

# ***How to Begin Development Today with the High Performance Floating TMS320C67x DSP***

*C6000 Applications Team*

## **ABSTRACT**

This application report describes how you can begin development now for the Texas Instruments (TI™) TMS320C67x generation of high-performance digital signal processors (DSPs). Because of the compatibility between TMS320C6000 generation devices, existing C6000 software tools and development platforms can be used to develop code for the C67x and other future devices. This capability allows for systems to be up and running when silicon becomes available.

## **Contents**

<b>1</b>	<b>Introduction</b> .....	<b>1</b>
<b>2</b>	<b>VelociTI Architecture — The Common Link</b> .....	<b>4</b>
2.1	Similarities Between TMS320C62x and TMS320C67x .....	4
2.2	TMS320C67x Additional Features .....	4
2.2.1	L Unit Capabilities .....	4
2.2.2	S Unit Capabilities .....	5
2.2.3	M Unit Capabilities .....	5
2.2.4	D Unit Capabilities .....	5
2.3	Begin Development Today .....	5
<b>3</b>	<b>TMS320C67x Instruction Set is a Superset to TMS320C62x Instruction Set</b> .....	<b>6</b>
<b>4</b>	<b>Begin Writing Code for the C67x Today Using Existing C6x Tools</b> .....	<b>7</b>
4.1	C6x Tools Support C Code Using Floating-Point Arithmetic .....	8
4.2	C6x Literature Available .....	9
<b>5</b>	<b>TMS320C67x Instructions</b> .....	<b>9</b>

## **List of Figures**

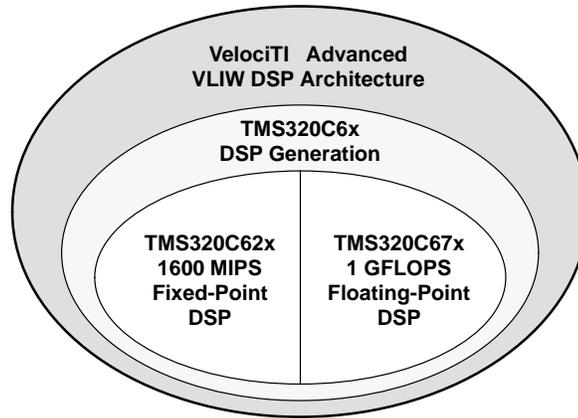
1	TMS320C6000 Generation of DSPs .....	2
2	TMS320C6000 Roadmap .....	2
3	TMS320C67x DSP Core Block Diagram .....	3
4	Differences Between TMS320C62x CPU and TMS320C67x CPU .....	3
5	TMS320C6000 Development Environment .....	5
6	Full Instruction Set for the C62x Fixed-Point and C67x Floating-Point DSPs .....	10

## **1 Introduction**

The Texas Instruments (TI™) TMS320C67x generation of high-performance digital signal processors (DSPs) now includes floating-point DSP capability with the introduction of the 1GFLOPS, TMS320C67x floating-point DSP core CPU technology (see Figure 1).

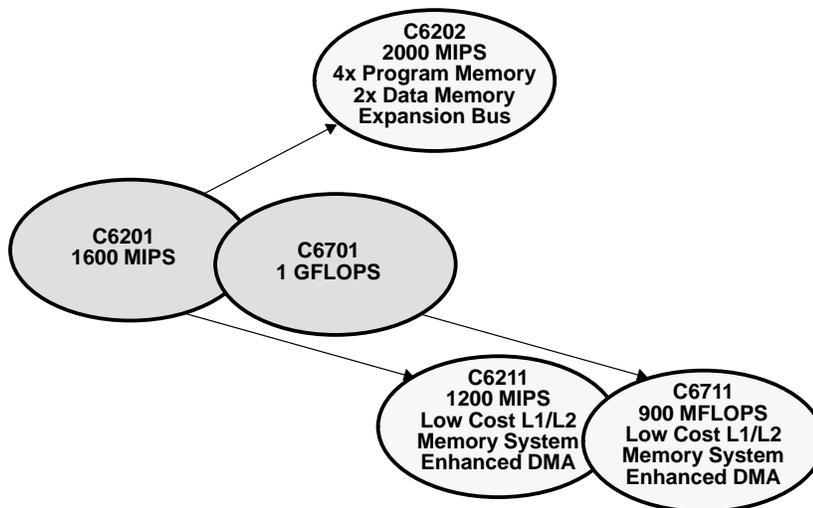
Trademarks are the property of their respective owners.

