TMS320C54x™ DSP Functional Overview

Literature Number: SPRU307A
September 1998 – Revised May 2000

TMS320C54x DSP Functional Overview

This document provides a functional overview of the devices included in the Texas Instrument TMS320C54x™ generation of digital signal processors. Included are descriptions of the central processing unit (CPU) architecture, bus structure, memory structure, on-chip peripherals, and the instruction set. Detailed descriptions of device-specific characteristics such as package pinouts, package mechanical data, and device electrical characteristics are included in separate device-specific data sheets.

Topic **Page**

1.1 Features

Table 1–1 provides an overview the TMS320C54x, TMS320LC54x, and TMS320VC54x fixed-point, digital signal processor (DSP) families (hereafter referred to as the ’54x unless otherwise specified). The table shows significant features of each device, including the capacity of on-chip RAM and ROM, the available peripherals, the CPU speed, and the type of package with its total pin count.

Features provided by the ’54x DSPs include:

- High-performance, low-power ’C54x CPU
  - Advanced multibus architecture with three separate 16-bit data memory buses and one program memory bus
  - 40-bit arithmetic logic unit (ALU), including a 40-bit barrel shifter and two independent 40-bit accumulators
  - 17- $\times$ 17-bit parallel multiplier coupled to a 40-bit dedicated adder for nonpipelined single-cycle multiply/accumulate (MAC) operation
  - Compare, select, and store unit (CSSU) for the add/compare selection of the Viterbi operator
  - Exponent encoder to compute an exponent value of a 40-bit accumulator value in a single cycle
  - Two address generators with eight auxiliary registers and two auxiliary register arithmetic units (ARAUs)
  - Data buses with a bus holder feature
  - Extended addressing mode for up to 8M $\times$ 16-bit maximum addressable external program space
  - Single-instruction repeat and block-repeat operations for program code
  - Block-memory-move instructions for better program and data management
  - Instructions with a 32-bit-long word operand
  - Instructions with two- or three-operand reads
  - Arithmetic instructions with parallel store and parallel load
  - Conditional store instructions
  - Fast return from interrupt
- On-chip peripherals
  - Software-programmable wait-state generator and programmable bank-switching
  - Phase-locked loop (PLL) clock generator with internal crystal oscillator or external clock source
  - Full-duplex standard serial port

- Time-division multiplexed (TDM) serial port
- Buffered serial port (BSP)
- Multichannel buffered serial port (McBSP)
- Direct memory access (DMA) controller
- 8-bit parallel host-port interface (HPI)
- Enhanced 8-bit parallel host-port interface (HPI8)
- 16-bit parallel host-port interface (HPI16)
- 16-bit timer with 4-bit prescaler
- Interprocessor first-in first-out (FIFO) unit (on multiple CPU devices)

❑ Power conservation features

- Software power consumption control with IDLE1, IDLE2, and IDLE3 power-down modes
- Ability to disable external address bus, data bus, and control bus signals under software control
- Ability to disable CLKOUT under software control
- Low-voltage device options to reduce power consumption without compromising performance

❑ On-chip scan-based emulation capability

❑ IEEE 1149.1† (JTAG) boundary scan test capability

❑ 5.0-V power supply devices with speeds up to 40 million instructions per second (MIPS) (25-ns instruction cycle time)

❑ 3.3-V power supply devices with speeds up to 80 MIPS (12.5-ns instruction cycle time)

❑ 2.5-V power supply devices with speeds up to 100 MIPS (10-ns instruction cycle time)

❑ 1.8-V power supply devices with speeds up to 200 MIPS (10-ns instruction cycle time per CPU core)

❑ 1.5-V power supply devices with speeds up to 532 MIPS (7.5-ns instruction cycle time per CPU core)

† IEEE Standard 1149.1-1990 Standard Test-Access Port and Boundary Scan Architecture.

Table 1–1. Characteristics of the ’54x Processors

	C541	LC541	LC541B	C542	LC542	LC543	LC545A	LC546A	LC548	LC549	VC549	U/VC5402	U/VC5409	VC5410	VC5416	VC5420	VC5421	VC5441
Memory (On-chip)																		
RAM (K) (single access)									24	24	24			56	64	168	64	256
RAM (K) (dual access)	5	5	5	10	10	10	6	6	8	8	8	16	32	8	64	32	64	128
RAM (K) (two-way shared)																	128	256
ROM (K)	28	28	28	2	2	2	48	48	2	16	16	4	16	16	16		4	
Memory Security	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√			
Extended Addressing																		
In program space (>64K)									8M	8M	8M	1M	8M	8M	8M	256 K	512 K	
Peripherals																		
Standard serial ports	2	2	2				1	1										
TDM serial ports				1	1	1			1	1	1							
Buffered serial ports (BSPs)				1	1	1	1	1	2	2	2							
Multichannel buffered serial ports (McBSPs)												2	3	3	3	6	6	12
DMA controller												√	√	√	√	√	√	√
Standard 8-bit HPI				√	√		√		√	√	√							
Enhanced 8-bit HPI												√	√	√	√			
16-bit HPI													√		√	√	√	√
Interprocessor FIFO																√	√	
Timers	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	2	2	8
Hardware PLL	√	√		√	√	√												
Programmable PLL			√				√	√	√	√	√	√	√	√	√	√	√	√
General-Purpose I/O pins																		
BIO / XF	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	
Multiplexed with McBSP/HPI												20	26	21	26	36	36	36
Dedicated																8	8	16

