
MPYUS		✓		
MPYSU		✓		
MPYH		✓		
MPYHU		✓		
MPYHUS		✓		
MPYHSU		✓		
MPYHL		✓		
MPYHLU		✓		
MPYHULS		✓		
MPYHSLU		✓		
MPYLH		✓		
MPYLHU		✓		
MPYLUHS		✓		
MPYLSHU		✓		
MV	✓		✓	✓
MVC†			✓	
MVK			✓	

† S2 only
‡ D2 only

Table 2–5. Functional Unit to Fixed-Point Instruction Mapping (Continued)

	Functional Units			
Instruction	**.L Unit**	**.M Unit**	**.S Unit**	**.D Unit**
MVKH			✓	
MVKLH			✓	
NEG	✓		✓	
NOP				
NORM	✓			
NOT	✓		✓	
OR	✓		✓	
SADD	✓			
SAT	✓			
SET			✓	
SHL			✓	
SHR			✓	
SHRU			✓	
SMPY		✓		
SMPYH		✓		
SMPYHL		✓		
SMPYLH		✓		
SSHL			✓	
SSUB	✓			
STB mem				✓
STH mem				✓
STW mem				✓
STB mem (15-bit offset)				✓‡
STH mem (15-bit offset)				✓‡
STW mem (15-bit offset)				✓‡
SUB	✓		✓	✓

† S2 only
‡ D2 only

Table 2–5. Functional Unit to Fixed-Point Instruction Mapping (Continued)

	Functional Units			
Instruction	**.L Unit**	**.M Unit**	**.S Unit**	**.D Unit**
SUBU	✓		✓	
SUBAB				✓
SUBAH				✓
SUBAW				✓
SUBC	✓			
SUB2			✓	
XOR	✓		✓	
ZERO	✓		✓	✓

† S2 only
‡ D2 only

Table 2–6 shows the mapping between instructions and functional units and Table 2–7 shows the mapping between functional units and instructions for the TMS320C67x floating-point instructions.

Table 2–6. Floating-Point Instruction to Functional Unit Mapping

.L Unit	**.M Unit**	**.S Unit**	**.D Unit**
ADDDP	MPYDP	ABSDP	ADDAD
ADDSP	MPYI	ABSSP	LDDW
DPINT	MPYID	CMPEQDP	
DPSP	MPYSP	CMPEQSP	
INTDP		CMPGTDP	
INTDPU		CMPGTSP	
INTSP		CMPLTDP	
INTSPU		CMPLTSP	
SPINT		RCPDP	
SPTRUNC		RCPSP	
SUBDP		RSQRDP	
SUBSP		RSQRSP	
		SPDP	

Table 2–7. Functional Unit to Floating-Point Instruction Mapping

	Functional Units				
Instruction	**.L Unit**	**.M Unit**	**.S Unit**	**.D Unit**	**Type**
ABSDP			✓		2-cycle DP
ABSSP			✓		Single cycle
ADDAD				✓	Single cycle
ADDDP	✓				ADDDP/ SUBDP
ADDSP	✓				Four cycle
CMPEQDP			✓		DP compare
CMPEQSP			✓		Single cycle
CMPGTDP			✓		DP compare
CMPGTSP			✓		Single cycle
CMPLTDP			✓		DP compare
CMPLTSP			✓		Single cycle
DPINT	✓				4-cycle
DPSP	✓				4-cycle
DPTRUNC	✓				4-cycle
INTDP	✓				INTDP
INTDPU	✓				INTDP
INTSP	✓				4-cycle
INTSPU	✓				4-cycle
LDDW				✓	Load
MPYDP		✓			MPYDP
MPYI		✓			MPYI
MPYID		✓			MPYID
MPYSP		✓			4-cycle
RCPDP			✓		2-cycle DP
RCPSP			✓		Single cycle
RSQRDP			✓		2-cycle DP
RSQRSP			✓		Single cycle

Table 2–7. Functional Unit to Floating-Point Instruction Mapping(Continued)

	Functional Units				
Instruction	**.L Unit**	**.M Unit**	**.S Unit**	**.D Unit**	**Type**
SPDP			✓		2-cycle DP
SPINT	✓				4-cycle
SPTRUNC	✓				4-cycle
SUBDP	✓				ADDDP/ SUBDP
SUBSP	✓				4-cycle

2.5 Addressing Modes

The addressing modes on the ’C62x and ’C67x are linear, circular using BK0, and circular using BK1. The mode is specified by the addressing mode register (AMR).

All registers can perform linear addressing. Only eight registers can perform circular addressing: A4–A7 are used by the .D1 unit and B4–B7 are used by the .D2 unit. No other units can perform circular addressing. **LDB/LDH/LDW**, **STB/STH/STW**, **ADDAB/ADDAH/ADDAW**, and **SUBAB/SUBAH/SUBAW** instructions all use the AMR to determine what type of address calculations are performed for these registers.

The ’C62x/C67x CPU has a load/store architecture, which means that the only way to access data in memory is with a load or store instruction. Table 2–8 shows the syntax of an indirect address to a memory location.

Table 2–8. Indirect Address Generation for Load/Store

Addressing Type	No Modification of Address Register	Preincrement or Predecrement of Address Register	Postincrement or Postdecrement of Address Register
Register indirect	*R	*++R *– –R	*R++ *R– –
Register relative	*+R[*ucst5*] *–R[*ucst5*]	*+ +R[*ucst5*] *– –R[*ucst5*]	*R+ +[*ucst5*] *R– –[*ucst5*]
Base + index	*+R[*offsetR*] *–R[*offsetR*]	*+ +R[*offsetR*] *– –R[*offsetR*]	*R+ +[*offsetR*] *R– –[*offsetR*]

For more information on addressing modes, see the *TMS320C62x/C67x CPU and Instruction Set Reference Guide*.

2.6 Interrupts

The ’C62x/C67x CPU has 14 interrupts. These are reset, the nonmaskable interrupt (NMI), and interrupts 4–15. These interrupts correspond to the $\overline{\text{RESET}}$, NMI, and INT4–INT15 signals on the CPU boundary. In some ’C62x/C67x devices, these signals may be tied directly to pins on the device, connected to on-chip peripherals, or may be disabled permanently by being tied inactive on chip. Generally, $\overline{\text{RESET}}$ and NMI are connected directly to pins on the device. Characteristics of interrupt servicing include:

- The IACK pin from the CPU is used to acknowledge an interrupt request.
- The INUM0–INUM3 pins indicate which interrupt vector is being serviced.
- Interrupt vectors are relocatable.
- If an interrupt vector can be serviced in one fetch packet, it can remain in on-chip memory for maximum performance.

For more information on interrupts, see the *TMS320C62x/C67x CPU and Instruction Set Reference Guide*.

Chapter 3

Memory

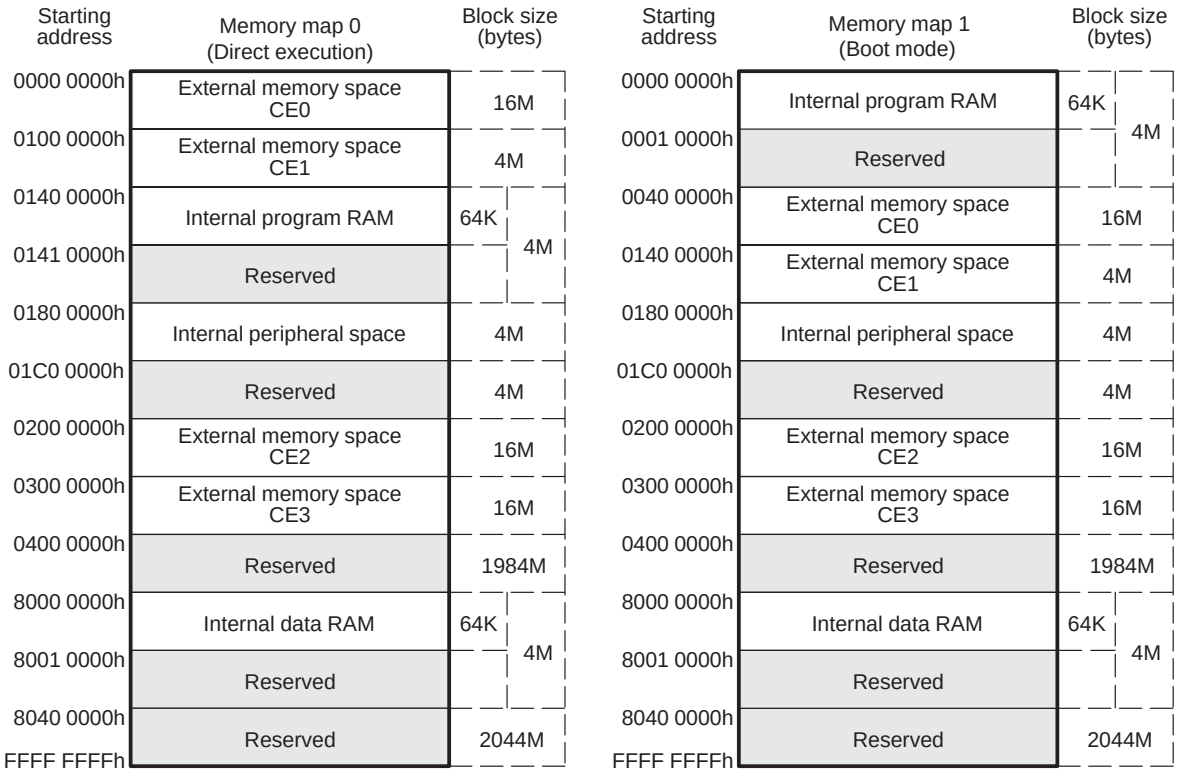
The TMS320C62x and TMS320C67x devices come with on-chip data and program memory that can be selected for use as program memory or program cache. The device is available with varying sizes of data memory, but many systems will be designed with external memory. Also, the external memory interface (EMIF) is used to interface to multiple synchronous and asynchronous memory types in these designs.

Topic **Page**

3.1 Memory Map

Figure 3–1 shows the memory map of the TMS320C62x/C67x DSP. The total memory address range of the ’C62x/C67x is 4G bytes (corresponding to 32-bit internal address representation). The memory map is divided into the internal program memory, internal data memory, three external memory spaces, and internal peripheral space. Five BOOTMODE pins determine which memory map is used. The program starts loading from external address 0 in direct execution, whereas in boot mode, the direct-memory access (DMA) controller loads the internal program memory contents from external PROM before starting execution at internal address 0.