Introduced in February 1997, the C6000 generation is based on TI's VelociTI™ architecture, an advanced very long instruction word (VLIW) architecture for DSPs. The C6000 generation of DSPs now includes code-compatible fixed-point and floating-point DSP versions. Both share the distinction of respectively being the industry's highest performing fixed-point and floating-point DSPs.



**Figure 1. TMS320C6000 Generation of DSPs**

The C67x joins the fixed-point C62x to create the industry's first code-compatible fixed-point and floating-point DSPs. Figure 2 shows the roadmap for the C6000 platform. The C67x series of DSPs began sampling in the second half of 1998, with the first devices providing 1 GFLOPS of performance.



**Figure 2. TMS320C6000 Roadmap**

The C67x delivers 1 GFLOPS at 167 MHz (6-ns cycle time with six 32-bit floating-point instructions per cycle). Figure 3 shows a block diagram of the C67x CPU core.

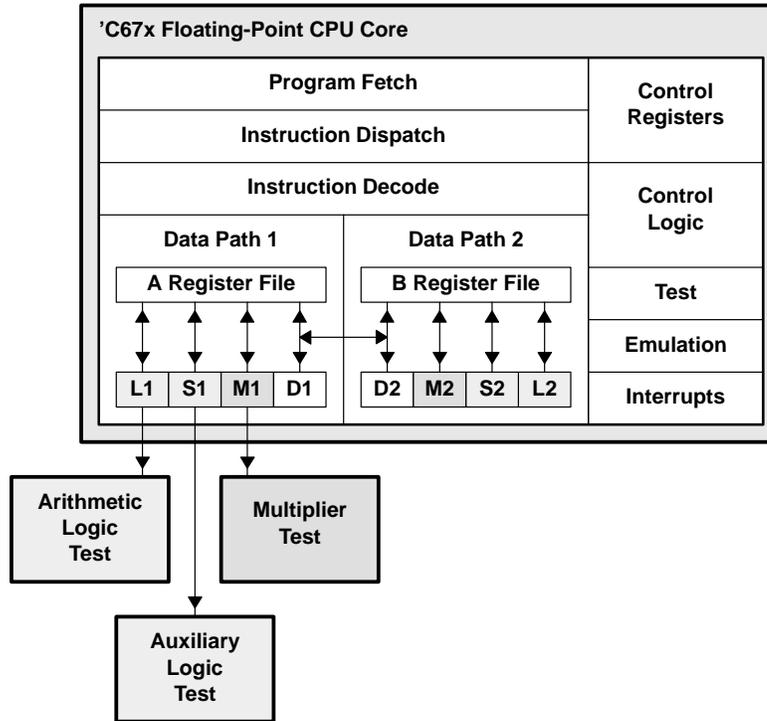


Figure 3. TMS320C67x DSP Core Block Diagram

The floating-point C67x CPU core differs from the fixed-point C62x CPU core only in the floating-point functionality of six of its eight functional units and the C67x wider bus structure. See Figure 4 for an illustration of these differences.

'C62x CPU			'C67x CPU		
M-Unit 1 Multiplier Unit	Control Register Emulation	M-Unit 1 Multiplier Unit	M-Unit 1 Multiplier Unit with Floating Point	Control Register Emulation	M-Unit 1 Multiplier Unit with Floating Point
D-Unit 1 Data Load/Store		D-Unit 1 Data Load/Store	D-Unit 1 Data Load/Store		D-Unit 1 Data Load/Store
S-Unit 1 Auxiliary Logic Unit		S-Unit 1 Auxiliary Logic Unit	S-Unit 1 Auxiliary Logic Unit with Floating Point		S-Unit 1 Auxiliary Logic Unit with Floating Point
L-Unit 1 Arithmetic Logic Unit	Decode	L-Unit 1 Arithmetic Logic Unit	L-Unit 1 Arithmetic Logic Unit with Floating Point	Decode	L-Unit 1 Arithmetic Logic Unit with Floating Point
Register File		Register File	Register File		Register File
Program Fetch and Dispatch			Program Fetch and Dispatch		

Figure 4. Differences Between TMS320C62x CPU and TMS320C67x CPU

## 2 VelociTI Architecture — The Common Link

The C62x and C67x members of the C6x generation of DSPs each combine the power of the VelociTI architecture with the efficiency of TI's revolutionary DSP development environment, to provide almost 10x increase over today's high performance DSPs. Both benefit from VelociTI's design as an ideal target for the optimizing tools, by making use of VelociTI's extensive parallelism and pipelining-which is scheduled by the development tools.