Table 1–1. Characteristics of the ’54x Processors (Continued)

	C541	LC541	LC541B	C542	LC542	LC543	LC545A	LC546A	LC548	LC549	VC549	U/VC5402	U/VC5409	VC5410	VC5416	VC5420	VC5421	VC5441
CPU Cycle Time/Speed																		
25 ns (40 MHz–40 MIPS)	√	√	√	√	√	√	√	√										
20 ns (50 MHz–50 MIPS)		√			√	√	√	√	√									
15 ns (66 MHz–66 MIPS)			√				√	√	√	√								
12.5 ns (80 MHz–80 MIPS)										√		UC	UC					
10 ns (100 MHz–100 MIPS)											√	VC	VC	√				
10 ns (100 MHz–200 MIPS)																√	√	
7.5 ns (133 MHz–532 MIPS)																		√
6.25 ns (160 MHz–160 MIPS)															√			
Number of CPUs	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	4
Package Types																		
100-pin TQFP (PZ)	√	√	√			√		√										
128-pin TQFP (PBK)					√		√											
144-pin TQFP (PGE)				√	√				√	√	√	√	√	√	√	√	√	
144-pin BGA (GGU)									√	√	√	√	√		√	√	√	
169-pin BGA (GGU)																		√
176-pin BGA (GGW)														√				
Supply Voltage (IO/Core)																		
5.0 V / 5.0 V	√			√														
3.3 V / 3.3 V		√	√		√	√	√	√	√	√								
3.3 V / 2.5 V											√			√				
3.3 V / 1.8 V												VC	VC			√	√	
3.3 V / 1.5 V															√			√
1.8 – 3.3 V / 1.8 V												UC	UC					

Table 1–1. Characteristics of the ’54x Processors (Continued)

	C541	LC541	LC541B	C542	LC542	LC543	LC545A	LC546A	LC548	LC549	VC549	U/VC5402	U/VC5409	VC5410	VC5416	VC5420	VC5421	VC5441
Boundary Scan Support																		
Full	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√		√	√
Limited (observe only)																√		
Bootloader (On-chip)																		
	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	†	√	†

† The ’VC5420 and ’VC5441 have no on-chip ROM bootloader. The ’VC5420 and ’VC5441 are capable of being boot-loaded via the host-port interface (HPI) or the external memory interface (EMIF).