Figure 3–1. TMS320C62x/C67x Memory Map



3.2 Internal Memory

Internal memory consists of 64K bytes of program memory and 64K bytes of data memory. Access to internal program memory is controlled by the program memory controller and access to internal data memory is controlled by the data memory controller. The entire 64K bytes of internal program memory can also be configured as cache.

3.2.1 Data Memory Access

The data memory controller services all requests to internal memory as well as all CPU data requests.

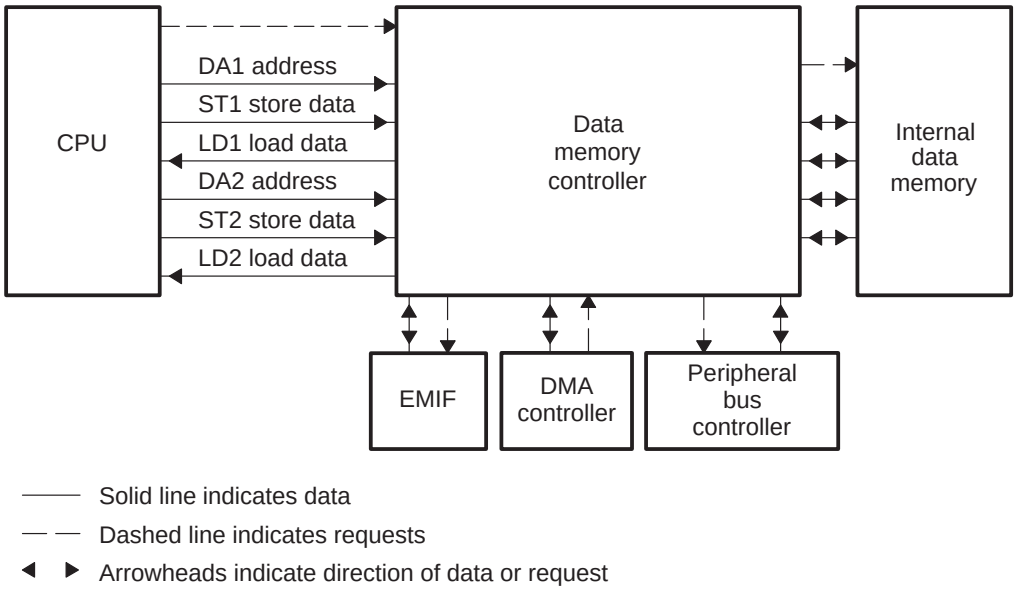
The CPU requests data reads and writes to:

- Internal data memory
- On-chip peripherals through the peripheral bus controller
- The EMIF

The DMA controller requests reads and writes to internal data memory.

The CPU sends requests to the data memory controller through the two address buses (DA1 and DA2). The data to be stored is transmitted through the CPU data store buses (ST1 and ST2). Load data is received through the CPU data load buses (LD1 and LD2). The CPU data requests are mapped based on the requested data's memory address range to either the internal data memory, internal peripheral space (through the peripheral bus controller), or the external memory interface. The data memory controller also connects to the internal data memory and performs CPU/DMA arbitration for the on-chip data RAM. Figure 3–2 shows the CPU, data memory controller, and peripheral bus connections.

Figure 3–2. Data Memory Controller Interconnect to Other Blocks



3.2.2 Internal Data Memory Organization

For revision 2 of the ’C6201, the 64K bytes of internal data RAM are organized in one block. This block is organized as four 8K banks of 16-bit halfwords. Both CPU and DMA can simultaneously access data that resides in different banks.

For revision 3 of the ’C6201, the 64K bytes of internal data RAM are organized in two blocks of 32K bytes. The DMA controller or side A and side B of the CPU can simultaneously access any portion of the internal memory without conflict, when using different blocks. Each block is organized as four 4K banks of 16-bit halfwords.

For the ’C6701, the 64K bytes of internal data RAM are organized in two blocks of 32K bytes each. Side A and side B of the CPU or the DMA controller can simultaneously access any portion of the internal data memory without conflict, when using different blocks. Each block is organized as eight 2K banks of 16-bit halfwords.

Both both sides of the CPU and the DMA can still simultaneously access data that resides in different banks within the same block without performance penalty.

This organization also allows the two CPU data ports, A and B, to simultaneously access neighboring 16-bit data elements inside the same block without a resource conflict.

Both the CPU and the DMA controller can read and write 8-bit bytes, 16-bit halfwords, and 32-bit words. The data memory controller performs arbitration between the CPU and the DMA controller independently for each 16-bit block. The following data alignment restrictions apply:

- Doublewords: (C67x only) Doublewords are aligned on even eight-byte (doubleword) boundaries, and always start at a byte address where the three LSBs are 0. Doublewords are only used on loads triggered by the LDDW instruction. Store operations do not use doublewords.
- Words: Words are aligned on even four-byte (word) boundaries, and always start at a byte address where the two LSBs are 0. A word access requires two adjacent 16-bit-wide banks.
- Halfwords: Halfwords are aligned on even two-byte (halfword) boundaries, and always start at byte addresses where the LSB is 0. Halfword accesses require the entire 16-bit-wide bank.
- Bytes: There are no alignment restrictions related to byte accesses.

Interleaved memory organization allows the CPU to access two addresses in memory simultaneously. In one CPU cycle, two simultaneous accesses to two different internal memory blocks occur without the necessity of delay slots. Two simultaneous accesses to the same internal memory bank stall the entire CPU pipeline for one CPU clock cycle, providing two accesses in two CPU clock cycles.

If a direct memory access to internal memory does not require the same 16-bit banks used by any CPU access, the DMA operation occurs in the same cycle. If the DMA controller has multiple consecutive requests to the block required by the CPU, the CPU is held off until all DMA accesses to the necessary blocks finish. The priority bit (PRI) can be set in the DMA channel primary control register to make DMA a higher priority than CPU memory access.

The CPU and the DMA controller support a programmable endianness. This endianness is selected by the LENDIAN (little endian) pin on the device.

3.2.3 Peripheral Bus

The peripherals are controlled by the CPU and the DMA controller through accesses of control registers. The CPU and DMA controller access these registers through the peripheral data bus. The DMA controller accesses the peripheral bus controller directly, while the CPU accesses the peripheral bus controller through the data memory controller.

The peripheral bus controller converts all peripheral bus accesses to word accesses. However, on read accesses both the CPU and DMA can extract the correct portions of the word to perform byte and halfword accesses properly. Any side effects caused by a peripheral control register read will occur, regardless of which bytes are read.

An isolated peripheral bus controller access from the CPU causes four CPU delay cycles. On consecutive accesses, an access after the first one requires only three CPU cycles due to the pipelined nature of the data memory controller's interface to the peripheral bus controller.

The peripheral bus controller performs arbitration between the CPU and the DMA controller for peripheral access. Like internal data access, the priority (PRI) bits in the DMA controller determine the priority between the CPU and the DMA controller. If a conflict occurs between the CPU and the DMA controller (via the data memory controller), the lower priority requester is held off until the higher priority requester completes all accesses to the peripheral bus controller. The peripheral bus is arbitrated as a single resource, so the lower priority resource is blocked from accessing all peripherals, not just the resource accessed by the higher priority requester.

3.3 External Memory Interface (EMIF)

The external memory interface (EMIF) connects the CPU and external memory, such as synchronous dynamic RAM (SDRAM), synchronous burst static RAM (SBSRAM), and asynchronous memory. The EMIF also provides 8-bit-wide and 16-bit-wide memory read capability to support low-cost boot ROM memories (flash, EEPROM, EPROM, and PROM). The EMIF supports high throughput interfaces to SDRAM, including burst capability.

For more information on memory, see the *TMS320C62x/C67x CPU and Instruction Set Reference Guide*. For more information on the EMIF, see section 4.3 of this book and the *TMS320C6201/C6701 Peripherals Reference Guide*.

Chapter 4

Peripherals

In addition to on-chip memory, the TMS320C62x and TMS320C67x devices also contain peripherals for communication with off-chip memory, coprocessors, host processors, and serial devices. All of these peripherals are briefly described here, but each ’C6x device has only a specific subset of them. See the data sheet for your particular device to determine the peripherals present.

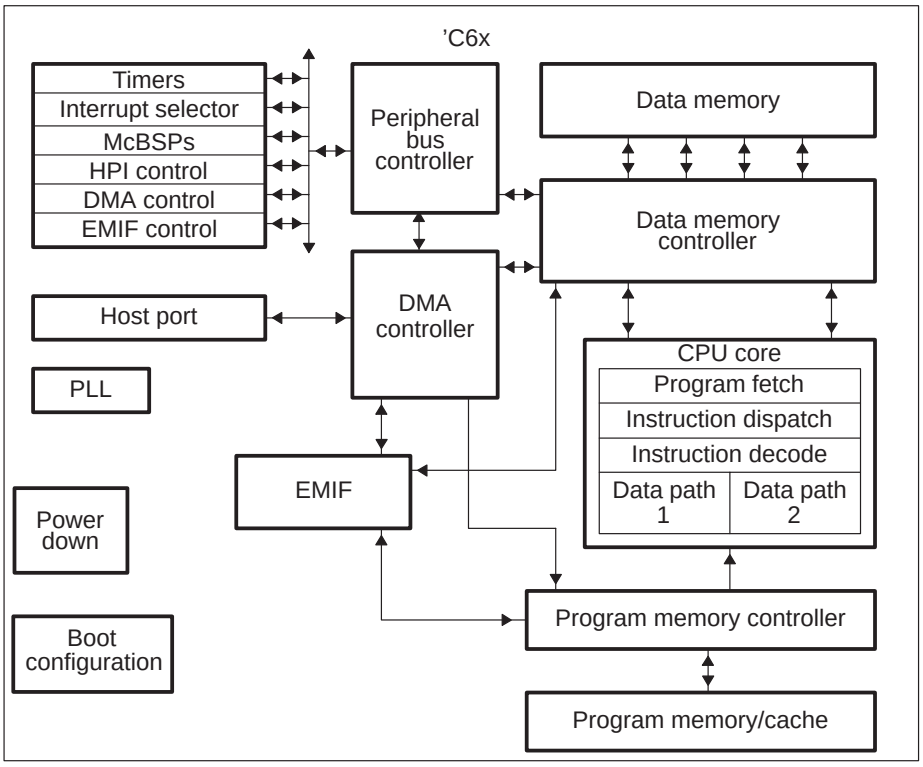
Topic **Page**

Peripherals for the ’C6x devices include:

- Direct memory access (DMA) controller
- Host-port interface (HPI)
- External memory interface (EMIF)
- Boot configuration
- Multichannel buffered serial ports (McBSPs)
- Interrupt Selector
- 32-bit timers
- Power-down logic

Figure 4–1 shows the block diagram for peripherals for the ’C6x devices.