Just as importantly, this common architecture affords designers a high degree of hardware and software compatibility. Maintaining commonality between the C62x and C67x cores, and symmetry between their datapaths, ensures one learning curve – and shorter design time.

### 2.1 Similarities Between TMS320C62x and TMS320C67x

- Peak eight 32-bit instructions/cycle
- VelociTI advanced very-long instruction word (VLIW) architecture
- Eight independent functional units (two each of .L, .D, .S, and .M)
- Load/store architecture with 32 32-bit general-purpose registers
- 100% conditional instructions
- Dual endian support
- Byte addressable, 32-bit address range
- Fully pipelined branches, zero overhead-branching
- Non-interlocked pipeline with uniform latency for fetch

### 2.2 TMS320C67x Additional Features

- Peak 1336 MIPS at 167 MHz (6-ns cycle time)
- Hardware support for both single precision (32-bit) IEEE and double precision (64-bit) IEEE floating-point operations
- Peak 1 GFLOPS single precision at 167 MHz
- Peak 420 MFLOPS double precision at 167 MHz
- Roadmap to 3 GFLOPS
- Roadmap to sub-\$50 prices
- 24x24-bit and 32x32-bit integer multiply

VelociTI's highly orthogonal design enables *sustained* throughput of up to eight instructions in parallel *every cycle*. With two each of .L, .S, .D, and .M units, all instructions have *at least* two functional units to choose from – and a number can be executed on *up to six* functional units. Following are the capabilities of the functional units.

#### 2.2.1 L Unit Capabilities

- 32/40-bit fixed-point arithmetic and compare operations
- 32/64-bit floating-point arithmetic and compare operations (IEEE single and double precision)

- 32-bit fixed-point logical operations
- Fixed/floating point conversions
- 64 to 32-bit floating-point conversions

### 2.2.2 S Unit Capabilities

- 32-bit fixed-point arithmetic operation
- 32/40-bit shifts and 32-bit bit-field operation
- Branching and constant generation
- 32/64-bit floating-point reciprocal, absolute value, compares, and 1/sqrt operations
- 32 to 64-bit floating-point conversions

### 2.2.3 M Unit Capabilities

- 16x16-bit fixed-point multiplies
- 32x32-bit fixed-point multiplies
- 32x32-bit single-precision floating-point multiplies
- 64x64-bit single-precision floating-point multiplies

### 2.2.4 D Unit Capabilities

- 32-bit add, subtract, linear, and circular address calculation
- 8/16/32/64-bit loads
- 8/16/32-bit stores

## 2.3 Begin Development Today

The C62x and C67x devices share the same state-of-the-art development tools (see Figure 5). That means, with one architecture and one set of software-development tools, you only climb one learning curve to access both the C6000 floating-point and fixed-point performance goals.

Visit our web site at <http://www.ti.com/sc/c67x> for more information on the C67x.