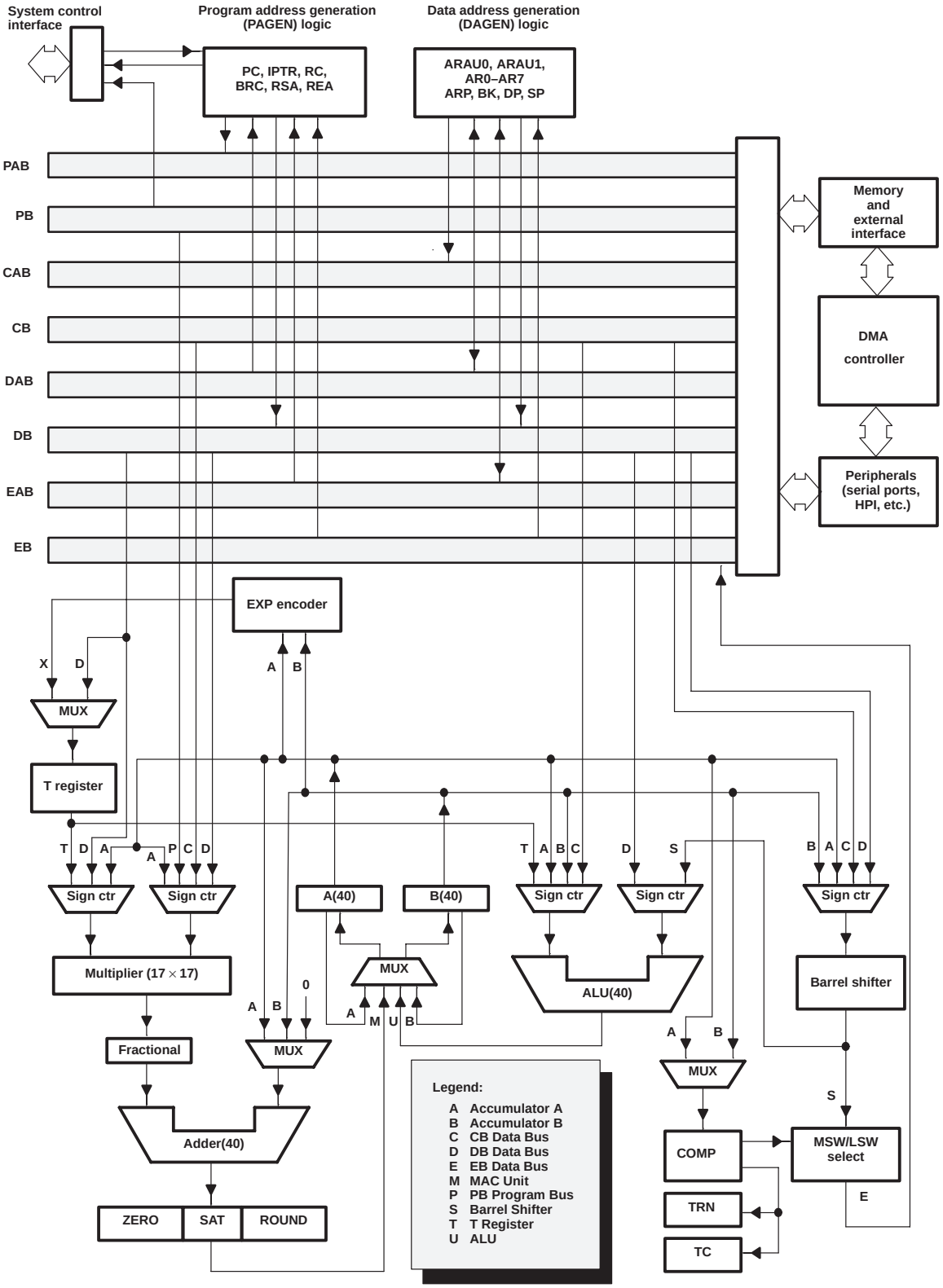
1.2 Architecture

The ’54x DSPs use an advanced, modified Harvard architecture that maximizes processing power by maintaining one program memory bus and three data memory buses. These processors also provide an arithmetic logic unit (ALU) that has a high degree of parallelism, application-specific hardware logic, on-chip memory, and additional on-chip peripherals. These DSP families also provide a highly specialized instruction set, which is the basis of the operational flexibility and speed of these DSPs.

Separate program and data spaces allow simultaneous access to program instructions and data, providing the high degree of parallelism. Two reads and one write operation can be performed in a single cycle. Instructions with parallel store and application-specific instructions can fully utilize this architecture. In addition, data can be transferred between data and program spaces. Such parallelism supports a powerful set of arithmetic, logic, and bit-manipulation operations that can all be performed in a single machine cycle. Also included are the control mechanisms to manage interrupts, repeated operations, and function calls.

Figure 1–1 is a functional block diagram that shows the principal blocks and bus structure in the ’54x devices.

Figure 1–1. Functional Block Diagram



1.2.1 Central Processing Unit (CPU)

The CPU of the ’54x devices contains:

- A 40-bit arithmetic logic unit (ALU)
- Two 40-bit accumulators
- A barrel shifter
- A 17 × 17-bit multiplier/adder
- A compare, select, and store unit (CSSU)

1.2.2 Arithmetic Logic Unit (ALU)

The ’54x devices perform 2s-complement arithmetic using a 40-bit ALU and two 40-bit accumulators (ACCA and ACCB). The ALU also can perform Boolean operations.

The ALU can function as two 16-bit ALUs and perform two 16-bit operations simultaneously when the C16 bit in status register 1 (ST1) is set.

1.2.3 Accumulators

The accumulators, ACCA and ACCB, store the output from the ALU or the multiplier / adder block; the accumulators can also provide a second input to the ALU or the multiplier / adder. The bits in each accumulator is grouped as follows:

- Guard bits (bits 32–39)
- A high-order word (bits 16–31)
- A low-order word (bits 0–15)

Instructions are provided for storing the guard bits, the high-order and the low-order accumulator words in data memory, and for manipulating 32-bit accumulator words in or out of data memory. Also, any of the accumulators can be used as temporary storage for the other.

1.2.4 Barrel Shifter

The ’54x’s barrel shifter has a 40-bit input connected to the accumulator or data memory (CB, DB) and a 40-bit output connected to the ALU or data memory (EB). The barrel shifter produces a left shift of 0 to 31 bits and a right shift of 0 to 16 bits on the input data. The shift requirements are defined in the shift-count field (ASM) of ST1 or defined in the temporary register (TREG), which is designated as a shift-count register. This shifter and the exponent detector normalize the values in an accumulator in a single cycle. The least significant bits (LSBs) of the output are filled with 0s and the most significant bits (MSBs) can be either zero-filled or sign-extended, depending on the state of the sign-extended mode bit (SXM) of ST1. Additional shift capabilities enable the processor to perform numerical scaling, bit extraction, extended arithmetic, and overflow prevention operations.