Figure 4–1. TMS320C6x Block Diagram



4.1 Direct Memory Access (DMA)

The direct memory access (DMA) controller transfers data between regions in the memory map without intervention by the CPU. The DMA allows movement of data to and from internal memory, internal peripherals, or external devices to occur in the background of CPU operation. The DMA has four independently programmable channels allowing four different contexts for DMA operation. In addition a fifth (auxiliary) channel allows the DMA to service requests from the host-port interface (HPI). In discussing DMA operations several terms are important:

- **Read transfer:** The DMA reads the data element from a source location in memory.
- **Write transfer:** The DMA writes the data element that was read during a read transfer to its destination location in memory.
- **Element transfer:** The combined read and write transfer for a single data element.
- **Frame transfer:** Each DMA channel has an independently programmable number of elements per frame. In completing a frame transfer, the DMA moves all elements in a single frame.
- **Block transfer:** Each DMA channel also has an independently programmable number of frames per block. In completing a block transfer, the DMA moves all frames it has been programmed to move.

The DMA has the following features:

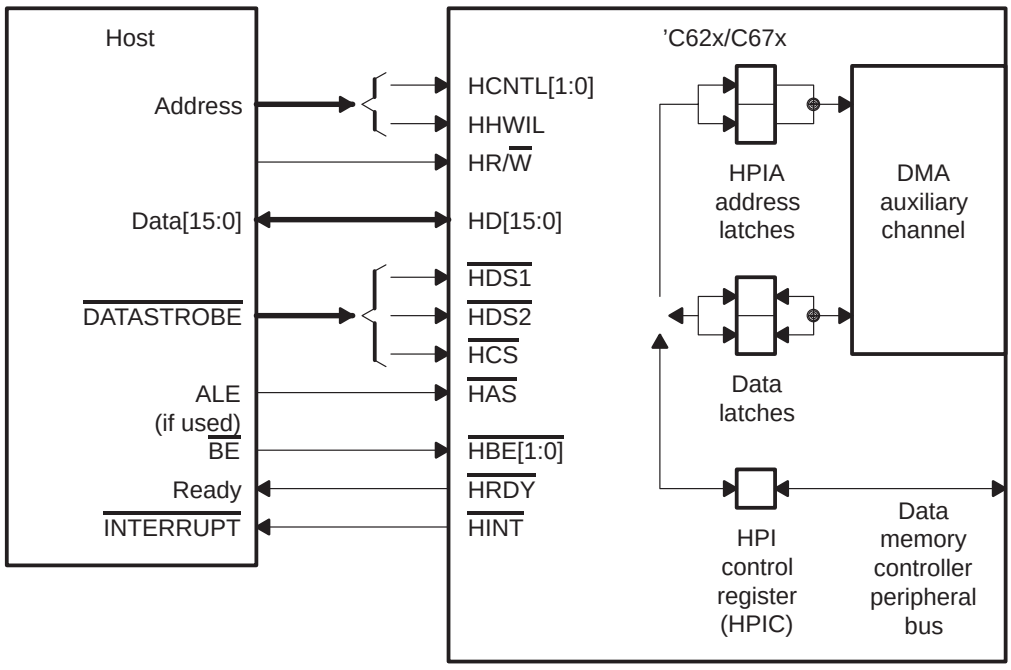
- **Background operation:** The DMA operates independently of the CPU.
- **High throughput:** Elements can be transferred at the CPU clock rate.
- **Four channels:** The DMA can keep track of the contexts of four independent block transfers.
- **Auxiliary channel:** This channel allows the host port to make requests into the CPU's memory space. This chapter discusses how the auxiliary channel requests are prioritized relative to other channels and the CPU. Detailed explanation of how it is used in conjunction with a peripheral is found in that peripheral's documentation.
- **Split operation:** A single channel may be used to simultaneously perform both the receive and transmit element transfers to or from two peripherals and memory, effectively acting like two DMAs.
- **Multiframe transfer:** Each block transfer can consist of multiple frames of a programmable size.

- **Programmable priority:** Each channel has independently programmable priorities versus the CPU.
- **Programmable address generation:** Each channel's source and destination address registers can have configurable indexes for each read and write transfer. The address may remain constant, increment, decrement, or be adjusted by a programmable value. The programmable value allows a distinct index for the last transfer in a frame and for the preceding transfers. This feature is used for multichannel sorting.
- **Full-address 32-bit address range:** The DMA can access any region in the memory map:
  - The on-chip data memory.
  - The on-chip program memory when mapped into memory space rather than being utilized as cache.
  - The on-chip peripherals.
  - The external memory interface (EMIF).
- **Programmable width transfers:** Each channel can be independently configured to transfer either bytes, 16-bit halfwords, or 32-bit words.
- **Autoinitialization:** Once a block transfer is complete, a DMA channel may automatically reinitialize itself for the next block transfer.
- **Event synchronization:** Each read, write, or frame transfer may be initiated by selected events.
- **Interrupt generation:** On completion of each frame transfer or of an entire block transfer, as well as on various error conditions, each DMA channel may send an interrupt to the CPU.

4.2 Host-port Interface (HPI)

The host-port interface (HPI) is a 16-bit-wide parallel port through which a host processor can directly access the CPU's memory space. The host device functions as a master to the interface, which increases ease of access. The host and CPU can exchange information via internal or external memory. The host also has direct access to memory-mapped peripherals. Connectivity to the CPU's memory space is provided through the DMA controller. Dedicated address and data registers not accessible to the CPU connect the HPI to the DMA auxiliary channel, which connects the HPI to the CPU's memory space. The HPI and the CPU can access the HPI control register (HPIC). The host can access the host address register (HPIA) and the host data register (HPID) as well, as the HPIC, using the external data and interface control signals. Figure 4–2 is a simplified diagram of the interface between the host and the 'C62x/C67x HPI.

Figure 4–2. Host-port Interface (HPI) Block Diagram



The HPI provides 32-bit data to the CPU with an economical 16-bit external interface by automatically combining successive 16-bit transfers. When the host device transfers data through HPID, the DMA auxiliary channel accesses the CPU's address space. The HPI supports high speed consecutive host accesses.

The 16-bit data bus, HD[15:0], exchanges information with the host. Because of the 32-bit word structure of the chip architecture, all transfers with a host consist of two consecutive 16-bit halfwords. On host data (HPID) write accesses, the $\overline{\text{HBE}}$[1:0] byte enables select the bytes in a 32-bit word to be written. For HPIA, HPIC, and HPID read accesses the byte enables are not used. The dedicated HHWIL pin indicates whether the first or second halfword is being transferred. An internal control register bit determines whether the first or second halfword is placed into the most significant halfword of a word.

The two data strobes ($\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$), the read/write select (HR/$\overline{\text{W}}$), and the address strobe ($\overline{\text{HAS}}$) enable the HPI to interface to a variety of industry-standard host devices with little or no additional logic required. The HPI can easily interface to hosts with multiplexed or dedicated address/data bus, a data strobe and a read/write strobe, or two separate strobes for read and write.

The host can access HPID with an optional automatic address increment of HPIA. This feature facilitates reading or writing to sequential word locations.

The HPI ready pin ($\overline{\text{HRDY}}$) allows insertion of host wait states. Wait states may be necessary depending on the latency of the memory accessed via the HPI, as well as on the rate of host access.

For more information on the host-port interface, see the *TMS320C6201/C6701 Peripherals Reference Guide*.

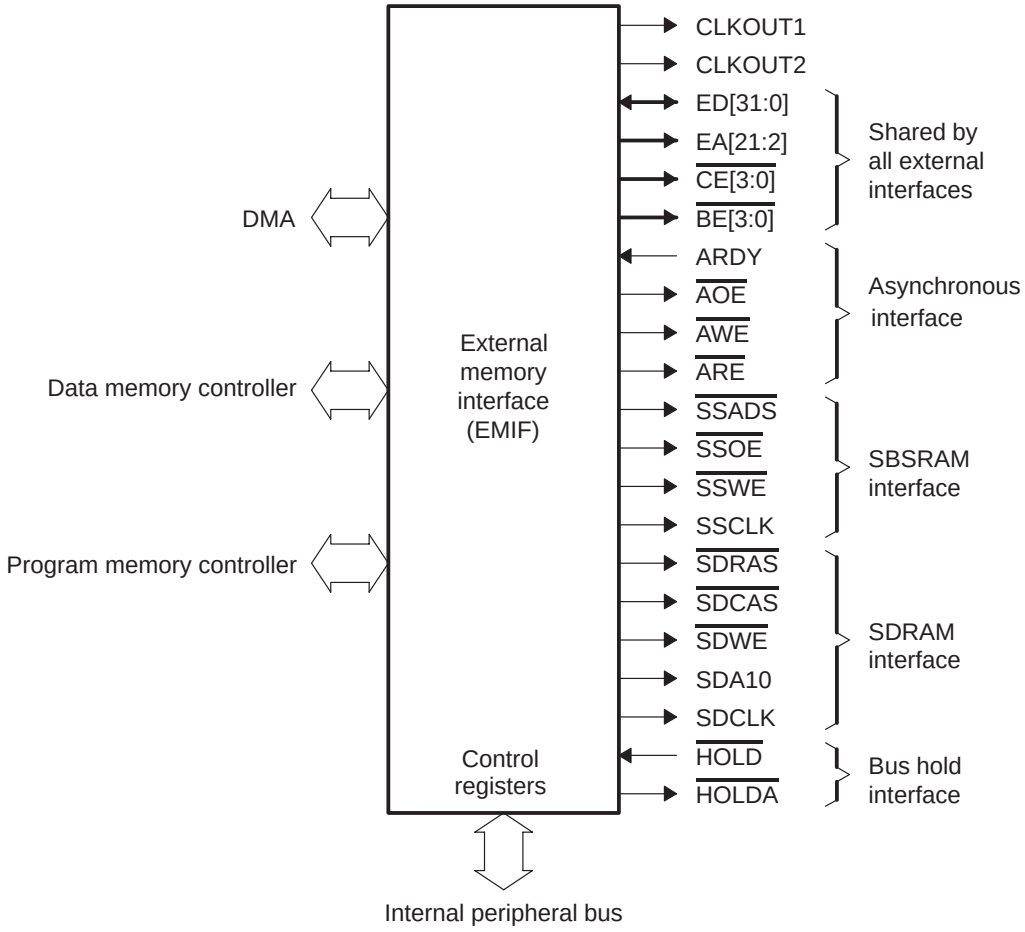
4.3 External Memory Interface (EMIF)

The external memory interface (EMIF) supports a glueless interface to several external devices, including:

- Synchronous burst SRAM (SBSRAM) running at the full rate and at half the rate of the CPU clock
- Synchronous DRAM (SDRAM) running at half the CPU clock rate
- Asynchronous devices, including asynchronous SRAM, ROM, and FIFOs. The EMIF provides highly programmable timing to these interfaces.
- An external shared-memory device

A block diagram of the EMIF is shown in Figure 4–3.