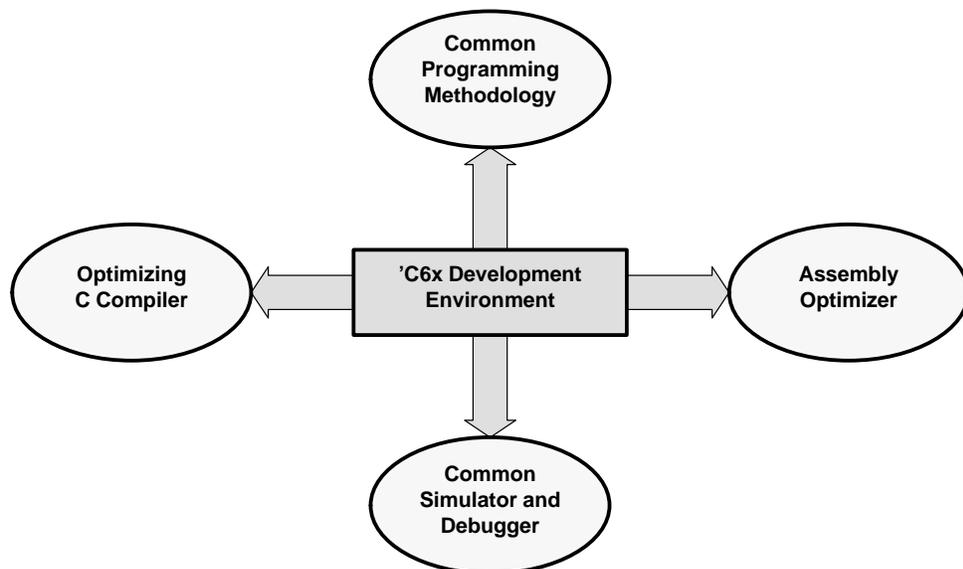


Figure 5. TMS320C6000 Development Environment

### 3 TMS320C67x Instruction Set is a Superset to TMS320C62x Instruction Set

The C67x is designed to ensure customers can begin development for their floating-point systems immediately. One key to that accessibility is the common foundation in the instruction sets. With only the addition of floating-point capabilities to six of the eight functional units, the C67x instruction set is a superset to the C62x instruction set. **All C62x instructions run unmodified on the C67x CPU core.** As a result, development can begin using the C62x instruction set currently supported in the C6x code generation tools.

The following is a list of the instructions that are unique to the C67x. (See the full set of C67x and C62x instructions on page 9.)

- .L Arithmetic Logic Unit
 

ADDSP	Single precision floating-point add
ADDDP	Double precision floating-point multiply
SUBSP	Single precision floating-point subtract
SUBDP	Double precision floating-point subtract
INTSP	Convert integer to single precision floating-point
INTDP	Convert integer to double precision floating-point
SPINT	Convert single precision floating-point to integer
DPINT	Convert double precision floating-point to integer
SPTRUNC	Convert single precision floating-point to integer with truncation
DPTRUNC	Convert double precision floating-point to integer with truncation
DPSP	Convert double precision floating-point to single precision floating-point
- .M Multiplier Unit
 

MPYSP	Single precision floating-point multiply
MPYDP	Double precision floating-point multiply
MPY24	24-bit x 24-bit integer multiply lower 32-bits of result out
MPY24H	24-bit x 24-bit integer multiply upper 32-bits of result out
MPYI	32-bit x 32-bit integer multiply lower 32-bits of result out
MPYID	32-bit x 32-bit integer multiply – 64-bits of result out
- .S Auxiliary Logic Unit
 

ABSSP	Single precision floating-point absolute value
ABSDP	Double precision floating-point absolute value
CMPGTSP	Single precision floating-point compare for greater than
CMPEQSP	Single precision floating-point compare for equality
CMPLTSP	Single precision floating-point compare for less than
CMPGTDP	Double precision floating-point compare for greater than
CMPEQDP	Double precision floating-point compare for equality
CMPLTDP	Double precision floating-point compare for less than

- |        |  |
|--------|--|
| RCPSP  | Single precision floating-point reciprocal approximation                   |
| RCPDP  | Double precision floating-point reciprocal approximation                   |
| RSQRSP | Single precision floating-point square-root reciprocal approximation       |
| RSQRDP | Double precision floating-point square-root reciprocal approximation       |
| SPDP   | Convert single precision floating-point to double precision floating-point |
- .D Data Load/Store Unit
 

LDDW	Load double Word (64-bit)
------	---------------------------

In one cycle, the C67x dispatches up to eight instructions – six of which can be floating-point. For example, the C67x executes two loads (8/16/32/64-bit), two single precision floating-point multiplies, two single precision floating-point adds, plus two other floating-point operations (for example, absolute value, compare, reciprocal approximation, or square-root reciprocal approximation). Single precision floating-point addition and multiplication operations require 3 delay slots for the result to be written to the register file, but a new instruction can be started on each cycle (single cycle throughput). Double precision floating-point operations require 6–9 delay slots for the result to be written to the register file. Double precision addition and multiplication instruction has 2- and 4- cycle throughput, respectively. Just like the C62x, the branch, load, and 16x16-bit integer multiplication instructions require 5, 4, and 1 delay slots, respectively, with single cycle throughput. All additional fixed-point instructions have zero delay slots and single cycle throughput.