1.2.5 Multiplier/Adder

The multiplier/adder performs 17 × 17-bit 2s-complement multiplication with a 40-bit accumulation in a single instruction cycle. The multiplier/adder block consists of several elements: a multiplier, adder, signed/unsigned input control, fractional control, a zero detector, a rounder (2s-complement), overflow/saturation logic, and TREG. The multiplier has two inputs: one input is selected from the TREG, a data-memory operand, or an accumulator; the other is selected from the program memory, the data memory, an accumulator, or an immediate value. The fast on-chip multiplier allows the '54x to perform operations such as convolution, correlation, and filtering efficiently.

In addition, the multiplier and ALU together execute multiply/accumulate (MAC) computations and ALU operations in parallel in a single instruction cycle. This function is used in determining the Euclid distance, and in implementing symmetrical and least mean square (LMS) filters, which are required for complex DSP algorithms.

1.2.6 Compare, Select, and Store Unit (CSSU)

The compare, select, and store unit (CSSU) performs maximum comparisons between the accumulator's high and low words, allows the test/control (TC) flag bit of status register 0 (ST0) and the transition (TRN) register to keep their transition histories, and selects the larger word in the accumulator to be stored in data memory. The CSSU also accelerates Viterbi-type butterfly computation with optimized on-chip hardware.

1.2.7 Program Control

Program control is provided by several hardware and software mechanisms:

- The program controller decodes instructions, manages the pipeline, stores the status of operations, and decodes conditional operations. Some of the hardware elements included in the program controller are the program counter, the status and control register, the stack, and the address-generation logic.
- Some of the software mechanisms used for program control include branches, calls, conditional instructions, a repeat instruction, reset, and interrupts.
- The '54x supports both the use of hardware and software interrupts for program control. Interrupt service routines are vectored through a relocatable interrupt vector table. Interrupts can be globally enabled/disabled and can be individually masked through the interrupt mask register (IMR). Pending interrupts are indicated in the interrupt flag register (IFR). For detailed information on the structure of the interrupt vector table, the IMR and the IFR, see the device-specific data sheets.

1.2.8 Status Registers (ST0, ST1)

The status registers, ST0 and ST1, contain the status of the various conditions and modes for the ’54x devices. ST0 contains the flags (OV, C, and TC) produced by arithmetic operations and bit manipulations in addition to the data page pointer (DP) and the auxiliary register pointer (ARP) fields. ST1 contains the various modes and instructions that the processor operates on and executes.

1.2.9 Auxiliary Registers (AR0–AR7)

The eight 16-bit auxiliary registers (AR0–AR7) can be accessed by the central airthmetic logic unit (CALU) and modified by the auxiliary register arithmetic units (ARAUs). The primary function of the auxiliary registers is generating 16-bit addresses for data space. However, these registers also can act as general-purpose registers or counters.

1.2.10 Temporary Register (TREG)

The TREG is used to hold one of the multiplicands for multiply and multiply/accumulate instructions. It can hold a dynamic (execution-time programmable) shift count for instructions with a shift operation such as ADD, LD, and SUB. It also can hold a dynamic bit address for the BITT instruction. The EXP instruction stores the exponent value computed into the TREG, while the NORM instruction uses the TREG value to normalize the number. For ACS operation of Viterbi decoding, TREG holds branch metrics used by the DADST and DSADT instructions.

1.2.11 Transition Register (TRN)

The TRN is a 16-bit register that is used to hold the transition decision for the path to new metrics to perform the Viterbi algorithm. The CMPS (compare, select, max, and store) instruction updates the contents of the TRN based on the comparison between the accumulator high word and the accumulator low word.

1.2.12 Stack-Pointer Register (SP)

The SP is a 16-bit register that contains the address at the top of the system stack. The SP always points to the last element pushed onto the stack. The stack is manipulated by interrupts, traps, calls, returns, and the PUSHD, PSHM, POPD, and POPM instructions. Pushes and pops of the stack predecrement and postincrement, respectively, all 16 bits of the SP.

1.2.13 Circular-Buffer-Size Register (BK)

The 16-bit BK is used by the ARAUs in circular addressing to specify the data block size.

1.2.14 Block-Repeat Registers (BRC, RSA, REA)

The block-repeat counter (BRC) is a 16-bit register used to specify the number of times a block of code is to be repeated when performing a block repeat. The block-repeat start address (RSA) is a 16-bit register containing the starting address of the block of program memory to be repeated when operating in the repeat mode. The 16-bit block-repeat end address (REA) contains the ending address if the block of program memory is to be repeated when operating in the repeat mode.

1.2.15 Interrupt Registers (IMR, IFR)

The interrupt-mask register (IMR) is used to mask off specific interrupts individually at required times. The interrupt-flag register (IFR) indicates the current status of the interrupts.