Figure 4–3. External Memory Interface (EMIF) Block Diagram



4.3.1 EMIF Registers

Control of the EMIF and the memory interfaces it supports is maintained through a set of memory-mapped registers within the EMIF. A write to any EMIF register will not finish until all pending EMIF accesses that use the register have finished. The memory-mapped registers are shown in Table 4–1.

Table 4–1. External Memory Interface (EMIF) Memory-Mapped Registers

Byte Address	Name
0180 0000h	EMIF global control
0180 0004h	EMIF $\overline{\text{CE1}}$ space control
0180 0008h	EMIF $\overline{\text{CE0}}$ space control
0180 000Ch	Reserved
0180 0010h	EMIF $\overline{\text{CE2}}$ space control
0180 0014h	EMIF $\overline{\text{CE3}}$ space control
0180 0018h	EMIF SDRAM control
0180 001Ch	EMIF SDRAM refresh period

The four $\overline{\text{CE}}$ space control registers correspond to the four $\overline{\text{CE}}$ spaces supported by the EMIF. The MTYPE field in each register identifies the memory type for the corresponding $\overline{\text{CE}}$ space. If MTYPE selects SDRAM or SBSRAM, the remaining fields in the register do not apply.

The SDRAM control register controls SDRAM parameters for all $\overline{\text{CE}}$ spaces that specify an SDRAM memory.

The SDRAM refresh period register controls the refresh period for SDRAM in terms of CLKOUT2 cycles (half the CPU clock rate). Optionally, the refresh period register can send an interrupt to the CPU. Thus, this register can be used as a general-purpose timer if SDRAM is not used by the system. The counter value can be read by the CPU.

4.3.2 SDRAM Interface

The EMIF supports 2-bank 16M-bit and 4-bank 64M-bit SDRAM, which offers system designers an interface to high-speed and high-density memory. Figure 4–4 illustrates the EMIF to SDRAM interface. Table 4–2 lists the suported memory configurations. The EA pins starting from pin13 connect to the SDRAM address pins starting at pin 11. The symbol m is 0 for a 16M bit interface and 2 for 64M bit interface.

Figure 4–4. EMIF to SDRAM Interface

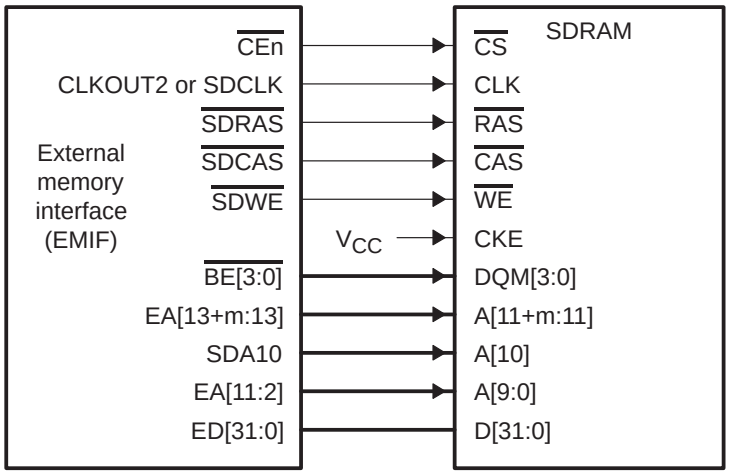


Table 4–2. SDRAM Memory Population

SDRAM Size	SDRAM Banks	SDRAM Width	Devices per CE Space	Memory Size per CE Space
16M bits	2	16 bits	2	4M bytes
16M bits	2	8 bits	4	8M bytes
64M bits	4	16 bits	2	16M bytes

During an SDRAM read, the selected bank is activated with the row address during the ACTV command. The EMIF uses a CAS latency of 3 and a burst length of 1. During read cycles, the 3-cycle latency causes data to appear three cycles after the corresponding column address.

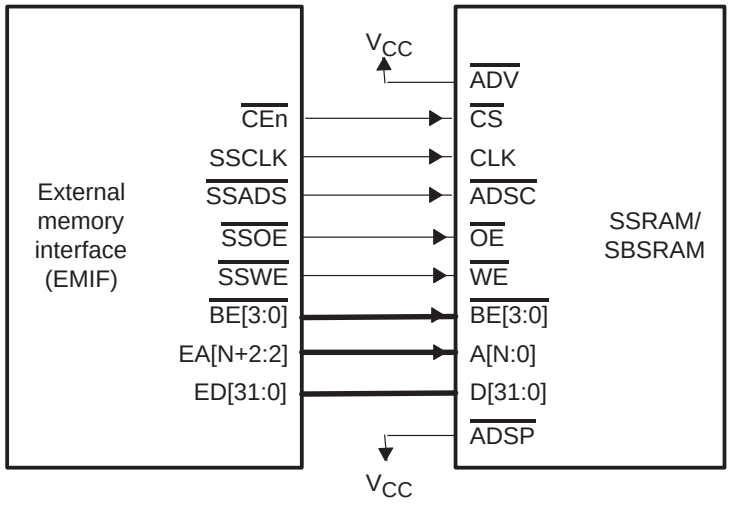
All SDRAM writes have a burst length of 1. The bank is activated with the row address during the ACTV command. There is no latency associated with writes, so data is output on the same cycle that the column address is received. Writes to invalid bytes are disabled via the appropriate DQM inputs. This feature allows for byte and halfword writes.

4.3.3 SBSRAM Interface

The EMIF interfaces directly to industry-standard synchronous burst SRAMs, see Figure 4–5. This memory interface allows a high-speed memory interface without some of the limitations of SDRAM. Since SBSRAM are SRAM devices, random accesses are possible during burst reads or writes. The SBSRAM interface can run at either the CPU clock speed or at half of this rate.

The four SBSRAM control pins are latched by the SBSRAM on the rising SSCLK edge to determine the current operation. These signals are valid only if the chip select line for the SBSRAM is low.

Figure 4–5. EMIF to SBSRAM Interface



4.3.4 Asynchronous Interface

The asynchronous interface offers configurable cycle types, which can be used to interface to a variety of memory and peripheral types, including SRAM, EPROM, and Flash memory, as well as FPGA and ASIC devices.

The following three figures show interfaces to SRAM (Figure 4–6), to FIFOs (Figure 4–7), and to ROM (Figure 4–8). Although ROM can be interfaced at any of the $\overline{CE}$ spaces, it is often used at $\overline{CE1}$, because only that space can be configured for widths smaller than 32 bits.

Figure 4–6. EMIF to SRAM Interface

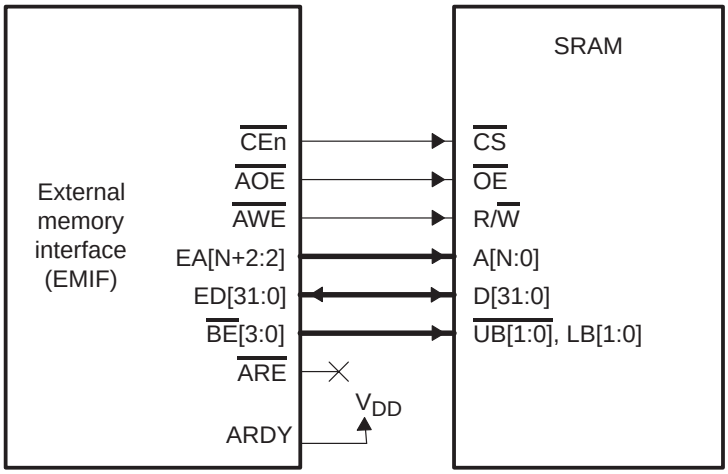


Figure 4–7. EMIF to FIFO Interface

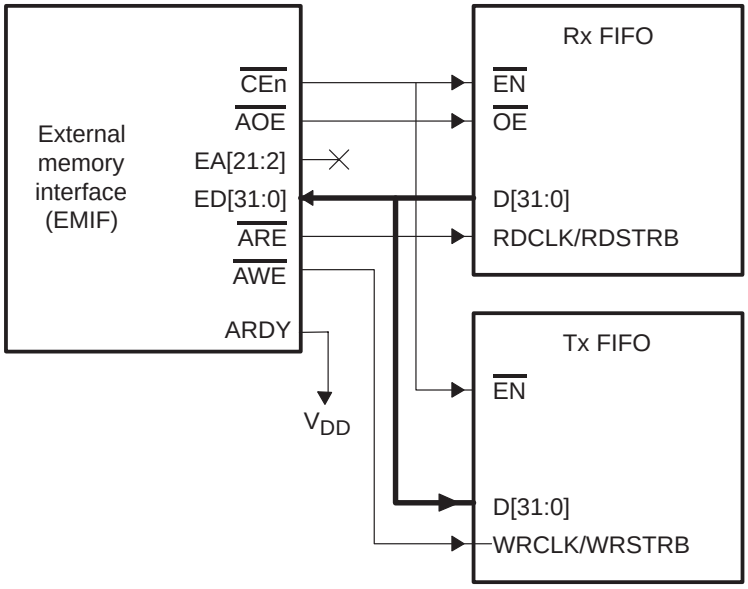
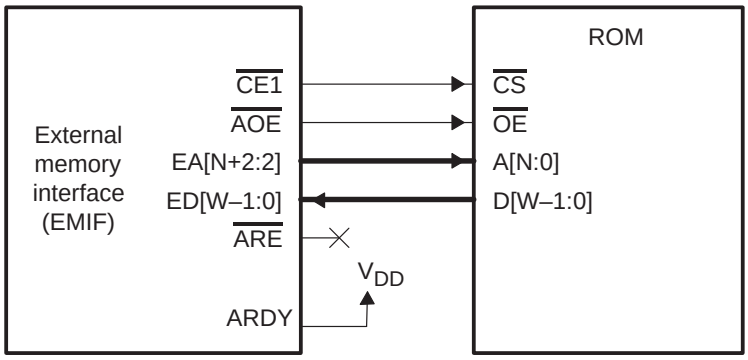