## 4 Begin Writing Code for the C67x Today Using Existing C6x Tools

You can begin writing the fixed-point portions of your C67x applications today in C, linear assembly, or parallel assembly language. Applications that require floating-point support usually also use fixed-point (integer) instructions. The fixed code can be developed using the existing C6x tools that currently only support the fixed-point C62x. The C67x executes C62x COFF executable programs since it supports all of the C62x instructions.

C code written using floating-point arithmetic can be executed today on the C62x, since the C compiler offers a floating-point library that emulates floating-point arithmetic using the C62x fixed-point instruction set. While the floating-point code does not run at the same speed as it will on the C67x, it allows simulation of floating-point routines – getting you started with your C67x code development *today*.

The following C62x code example shows the use of fixed- and floating-point variables.

```

Float DotProduct (float *m, float *n, short count)
{
    short i;
    float product, sum = 0;
    for (I=0; I<count; I++
    {
        product = m[i] * n[i];
        sum +=product;
    }
    return sum
}
    
```

The core loop from a collision detection algorithm below is an example of C67x assembly code.

```

LOOP:
        LDDW      .D1      *A0++[2],A5:A4    ;load y0:x0 from memory
|| [B2]  ZERO     .D2      B0                ;if(retval) cntr = 0
||      SUBSP    .L1      A13,A2,A14        ;sub0 = sum1 - pnt
||      ADDSP    .L2      B9,B12,B13       ;sum3 = prod3 = sum2
|| [B0]  B        .S1      LOOP             ;if9cntr) branch to loop
||      CMPLTSP .S2      B15,B8,B1         ;if(abs1 < dB) if 1=1
||      MPYSP    .M1X     A5,B7,A10,        ;prod1 + y0 * p1
||      MPYSP    .M2      B4,B7,B10        ;prod4 =y1 *p1

|| [B0]  LDDW    .D1      *A0--[4],B5:B4    ;load z1:y1 from memory
||      SUB      .D2      B0,2,B0          ;
||      ADDSP    .L1      A9,A10,A12        ;sum0+prod0+prod1
||      SUBSP    .L2X     B13, A2, B14      ;sub1 +sum3-pnt
||      ABSSP    .S1      A14,A15          ;abs0 + abs (sub0)
|| [A1]  MVK     .S2      1, B2             ;if(if0) retval=1
||      MPYSP    .M1X     A6,B3,A11        ;prod2=z0*p2
||      MPYSP    .M2      B5,B3,B11        ;prod5=z1*p2

        LDDW    .D1      *A0++[5],A7:A6    ;loadx1:z0 from memory
|| [B1]  MV      .D2      B1,B2            ;if (if1) retval=if1=1
||      ADDSP    .L1      A12,A11,A13       ;sum1=sum0+prod2
||      ADDSP    .L2      B10,B11,B12      ;sum2=prod4+prod5
||      CMPLTSP .S1      A15,A8,A1         ;if(abs0,dA) if0=1
||      ABSSP    .S2      B14,B15          ;abs1=abs(sub1)
||      MPYSP    .M1X     A4,B6,A9         ;prod0=x0*p0
||      MPYSP    .M2X     A7,B6,B9         ;prod3=x1*p0

```

#### 4.1 C6x Tools Support C Code Using Floating-Point Arithmetic

C6x tools are available today to support the C62x and to begin your C67x code development. The following C6x development tools are available:

- XDS510 Emulator Hardware with JTAG Emulation cable
- TMD500510 PC Version
- XDS510 C6x C Source Debugger Software
- TMDX3240160-07 PC Version
- TMS320C6201 Test and Emulation Board (TEB)
- TMDX326106201
- C6x Optimizing C Compiler/Assembly
- Optimizer/Assembler/Linker
- TMDX3246855-07 PC Version
- TMDX3246555-07 SPARC Version
- C6x Simulator Software

- TMDX3246851-07 PC Version
- TMDX3246551-07 SPARC Version

## 4.2 C6x Literature Available

A great deal of literature is available today for the C6000 devices.

- TMS320C6000 CPU and Instruction Set Reference Guide
- TMS320C6000 Peripherals Reference Guide
- TMS320C6000 Technical Brief
- TMS320C6000 Programmer's Guide
- TMS320C6000 Evaluation Module Reference Guide
- TMS320C6000 Peripheral Support Library Programmer's Reference
- TMS320C6000 Assembly Language Tools User's Guide
- TMS320C6000 Optimizing C Compiler User's Guide
- TMS320C6000 C Source Debugger User's Guide
- TMS320C6000 C Source Debugger For SPARCstation's

Visit our web site at <http://www.ti.com/sc/docs/dsps/products/c6000/index.htm> for more information.