1.2.16 Processor-Mode Status Register (PMST)

The processor-mode status register (PMST) controls memory configurations of the ’54x devices.

1.2.17 Power-Down Modes

There are three power-down modes, activated by the IDLE1, IDLE2, and IDLE3 instructions. In these modes, the ’54x devices enter a dormant state and dissipate considerably less power than in normal operation. The IDLE1 instruction is used to shut down the CPU. The IDLE2 instruction is used to shut down the CPU and on-chip peripherals. The IDLE3 instruction is used to shut down the ’54x processor completely. This instruction stops the PLL circuitry as well as the CPU and peripherals.

1.3 Bus Structure

The ’54x device architecture is built around eight major 16-bit buses:

- One program-read bus (PB) which carries the instruction code and immediate operands from program memory
- Two data-read buses (CB, DB) and one data-write bus (EB), which interconnect to various elements, such as the CPU, data-address generation logic (DAGEN), program-address generation logic (PAGEN), on-chip peripherals, and data memory
  - The CB and DB carry the operands read from data memory.
  - The EB carries the data to be written to memory.
- Four address buses (PAB, CAB, DAB, and EAB), which carry the addresses needed for instruction execution

The ’54x devices have the capability to generate up to two data-memory addresses per cycle, which are stored into two auxiliary register arithmetic units (ARAU0 and ARAU1).

The PB can carry data operands stored in program space (for instance, a coefficient table) to the multiplier for multiply/accumulate operations or to a destination in data space for the data-move instruction. This capability allows implementation of single-cycle three-operand instructions such as FIRS.

The ’54x devices also have an on-chip bidirectional bus for accessing on-chip peripherals; this bus is connected to DB and EB through the bus exchanger in the CPU interface. Accesses using this bus can require more than two cycles for reads and writes depending on the peripheral’s structure.

The ’54x devices can have bus holders connected to the data bus and the HPI data bus. Bus holders ensure that the data bus does not float. When bus holders are enabled, the data bus maintains its previous level. Setting bit 1 of the bank-switching control register (BSCR) enables bus holders and clearing bit 1 disables the bus holders. A reset automatically disables the bus holders.

Selected devices also have equivalent bus holders connected to the address bus. The bus holders ensure that the address bus does not float when in high impedance. For these devices, the bus holders are always enabled.

Table 1–2 summarizes the buses used by various types of accesses.

Table 1–2. Bus Usage for Accesses

Access Type	Address Bus				Program Bus		Data Bus	
	PAB	CAB	DAB	EAB	PB	CB	DB	EB
Program read	√				√			
Program write	√							√
Data single read			√				√	
Data dual read		√	√			√	√	
Data long (32-bit) read		√(hw)	√(lw)			√(hw)	√(lw)	
Data single write				√				√
Data read/data write			√	√			√	√
Dual read/coefficient read	√	√	√		√	√	√	
Peripheral read			√				√	
Peripheral write				√				√

Legend:
hw = high 16-bit word
lw = low 16-bit word

1.4 Memory

The minimum memory address range for the ’54x devices is 192K words — composed of 64K words in program space, 64K words in data space, and 64K words in I/O space. Selected devices also provide extended program memory space of up to 8M words. The program memory space contains the instructions to be executed as well as tables used in execution. The data memory space stores data used by the instructions. The I/O memory space interfaces to external memory-mapped peripherals and can also serve as extra data storage space.

The ’54x DSPs provide both on-chip RAM and ROM to improve system performance and integration.

Table 1–3 shows the on-chip memory options available on the ’54x family of devices.

Table 1–3. On-Chip Memory Options

	C541	LC541	LC541B	C542	LC542	LC543	LC545A	LC546A	LC548	LC549	VC549	U/VC5402	U/VC5409	VC5410	VC5416	VC5420	VC5421	VC5441
Memory (On-chip)																		
RAM (K) (single access)									24	24	24			56	64	168	64	256
RAM (K) (dual access)	5	5	5	10	10	10	6	6	8	8	8	16	32	8	64	32	64	128
RAM (K) (two-way shared)																	128	256
ROM (K)	28	28	28	2	2	2	48	48	2	16	16	4	16	16	16		4	
Memory Security	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√			

1.4.1 On-Chip ROM

The ’54x devices include on-chip maskable ROM that can be mapped into program memory or data memory depending on the device. On-chip ROM is mapped into program space by the microprocessor/microcontroller (MP/$\overline{\text{MC}}$) mode control pin. On-chip ROM that can be mapped into data space is controlled by the DROM bit in the processor mode status register (PMST). This allows an instruction to use data stored in the ROM as an operand.

Customers can arrange to have the ROM of the ’54x programmed with contents unique to any particular application.