Figure 4–8. EMIF to ROM Interface



The EMIF supports 8-, 16-, and 32-bit-wide ROMs. In Figure 4–8 the W in ED to D lines denotes the number of data bits of the ROM. The access modes are selected by the MTYPE field in the EMIF $\overline{CE}$ space control register. In reading data from these narrow-width memory spaces, the EMIF packs multiple reads into one 32-bit-wide value. This mode is primarily intended for word access to 8-bit and 16-bit ROM devices. Thus, the following restrictions apply:

- Read operations always read 32 bits, regardless of the access size or the memory width.
- The address is shifted up appropriately to provide the correct address to the narrow memory. The shift amount is 1 for 16-bit ROM and 2 for 8-bit ROM. Thus, the high address bits are shifted out and accesses wrap around if that $\overline{CE}$ space spans the entire EA bus.
- The EMIF always reads the lower addresses first and packs these into the LSBytes. The EMIF packs subsequent accesses into the higher order bytes. Thus, the expected packing format in ROM is always little-endian regardless of the value of the LENDIAN bit.

In 8-bit ROM mode, the address is left shifted by 2 to create a byte address on EA to access byte-wide ROM. The EMIF always packs four consecutive bytes aligned on a 4-byte (byte address = 4N) boundary into a word access. The bytes are fetched in the address order: 4N, 4N + 1, 4N + 2, 4N + 3. Bytes are packed into the 32-bit word in the following little-endian order, from MSByte to LSByte: 4N + 3, 4N + 2, 4N + 1, 4N.

In 16-bit ROM mode, the address is left shifted by 1 to create a halfword address on EA to access 16-bit-wide ROM. The EMIF always packs two consecutive halfwords aligned on a 4-byte (byte address = 4N) boundary into a word access. The halfwords are fetched in the address order: 4N, 4N + 2. Halfwords are packed into the 32-bit word in the following little-endian order, from most significant halfword to least significant halfword: 4N + 2, 4N.

4.4 Boot Configuration Logic

The ’C62x and ’C67x provide a variety of boot configurations for proper device initialization. These configurations determine what actions the ’C62x/C67x performs after device reset to prepare for initialization. These boot configurations, which are set by external input pins, determine:

- The memory map the device selects. The memory map determines whether internal or external memory is mapped at address 0.
- The type of external memory at address 0 (if external memory is mapped at address 0)
- The boot process used to initialize the memory at address 0 before the CPU is allowed to run

4.4.1 Device Reset

The external device reset is the active-low $\overline{\text{RESET}}$ signal. While $\overline{\text{RESET}}$ is low, the device is held in reset. During this period the device is initialized to the prescribed reset state. All 3-state outputs are placed into the high-impedance state. All other outputs are returned to their default state. $\overline{\text{RESET}}$ is latched with the device CLKIN signal, as well as with the CPU clock. Thus, $\overline{\text{RESET}}$ has minimum low time in terms of CLKIN as well as CPU clock (CLKOUT1) cycles. The precise timing requirements for device reset are described in the data sheet for each particular device. The rising edge of $\overline{\text{RESET}}$ starts the processor running with the prescribed boot configuration.

4.4.2 Boot Configuration

External pins BOOTMODE[4:0] determine the boot configuration. The values of BOOTMODE[4:0] are latched with the rising edge of $\overline{\text{RESET}}$. These signals must be valid for the proper setup and hold times. See the data sheet for specific timing requirements.

Three types of boot processes are available:

- **No boot process(direct-execution startup):** The CPU simply starts running from the memory located at address 0. When this memory location resides on SDRAM, the CPU is held until SDRAM initialization finishes.
- **ROM boot process:** The memory located at the beginning of external memory space $\overline{CE1}$ (see Figure 3–1 on page 3-2) is copied to address 0 by the DMA controller. Although the boot process begins when the device is released from external reset, this transfer occurs while the CPU is held in reset internally. The amount of memory copied is 16K words of 32 bits each. This process lets you choose the width of the ROM. In this case, the EMIF can automatically assemble consecutive 8-bit bytes or 16-bit half-words to form the 32-bit instruction words to be moved. These values are expected to be stored in little-endian format in the external memory, which is typically a ROM device.
- **HPI boot process:** In the HPI (host-port interface) boot process, the CPU is held in reset while the remainder of the device is released from reset. During this period, an external host can initialize the CPU's memory space as necessary through the HPI, including external memory configuration registers. Once the necessary external memory has been configured, the host can access any external sections it needs to complete initialization. Once the host is finished with all necessary initialization, the host writes a 1 to the DSPINT bit in the HPI control register (HPIC). This write causes an active transition on the DSPINT signal. In turn, this transition causes the boot configuration logic to remove the CPU from its reset state. The CPU then begins running from address 0. The DSPINT condition is not latched by the CPU, because it occurs while the CPU is still in reset. Also, DSPINT wakes up the CPU from internal reset only if the HPI boot process is selected.

4.5 Multichannel Buffered Serial Port (McBSP)

The ’C62x/C67x multichannel buffered serial port (McBSP) is based on the standard serial port interface found on the TMS320C2x, ’C2xx, ’C5x, and ’C54x devices. The standard serial port interface provides:

- Full-duplex communication
- Double-buffered data registers, which allow a continuous data stream
- Independent framing and clocking for receive and transmit
- Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- External shift clock generation or an internal programmable frequency shift clock

In addition, the McBSP has the following capabilities:

- Direct interface to:
  - T1/E1 framers
  - MVIP and ST–BUS compliant devices
  - IOM-2 compliant devices
  - AC97 compliant devices
  - IIS compliant devices
  - SPI™ devices
- Multichannel transmit and receive of up to 128 channels.
- A wider selection of data sizes including 8, 12, 16, 20, 24, or 32 bits

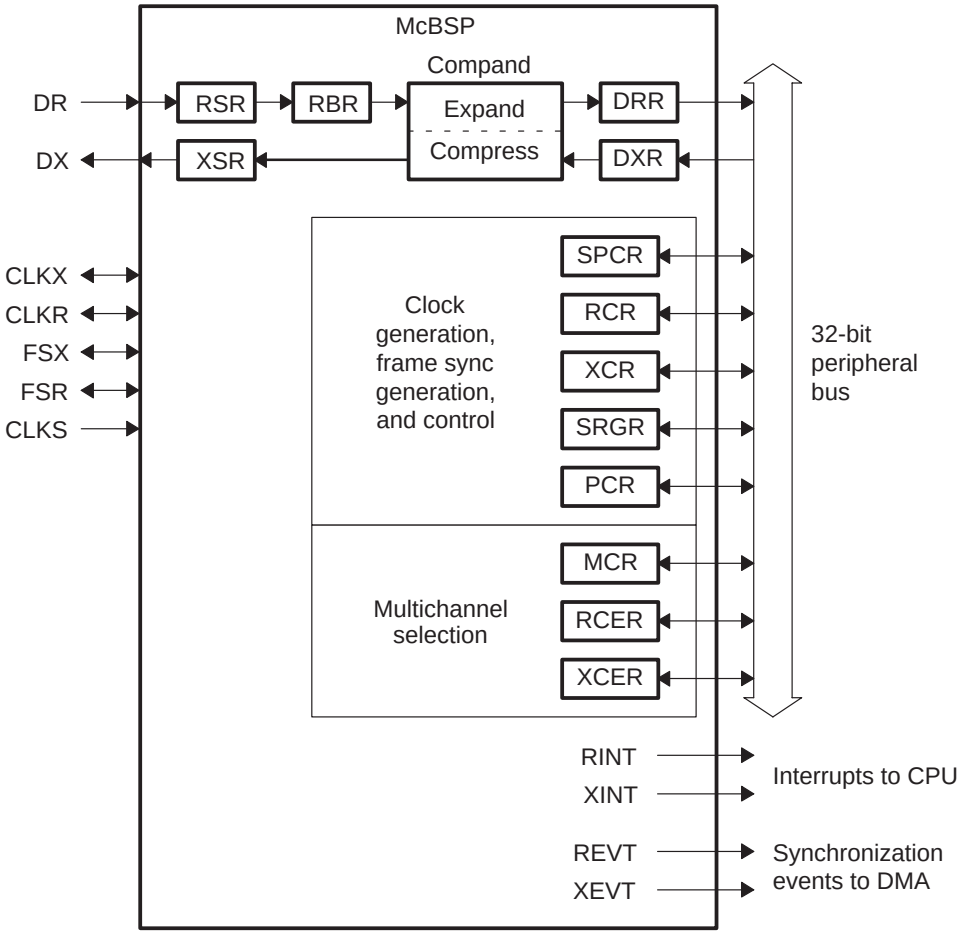
Note:

Data sizes are referred to as *word* or *serial word* throughout this document. Therefore, when *word* is used, it can be 8, 12, 16, 20, 24, or 32 bits, in contrast to the true definition of word as being 32 bits.

- µ-Law and A-Law companding
- 8-bit data transfers with LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- Highly programmable internal clock and frame generation

The McBSP consists of a data path and control path. Seven pins connect the control and data paths to external devices as shown in Figure 4–9.