## 5 TMS320C67x Instructions

Figure 6 lists the full instruction set for the C62x fixed-point and C67x floating-point DSPs. Floating-point capabilities are achieved with the additional instructions indicated below.

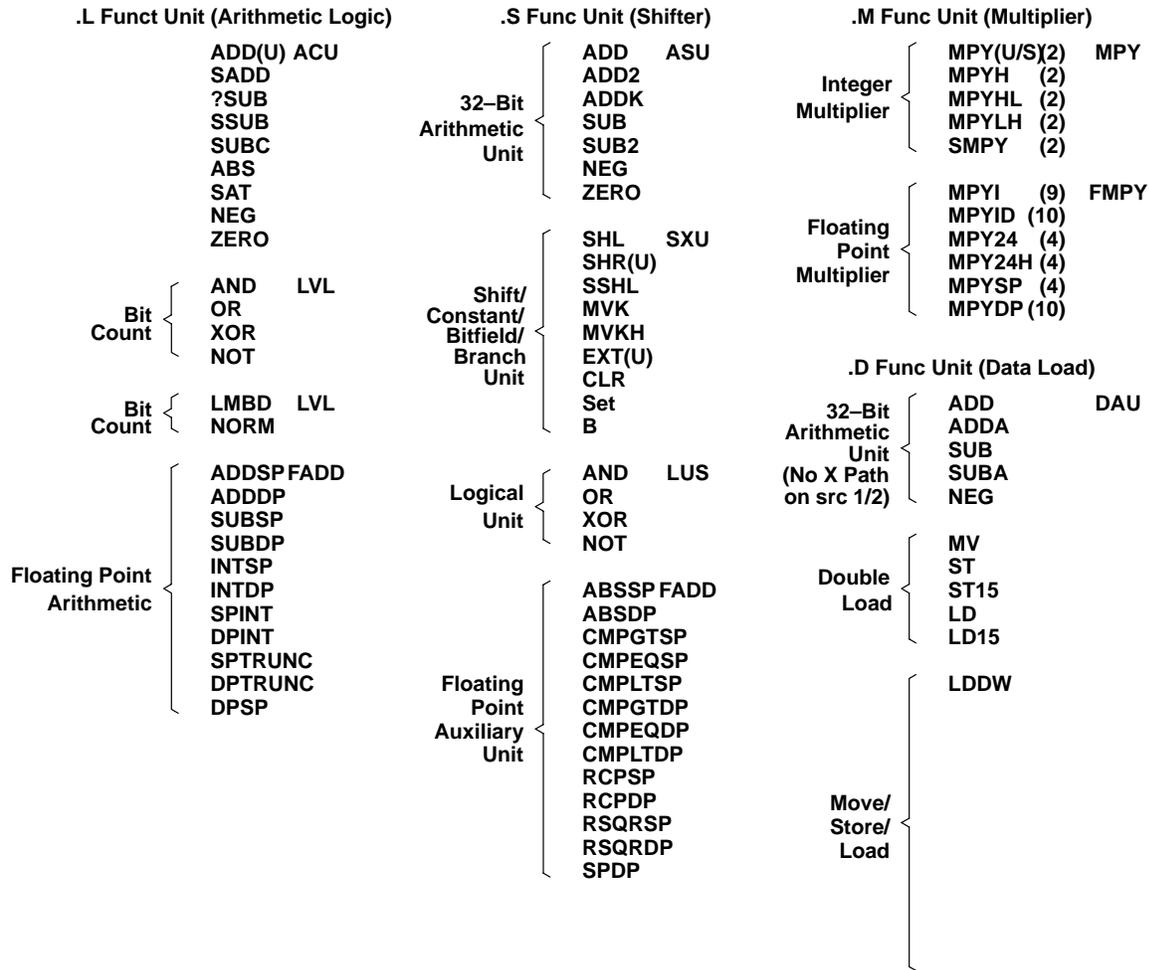


Figure 6. Full Instruction Set for the C62x Fixed-Point and C67x Floating-Point DSPs

## IMPORTANT NOTICE

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

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

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

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

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

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

### Mailing Address:

Texas Instruments  
Post Office Box 655303  
Dallas, Texas 75265