1.4.2 Bootloader

A bootloader is available in the standard ’54x on-chip ROM. This bootloader can be used to transfer user code from an external source to anywhere in the program memory at power up automatically. If the MP/$\overline{\text{MC}}$ pin of the device is sampled low during a hardware reset, execution begins at location FF80h of the on-chip ROM. This location contains a branch instruction to the start of the bootloader program. The standard ’54x devices provide different ways to download the code to accommodate various system requirements:

- Parallel from 8-bit or 16-bit-wide EPROM
- Parallel from I/O space in 8-bit or 16-bit mode
- Serial port boot in 8-bit or 16-bit mode through the standard serial port, the TDM serial port, the buffered serial port (BSP), or the multichannel buffered serial port (McBSP)
- Host port interface boot (standard HPI, HPI8, and HPI16)
- Warm boot (restart of an application without reloading the code)

The bootloader options on each ’54x device are determined by the peripheral mix available on that device. See Table 1–1 for information on the peripherals available on each device.

On select devices, in addition to the bootloader, the standard on-chip ROM also contains complex FFT algorithms, μ-law/A-law expansion tables, and a sine look-up table.

1.4.3 On-Chip Dual-Access RAM (DARAM)

Dual-access RAM blocks can be accessed twice per machine cycle. This memory is intended primarily to store data values; however, it can be used to store program as well. At reset, the DARAM is mapped into data memory space. DARAM can be mapped into program/data memory space by setting the OVLY bit in the PMST register.

1.4.4 On-Chip Single-Access RAM (SARAM)

Each of the SARAM blocks is a single-access memory. This memory is intended primarily to store data values; however, it can be used to store program as well. SARAM can be mapped into program/data memory space by setting the OVLY bit in the PMST register.

1.4.5 On-Chip Two-Way Shared RAM

Select 54x devices with multiple CPU cores include two-way shared RAM blocks that allow simultaneous program space access from two CPU cores. Each CPU can perform a single access with zero-states to any location in the two-way shared RAM during each clock cycle.

This shared RAM is most efficiently used when the two CPUs are executing identical programs. In this case, the amount of program memory required for the application is effectively reduced by 50% since both CPUs can execute from the same RAM.

1.4.6 On-Chip Memory Security

A security feature is included on 54x devices to prevent the on-chip memory contents from being extracted by a user. This feature is enabled during the manufacturing process and is **ONLY** available to customers that order custom ROM programming. Consequently, memory security cannot be enabled/disabled by the user.

When the memory security feature is enabled, access to on-chip memory is protected in the following ways:

- **Emulation access:** The security feature completely disables the scan-based emulation capability of the ’54x to prevent the use of a debugger utility. Note that this only affects emulation, and does not prevent the use of the JTAG boundary scan test capability.
- **HPI access:** On select devices, HPI accesses are restricted when the security feature is enabled. These restrictions are described in Table 1–4.
- **CPU access:** The security feature prohibits the DSP CPU from accessing the on-chip memory. There are two levels of security associated with CPU accesses.
  - **ROM security option**. This option is the least secure, because it only protects the on-chip ROM and does not protect the on-chip RAM. When the ROM security option is enabled, any instruction fetched from external memory or on-chip RAM is prohibited from accessing on-chip ROM and reads invalid data (0FFFFh). Only instructions fetched from the on-chip ROM can be used to access the contents of the ROM.

- **ROM/RAM security option**. This option is the most secure, because it protects both the on-chip ROM and the on-chip RAM. When the ROM/RAM security option is enabled, any instruction fetched from external memory is prohibited from accessing on-chip ROM or RAM and reads invalid data (0FFFFh). Only instructions fetched from on-chip ROM or on-chip RAM can access the on-chip memory. The ROM/RAM security option also internally forces the device into microcomputer mode (MP/MC bit forced to zero), preventing the ROM from being disabled.

Table 1–4. On-Chip Memory Secruity HPI Access Restrictions

Device	HPI Writes	HPI Reads
C541, LC541, LC541B, C542, LC542. LC543, LC545A, LC546A, LC548, LC549, VC5402	Unrestricted	Unrestricted
VC5409	Unrestricted	Not allowed
VC5410	Unrestricted	Restricted to addresses (1000h to 17FFh)
VC5416	Unrestricted	Restricted to addresses (4000h to 5FFFh)

1.4.7 Program Memory

The standard external program memory space on the ’54x devices addresses up to 64K 16-bit words. Software can configure their memory cells to reside inside or outside of the program address map. When the cells are mapped into program space, the device automatically accesses them when their addresses are within bounds. When the program-address generation (PAGEN) logic generates an address outside its bounds, the device automatically generates an external access. The advantages of operating from on-chip memory are as follows:

- Higher performance because no wait states are required
- Lower cost than external memory
- Lower power than external memory

The advantage of operating from off-chip memory is the ability to access a larger address space.

1.4.7.1 Relocatable Interrupt Vector Table

The reset, interrupt, and trap vectors are addressed in program space. These vectors are soft — meaning that the processor, when taking the trap, loads the program counter (PC) with the trap address and executes the code at the vector location. Four words are reserved at each vector location to accommodate a delayed branch instruction: either two 1-word instructions or one 2-word instruction, which allows branching to the appropriate interrupt service routine with minimal overhead.