Figure 4–9. Multichannel Buffered Serial Port (McBSP) Internal Block Diagram



Data is communicated to devices interfacing to the McBSP via the data transmit (DX) pin for transmit and the data receive (DR) pin for receive. Control information in the form of clocking and frame synchronization is communicated via CLKX, CLKR, FSX, and FSR. The peripheral device communicates to the McBSP via 32-bit-wide control registers accessible via the internal peripheral bus. The CPU or DMA controller reads the received data from the data receive register (DRR) and writes the data to be transmitted to the data transmit register (DXR). Data written to the DXR is shifted out to DX via the transmit shift register (XSR). Similarly, receive data on the DR pin is shifted into the receive shift register (RSR) and copied into the receive buffer register (RBR). RBR is then copied to DRR, which can be read by the CPU or the DMA controller. This

allows internal data movement and external data communications simultaneously. The remaining registers accessible to the CPU configure the control mechanism of the McBSP. These registers are listed in Table 4–3. The control block consists of internal clock generation, frame synchronization signal generation, control for both of these, and multichannel selection. This control block sends notification of important events to the CPU and the DMA controller via four signals as shown in Table 4–4.

Table 4–3. Multichannel Buffered Serial Port (McBSP) Registers

Abbreviation	Register Name
RBR	McBSP receive buffer register
RSR	McBSP receive shift register
XSR	McBSP transmit shift register
DRR	McBSP data receive register
DXR	McBSP data transmit register
SPCR	McBSP serial port control register
RCR	McBSP receive control register
XCR	McBSP transmit control register
SRGR	McBSP sample rate generator register
MCR	McBSP multichannel register
RCER	McBSP receive channel enable register
XCER	McBSP transmit channel enable register
PCR	McBSP pin control register

Table 4–4. Multichannel Buffered Serial Port (McBSP) CPU Interrupts and DMA Event Synchronization

Interrupt Name	Description
RINT	Receive interrupt to CPU
XINT	Transmit Interrupt to CPU
REVT	Receive synchronization event to DMA controller
XEVT	Transmit synchronization event to DMA controller

4.6 Timers

The ’C62x/C67x has two 32-bit general-purpose timers that you can use to:

- Time events
- Count events
- Generate pulses
- Interrupt the CPU
- Send synchronization events to the DMA controller

The timer has two signaling modes and can be clocked by an internal or an external source. The timer has an input pin (TINP) and an output pin (TOUT). The TINP can be used as a general-purpose input, and the TOUT pin can be used for a general-purpose output.

With an internal clock, the timer can signal an external A/D converter to start a conversion, or it can trigger the DMA controller to begin a data transfer. With an external clock, for example, the timer can count external events and interrupt the CPU after a specified number of events.

4.7 Interrupt Selector

The ’C62x/C67x peripheral set produces 16 interrupt sources. The CPU has 12 interrupts available for use. The interrupt selector allows you to choose which 12 of the 16 your system needs to use. The interrupt selector also allows you to effectively change the polarity of external interrupt inputs.

Table 4–5 lists the available interrupts.

Table 4–5. Peripheral Interrupts

Interrupt Selection Number	Interrupt Abbreviation	Interrupt Description
0000b	DSPINT	Host port host to DSP interrupt
0001b	TINT0	Timer 0 interrupt
0010b	TINT1	Timer 1 interrupt
0011b	SD_INT	EMIF SDRAM timer interrupt
0100b	EXT_INT4	External interrupt pin 4
0101b	EXT_INT5	External interrupt pin 5
0110b	EXT_INT6	External interrupt pin 6
0111b	EXT_INT7	External interrupt pin 7
1000b	DMA_INT0	DMA channel 0 interrupt
1001b	DMA_INT1	DMA channel 1 interrupt
1010b	DMA_INT2	DMA channel 2 interrupt
1011b	DMA_INT3	DMA channel 3 interrupt
1100b	XINT0	McBSP 0 transmit interrupt
1101b	RINT0	McBSP 0 receive Interrupt
1110b	XINT1	McBSP 1 transmit interrupt
1111b	RINT1	McBSP 1 receive interrupt

4.8 Power-Down Logic

Most of the operating power of CMOS logic is dissipated during circuit switching from one logic state to another. By preventing some or all of the chip's logic from switching, significant power savings can be realized without losing any data or operational context. Power-down mode PD1 blocks the internal clock inputs at the boundary of the CPU, preventing most of its logic from switching. PD1 effectively shuts down the CPU. Additional power savings are accomplished in power-down mode PD2, in which the entire on-chip clock structure (including multiple buffers) is halted at the output of the PLL. Power-down mode PD3 shuts down the entire internal clock tree (like PD2) and also disconnects the external clock source (CLKIN) from reaching the PLL. Wake-up from PD3 takes longer then wake-up from PD2 because the PLL needs to be relocked, just as it does following power up.

PD2 and PD3 assert the $\overline{\text{PD}}$ pin for external recognition of these two power-down modes. In addition to power-down modes, the IDLE instruction provides lower CPU power consumption by executing multiple NOPs. The IDLE instruction terminates only upon servicing an interrupt.

Chapter 5

Development Support

The TMS320C62x and TMS320C67x design environment reflects the unique nature of the advanced VLIW architecture. The environment includes code generation tools, evaluation tools, documentation, online help with various tools, and a web site on the Internet (www.ti.com/sc/C6x) with complete technical documentation.

Topic **Page**

5.1 Code Generation Tools

A complete development tool set for both the PC and Sun workstations includes the following:

- C compiler
- Assembly optimizer
- Assembler
- Linker
- Evaluation tools

The environment is founded on the generation's highly advanced C compiler and TI's revolutionary assembly optimizer. Figure 5–1 shows a flow of the process to develop code.

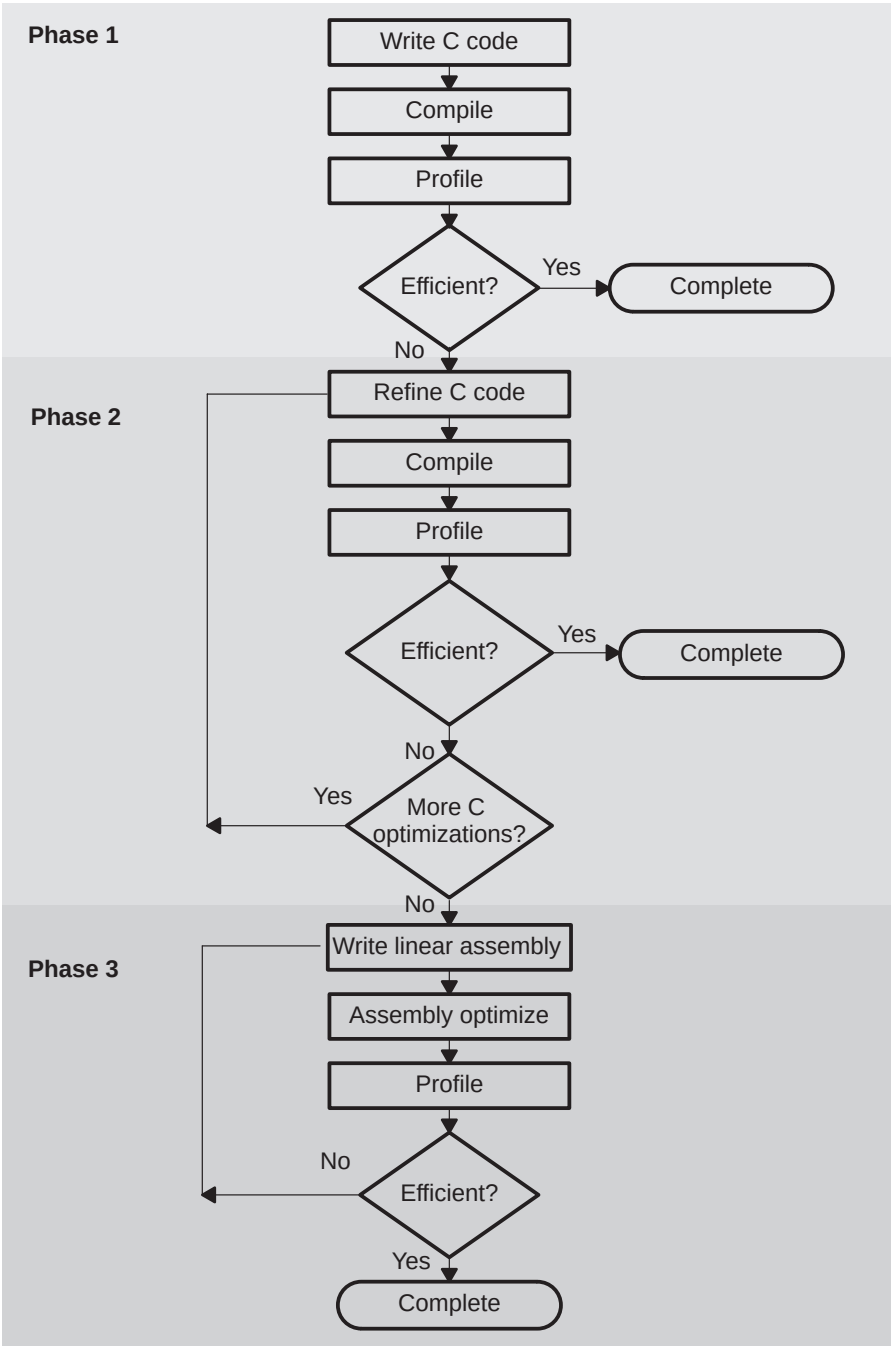
The 'C6x generation's C compiler eliminates the need for extensive knowledge of DSP architecture, allowing you to take full advantage of the world's most powerful DSP. This highly-structured, architecture-independent C code development environment dramatically reduces development time for new products. At the same time, it maintains the inherent performance benefits of the 'C6x generation's advanced VLIW architecture. The 'C6x compiler offers up to a 3X improvement in efficiency over existing fixed-point C compilers for DSP.

For application code sections that require the fine tuning of assembly code, the 'C6x generation's unique assembly optimizer provides the same transparent programming capability as the C compiler. The tool supports automatic scheduling, optimizing, and separation of fine-grained parallel tasks from linear assembly code, delivering a level of simplicity and power that is unprecedented in assembly-level tools.

The tools take C or assembly source code and implement many different optimizations, including software pipelining, to intelligently find and exploit the unique instruction-level parallelism of the 'C6x. After each step in the process, the 'C6x tools allow you to evaluate their results and take appropriate steps to achieve the highest level of parallelism in your code.

Initially, all C code — new or reused from other applications — is run through the C compiler for the 'C6x. Using the evaluation tools described in the following section, you can evaluate the code for efficiency. If the performance is sufficient for the particular application, then the application has been completed, achieving the fastest possible time-to-market and incurring minimal engineering cost.

Figure 5–1. Code Development Flow Chart



A designer who needs to improve code efficiency can use intrinsics, command-line options, and source-code enhancements:

- The ’C6x design tools feature two sets of intrinsics. The first set includes intrinsics that perform DSP-specific operations that are not supported directly in C. The second set is designed to facilitate 16-bit operation on a 32-bit machine. These intrinsic functions can be invoked to tune the performance of the C code. Some of the most commonly used intrinsics are described in Table 5–1.

Table 5–1. Selected TMS320C6x C Compiler Intrinsics

C Compiler Intrinsic	Assembly Instruction	Description
uint _**clr(**uint *src2,* uint *csta*, uint *cstb***);**	**CLR**	Clears the specified field in src2. The beginning and ending bits of the field to be cleared are specified by csta and cstb, respectively.
int _**ext(**uint *src2,* uint *csta*, int *cstb***);**	**EXT**	Extracts the specified field in src2, sign-extended to 32 bits. The extract is performed by a shift left followed by a signed shift right; csta and cstb are the shift left and shift right amounts, respectively.
uint _**lmbd(**uint s*rc1,* uint *src2***):**	**LMBD**	Searches for a leftmost 1 or 0 of src2 determined by the LSB of src1. Returns the number of bits up to the value change.
int _**mpy(**int *src1,* int *src2***);**	**MPY**	Multiplies the 16 LSBs of src1 by the 16 LSBs of src2 and returns the result. Values can be signed or unsigned.
int _**sadd(**int *src1,* int *src2***);**	**SADD**	Adds src1 to src2 and saturates the result. Returns the result.
uint _**set(**uint *src2,* uint *csta*, uint *cstb***);**	**SET**	Sets the specified field in src2 to all 1s and returns the src2 value. The beginning and ending bits of the field to be set are specified by csta and cstb, respectively.
uint _**subc(**uint *src1,* uint *src2***);**	**SUBC**	Conditional subtract divide step

- You can experiment with several command-line options that cause the compiler to perform more aggressive optimization. One particularly useful option instructs the compiler to compile an entire application at once, giving the compiler visibility across program sections and more knowledge of the way in which variables and functions are used. Another option causes the compiler to perform global optimizations across an entire application.

- Source-code enhancements can be made to exploit specific features of the ’C6x architecture. For example, the ’C6x has support for operating on words containing two 16-bit quantities; therefore, you can use 32-bit loads and stores when operating on arrays containing 16-bit data and easily achieve a 2X performance improvement.

Taken together, these actions result in a large amount of parallelism in C code.

For ultra-high performance applications, extracting every last bit of throughput from the application code may be necessary. The profiler can identify critical code segments that might benefit most from being generated in assembly language.

For these program sections, you write simple, linear ’C6x assembly code that is input to the assembly optimizer. This assembly code is ’C6x instructions written without concern for parallel instructions, instruction latencies, or register usage.

The assembly optimizer tool schedules the instructions, taking into account the architectural parallelism. The tool honors ’C6x latency requirements, maximizes parallel code, and performs register allocation.

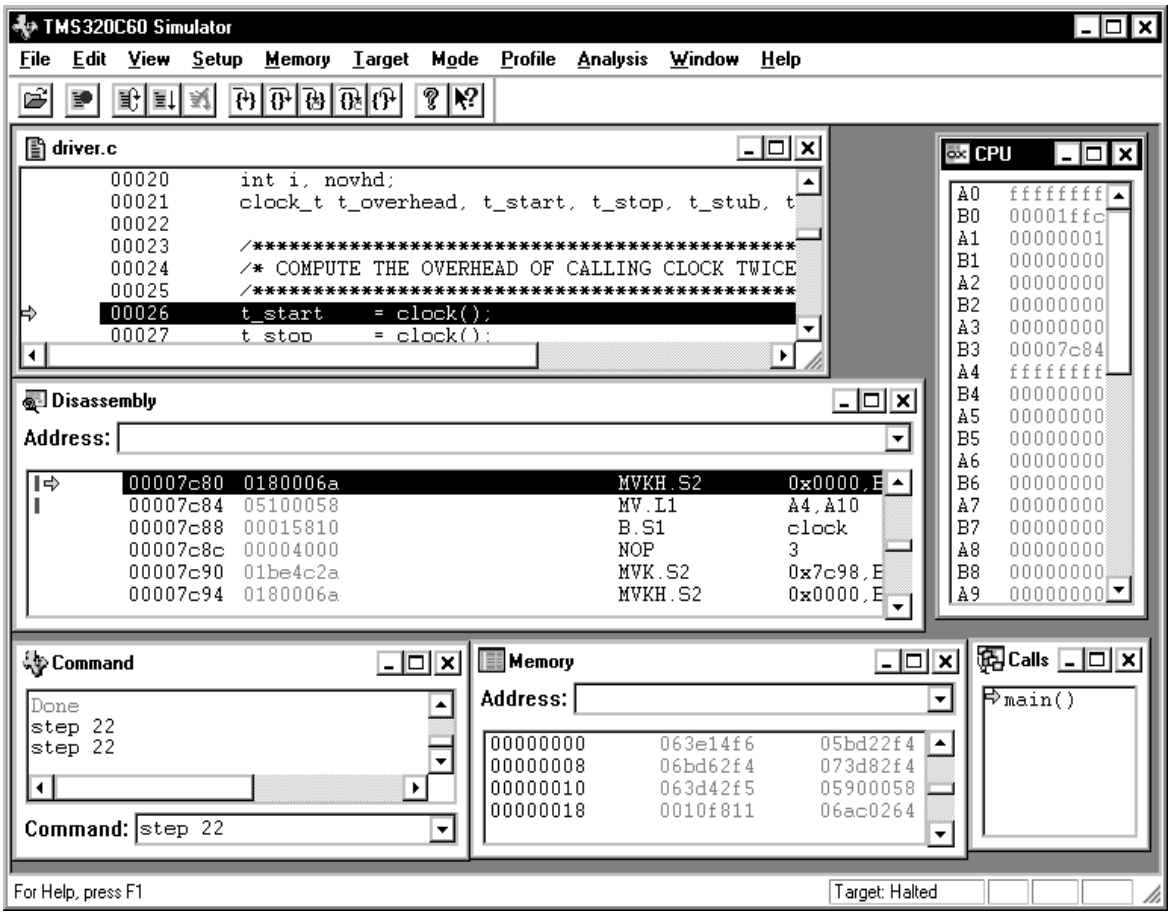
5.2 Evaluation Tools

The evaluation tools include the following items:

- Windows debugger interface
- Simulator
- Hardware emulation board

The ’C6x development environment provides a new intuitive Windows-based graphical user interface (GUI) for debugging. The debugger interface features windows for source, assembly, call stack, memory, registers, and watch expressions, as well as menu and tool bars. The debugger offers one-click breakpoint setting and dialogs for editing breakpoints. The debugger also incorporates a dynamic profiler to help you find bottlenecks and improve code efficiency. Figure 5–2 shows the C debugger’s basic Windows interface.

Figure 5–2. Windows C Debugger Interface



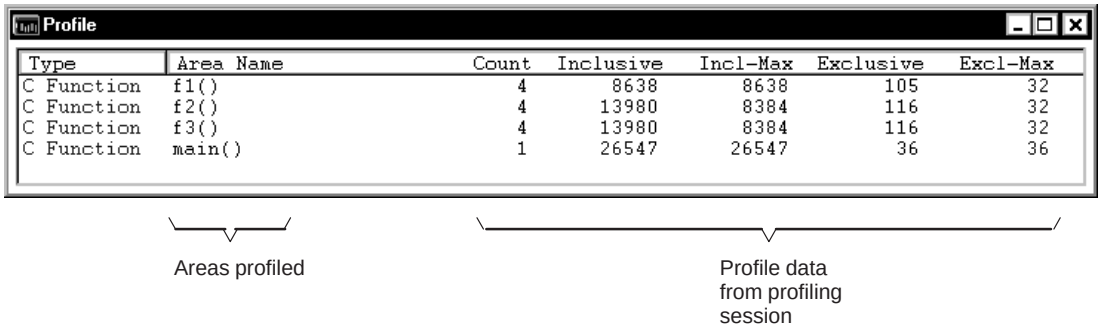
TI provides ’C6201 scan-based emulation systems that support hardware and software debugging of target systems via a JTAG-emulation cable. Scan-based emulation is a unique, nonintrusive approach to system emulation, integration, and debugging.

Initially, TI is offering a stand-alone ’C6201 test and emulation board (TEB) that interfaces with the host platform through the XDS510™ and XDS510WS™ emulators through the IEEE Standard 1149.1 (JTAG)-compliant port. The board features a prototyping area for adding user-defined peripherals. With the addition of other ’C6x generation members, TI will continually add functionality to the common development environment as well. Capabilities will ultimately include a PC plug-in evaluation module (EVM) board, a low-cost PC-based board that is well-suited for software algorithm development.

The dynamic profiler integrated into the ’C6x debugger creates cycle histograms that are continuously updated as the code runs. It can show graphically which functions, ranges, and lines in an application are performance bottlenecks.

The statistics collected during a profiling session are displayed in the Profile window. Figure 5–3 shows an example of this window.

Figure 5–3. An Example of the Profile Window



You can modify the Profile window to display selected profile areas or different data; you can also sort the data differently.

A timing display can be built into the application by inserting a few function calls in the code. The resulting simple cycle counts, obtained without using the profiler or the debugger, can be printed automatically to allow you to track the changes in execution speed of an algorithm over time. This output, while less sophisticated, is continuously available with no further action.

5.3 Third-Party Support

TI has a long history of strong third-party support and this continues with the ’C62x/C67x devices. Table 5–2 lists the third-party contacts supporting the ’C62x/C67x devices and their product areas with telephone numbers and electronic mail addresses.