At device reset, the reset, interrupt, and trap vectors are mapped to address FF80h in program space. However, these vectors can be remapped to the

beginning of any 128-word page in program space after device reset. This is done by loading the interrupt vector pointer (IPTR) bits in the PMST register with the appropriate 128-word page boundary address. After loading IPTR, any user interrupt or trap vector is mapped to the new 128-word page.

NOTE: The hardware reset ($\overline{RS}$) vector cannot be remapped, because the hardware reset loads the IPTR with 1s. Therefore, the reset vector is always fetched at location FF80h in program space. In addition, for the ’54x, 128 words are reserved in the on-chip ROM for device-testing purposes. Application code written to be implemented in on-chip ROM must reserve these 128 words at addresses FF00h–FF7Fh in program space.

1.4.7.2 Extended Program Memory

Selected ’54x devices use a page-extended memory scheme in program space to allow access of up to 8M of program memory. This extended program memory is organized into a maximum of 128 pages (0–127), each 64K in length. Devices which implement the extended program memory scheme include the following additional features:

- Maximum of seven additional address lines (for a total of 23)
- An extra memory-mapped register [program counter extension register (XPC)]
- Six new instructions for addressing extended program memory space:
  - FB[D] — Far branch
  - FBACC[D] — Far branch to the location specified by the value in accumulator A or accumulator B
  - FCALA[D] — Far call to the location specified by the value in accumulator A or accumulator B
  - FCALL[D] — Far call
  - FRET[D] — Far return
  - FRETE[D] — Far return with interrupts enabled
- Two ’54x instructions are extended:
  - READA — Read program memory addressed by accumulator A and store in data memory
  - WRITA — Write data to program memory addressed by accumulator A

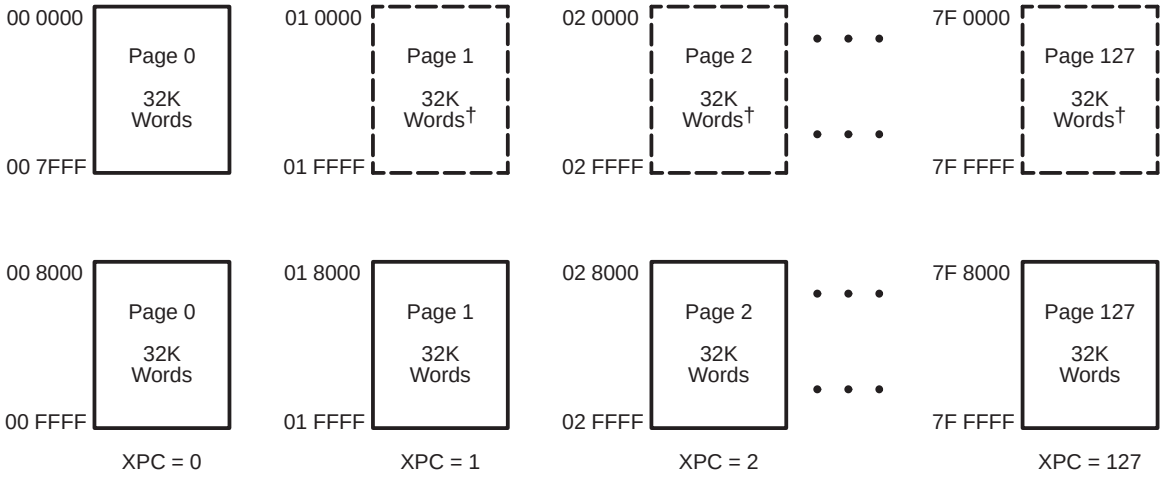
For more information on these six new instructions and the two extended instructions, refer to the instruction set summary (Table 1–11) and to the *TMS320C54x DSP Reference Set, Volume 2, Mnemonic Instruction Set*, literature number SPRU172. And for more information on extended program memory, refer to the *TMS320C54x DSP Reference Set, Volume 1, CPU and Peripherals*, literature number SPRU131.

Table 1–5 shows the extended program addressing options available in the ’54x family.

Table 1–5. Extended Program Memory Addressing Options

	C541	LC541	LC541B	C542	LC542	LC543	LC545A	LC546A	LC548	LC549	VC549	U/VC5402	U/VC5409	VC5410	VC5416	VC5420	VC5421	VC5441
Extended Addressing																		
In program space (>64K)									8M	8M	8M	1M	8M	8M	8M	256 K	512 K	

Figure 1–2. Extended Program Memory



† These pages are available when OVLY = 0, when on-chip RAM is not mapped in program space or data space. When OVLY = 1, the first 32K words are all on page 0 when on-chip RAM is mapped in program space or data space.

NOTE A: When the on-chip RAM is enabled in program space, all accesses to the region xx 0000 – xx 7FFF, regardless of page number, are mapped to the on-chip RAM at 00 0000 – 00 7FFF.