Table 5–2. Contacts for Third-Party Support

Third-Party Contact	Product Area	Phone Number	e-mail Address
Ariel Corporation	High-performance VME64 platform and computer telephony products	609 860–2900	ariel@ariel.com
Cheops GmbH & Co KG	Industrial and medical imaging and high speed/high resolution video conferencing	49 8861 2369 0	email@cheops-bv.de
D2 Technologies, Inc.	Embedded Voice Processing (EVP™) computer telephony software	805 564–3424	blandon@d2tech.com
DSP Research, Inc.	TIGER development boards and OEM systems	408–773–1042	info@dspr.com
DSP Software Engineering, Inc.	Multichannel V.34bis modem and telecom software	617–275–3733	info@dspse.com
Eonic Systems, Inc.	Real-time operating systems — Virtuoso Nano™, Classico™, and MicroLite™	301–572–5000	info@eonic.com
GO DSP Corporation	Code Composer™ support and next generation development tool, Code Maestro™	416–599–6868	gdasilva@go–dsp.com
HotHaus Technologies, Inc.	HausWare — DSP software architecture for embedded telecommunications applications	604–278–4300	info@hothaus.com
Innovative Integration, Inc.	PCI6201 DSP coprocessor for telecom, communications, and data acquisition applications	818–865–6150	techsprt@innovative-dsp.com
Loughborough Sound Images	PCI/C6200 — signal processing platform and PCI/C6220 telecommunications/high density DSP telephony platform	+44 0 1509 634444	
Pentek, Inc	Scalable multiprocessor board for the VMEbus (model 9134)	201–818–5900	info@pentek.com

Table 5–2. Contacts for Third-Party Support (Continued)

Third-Party Contact	Product Area	Phone Number	e-mail Address
Signals & Software Ltd. (SASL)	Very high density ISP modem solution	44 181 426 9533	davem@sasl.demon.co.uk
Spectrum Signal Processing	Hardware, interface silicon, and CTI software for DSPs	604–421–5422	sales@spectrumsignal.com
Spectron Micro-systems	Real-time SPOX operating-systems	805–968–5100	info@spectron.com
Radisys, Inc.	SPIRIT-6000 series of high-performance board-level platforms and software development tools	617–235–6824	info@radisys.com
ViaDSP, Inc.	InvisiLink™ line of software and firmware for high density computer telephony boards	508–369–0048	dpenny@viadsp.com
White Mountain DSP, Inc.	Emulation and multiplatform debug support. Mountain-510, Mountain-510/WS and Mountain-510/LT PCMCIA Card	603–883–2430	info@wmdsp.com

5.4 Web Site and Documentation

Visit the web site at **www.ti.com/sc/C6x** for information, an interactive multimedia technical overview (MeTO), documentation, and schedule of 'C6x design workshops. MeTO describes features of the devices in a visual way with graphics in a point-and-click display for ease of navigation. The web site offers a complete training schedule of design workshops and seminars. Applications assistance and frequently asked questions (FAQ) are also on the web site.

Documentation is available directly from the web site in downloadable files for printing. There is a complete list of documentation available in this book's *Preface* under *Related Documentation From Texas Instruments*.

Appendix A

Glossary

A

address: The location of program code or data stored; an individually accessible memory location.

ALU: See *arithmetic logic unit.*

application-specific integrated circuit (ASIC): A custom chip designed for a specific application. It is designed by integrating standard cells from a library.

arithmetic logic unit (ALU): The hardware of the CPU that performs arithmetic and logic function.

assembler: A software program that creates a machine-language program from a source file that contains assembly language instructions, directives, and macro definitions. The assembler substitutes absolute operation codes or relocatable codes for symbolic addresses.

assembly optimizer: A software program that optimizes linear assembly code, which is assembly code that has not been register-allocated or scheduled. The assembly optimizer is automatically invoked with the shell program, cl6x, when one of the input files has an .sa extension.

ASIC: See *application-specific integrated circuit.*

B

boot: The process of loading a program into memory.

boot configuration: A set of parameters defining how a device is booted.

boot loader: A built-in segment of code that transfers code from an external source to program memory at power up.

C

cache: A fast storage buffer in the central processing unit of a computer.

central processing unit (CPU): The unit that coordinates the functions of a processor.

circular addressing: An address mode in which a finite set of addresses is reused by linking the largest address back to the smallest address.

clock cycles: A periodic or sequence of events based on the input from the external clock.

code: A set of instructions written to perform a task; a computer program or part of a program.

compiler: A computer program that translates programs in a high-level language into their assembly-language equivalents.

control register: A register that contains bit fields which define the way a device operates.

control register file: A set of control registers.

CPU: See *central processing unit.*

crosspath: A link between register files to provide communication between the CPU units.

D

data memory: A region of memory used for storing or manipulating data, separate from the region used for storing program code.

direct memory access (DMA): Memory access that does not use the CPU; used for data transfer directly between memory and a peripheral.

direct memory access (DMA) controller: Specialized circuitry that transfers data from memory to memory without using the CPU.

DMA: See *direct-memory access.*

DRAM: See *dynamic random-access memory.*

dynamic random-access memory (DRAM): Memory that can be read and written by the microprocessor and whose storage locations can be accessed in any order but must be refreshed (recharged) periodically to retain data or program code.

E

E1: A European high-speed network communication service that operates at 2.048M bits per second and uses A-law companding.

erasable programmable read-only memory (EPROM): A memory device whose contents are erasable (usually via UV light), and programmable.

execute packet: A group of instructions that execute in parallel.

external interrupt: A hardware interrupt triggered by a pin.

external memory interface (EMIF): Microprocessor hardware which is used to read to and write from off-chip memory.

F

field programmable gate array (FPGA): An integrated circuit that contains an array of gates that can be programmed after manufacture, typically at the time of installation.

first-in, first-out (FIFO): A method for managing a set of items to which additions and deletions are made; items are added to one end of the list and removed from the other.

fixed-point processor: A processor which does arithmetic operations using integer arithmetic with no exponents.

flash memory: Electronically erasable, programmable nonvolatile (read-only) memory.

floating-point processor: A processor capable of handling floating-point arithmetic where real operands are represented using exponents.

G

global interrupt enable bit (GIE): A bit in the control status register (CSR) that is used to enable or disable maskable interrupts.

H

halfword: For this device, a halfword is taken as a 16-bit data item taken as a unit.

hardware interrupt: An interrupt triggered through physical connections with on-chip peripherals or external devices.

host port interface (HPI): A parallel interface that the CPU uses to communicate with a host processor.

I

indirect addressing: Indirect addressing: An addressing mode in which an address points to another pointer rather than to the actual data; this mode is prohibited in RISC architecture.

instruction fetch packet: A group of up to eight instructions held in memory for execution by the CPU.

interrupt: A signal sent by hardware or software to a processor requesting attention. An interrupt tells the processor to suspend its current operation, save the current task status, and perform a particular set of instructions. Interrupts communicate with the operating system and prioritize tasks to be performed.

interrupt service fetch packet (ISFP): A fetch packet used to service interrupts. If eight instructions are insufficient, the user must branch out of this block for additional interrupt service. If the delay slots of the branch do not reside within the ISFP, execution continues from execute packets in the next fetch packet (the next ISFP).

L

latency: The delay between the occurrence of a condition and the reaction of the device. Also, in a pipeline, the necessary delay between the execution of two instructions to ensure that the values used by the second instruction are correct.

least significant bit (LSB): The lowest order bit in a word.

M

maskable interrupt: A hardware interrupt that can be enabled or disabled through software.

million instructions per second (MIPS): A measure of the execution speed of a computer.

most significant bit (MSB): The highest order bit in a word.

multichannel buffered serial port (McBSP): An on-chip full-duplex circuit that provides direct serial communication through several channels to external serial devices.

multiplexer: A device for selecting one of several available signals.

multiplier: A CPU component that multiplies the contents of two registers.

multivendor internet protocol: A standard network protocol supported by several major network communication vendors.

N

nonmaskable interrupt (NMI): An interrupt that can be neither masked nor disabled.

normalization: The reduction of a complex data structure to its simplest form or of a circuit to its lowest number of gates.

O

overflow: A condition in which the result of an arithmetic operation exceeds the capacity of the register used to hold that result.

P

packing: Minimizing the space occupied by data or memory through the elimination of discontinuous spaces between segments.

parallelism: Sequencing events to occur simultaneously. Parallelism is achieved in a CPU by using instruction pipelining.

peripheral: A device connected to and usually controlled by a host device.

pipeline: A method of executing instructions in which the output of one process serves as the input to another, much like an assembly line. These processes become the stages or phases of the pipeline.

pipeline processing: A technique that provides simultaneous, or parallel, processing within the computer. It refers to overlapping operations by moving data or instructions into a conceptual pipe with all stages of the pipe processing simultaneously.

phase-locked loop (PLL): A circuit for synchronizing a variable oscillator with the phase of the transmitted signal.

program cache: A fast memory cache for storing program instructions allowing for quick execution.

program fetch unit: The CPU hardware that retrieves program instructions.

program memory: A memory region used for storing and executing programs, separate from the region used for storing data.

R

random-access memory (RAM): A type of memory device in which the individual locations can be accessed in any order.

register: A small area of high speed memory, located within a processor or electronic device, that is used for temporarily storing data or instructions. Each register is given a name, contains a few bytes of information, and is referenced by programs.

reduced-instruction set computer (RISC): A computer whose instruction set and related decode mechanism are much simpler than those of microprogrammed complex instruction set computers. The result is a higher instruction throughput and a faster real-time interrupt service response from a smaller, cost-effective chip.

reset: A means of bringing the CPU to a known state by setting the registers and control bits to predetermined values and signaling execution to start at a specified address.

S

saturation: A state where any further input no longer results in the expected output.

synchronous burst static random-access memory (SBSRAM): RAM whose content does not have to be refreshed periodically. Transfer of data is at a fixed rate relative to the clock speed of the device.

synchronous dynamic random-access memory (SDRAM): RAM whose content is refreshed periodically or the data is lost. Transfer of data is at a fixed rate relative to the clock speed of the device.

shifter: A hardware unit that shifts bits in a word to the left or to the right.

T

T1: An American high-speed network communication service running at 1.544M bits per second and uses μ-law companding.

V

VelociTI: Architecture developed by TI that features very long instruction words.

VLIW: Very long instruction word.

W

word: A set of bits (32 bits for the ’C6x devices) that is stored, addressed, transmitted, or operated as a unit.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current and complete.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.