1.4.8 Data Memory

The data memory space on the ’54x device addresses 64K of 16-bit words. The device automatically accesses the on-chip RAM when addressing within its bounds. When an address is generated outside the RAM bounds, the device automatically generates an external access.

The advantages of operating from on-chip memory are as follows:

- Higher performance because no wait states are required
- Higher performance because of better flow within the pipeline of the CALU
- Lower cost than external memory
- Lower power than external memory

In addition to general-purpose data memory, the CPU maintains a set of memory-mapped registers in data memory for processor configuration and configuration/communication with the device peripherals. For detailed information on the implementation of the memory-mapped CPU and peripheral control registers, see the device-specific data sheets.

1.5 On-Chip Peripherals

All the ’54x devices have the same CPU structure; however, they have different on-chip peripherals connected to their CPUs. The on-chip peripheral options provided are:

- Software-programmable wait-state generator
- Programmable bank-switching
- Parallel I/O ports
- DMA controller
- Host-port interface (standard 8-bit, enhanced 8-bit, and 16-bit)
- Serial ports (standard, TDM, BSP, and McBSP)
- General-purpose I/O pins
- 16-bit timer with 4-bit prescaler
- Phase-locked loop (PLL) clock generator

1.5.1 Software-Programmable Wait-State Generators

The software-programmable wait-state generator can be used to extend external bus cycles to interface with slower off-chip memory and I/O devices. The software wait-state generator is incorporated without any external hardware. For off-chip memory access, a number of wait states can be specified for every 32K-word block of program and data memory space, and for one 64K-word block of I/O space within the software wait-state register (SWWSR). The software wait-state generator is programmable up to 7 or 14 wait states depending on the device. For more specific information on the software wait-state generation capability, see the device-specific data sheet.

1.5.2 Programmable Bank-Switching

Programmable bank-switching can be used to insert one cycle automatically when crossing memory-bank boundaries inside program memory or data memory space. One cycle can also be inserted when crossing from program-memory space to data-memory space (’54x) or from one program memory page to another program memory page on selected devices. This extra cycle allows memory devices to release the bus before other devices start driving the bus; thereby avoiding bus contention. The size of memory bank for the bank-switching is defined by the bank-switching control register (BSCR). For specific information on the bank-switching capabilities of a specific device, see the device-specific data sheet.

1.5.3 Parallel I/O Ports

Each ’54x device has a total of 64K I/O ports. These ports can be addressed by the PORTR instruction or the PORTW instruction. The $\overline{IS}$ signal indicates a read/write operation through an I/O port. The devices can interface easily with external devices through the I/O ports while requiring minimal off-chip address-decoding circuits.

1.5.4 Direct Memory Access (DMA) Controller

The ’54x direct memory access (DMA) controller transfers data between points in the memory map without intervention by the CPU. The DMA allows movements of data to and from internal program/data memory, internal peripherals (such as the McBSPs), or external memory devices to occur in the background of CPU operation. The DMA has six independent programmable channels, allowing six different contexts for DMA operation.

The DMA has the following features:

- The DMA operates independently of the CPU.
- The DMA has six channels. The DMA can keep track of the contexts of six independent block transfers.
- The DMA has higher priority than the CPU for both internal and external accesses.
- Each channel has independently programmable priorities.
- Each channel’s source and destination address registers can have configurable indexes through memory on each read and write transfer, respectively. The address may remain constant, postincrement, postdecrement, or be adjusted by a programmable value.
- Each read or write transfer may be initialized by selected events.
- On completion of a half-block or full-block transfer, each DMA channel may send an interrupt to the CPU.
- On-chip-RAM-to-off-chip-memory DMA transfer requires 5 cycles while off-chip-memory-to-on-chip-RAM DMA transfer requires 5 cycles.
- The DMA can perform double-word transfers (a 32-bit transfer of two 16-bit words).

1.5.4.1 DMA Memory Map

The DMA memory map includes access to on-chip memory on all devices and access to external memory on selected devices. The DMA memory map for on-chip memory is unaffected by the state of the memory control bits: MP/MC, DROM, and OVLY. For specific information on DMA implementations and memory maps, see the device-specific data sheets.

1.5.4.2 DMA Priority Level

Each DMA channel can be independently assigned high or low priority relative to each other. Multiple DMA channels that are assigned to the same priority level are handled in a round-robin manner.

1.5.4.3 DMA Source/Destination Address Modification

The DMA provides flexible address-indexing modes for easy implementation of data management schemes such as autobuffers and circular buffers. Source and destination addresses can be indexed separately and can be postincremented, postdecremented, or postincremented with a specified index offset.

1.5.4.4 DMA in Autoinitialization Mode

The DMA can automatically reinitialize itself after completion of a block transfer. Some of the DMA registers can be preloaded for the next block transfer through DMA global reload registers (DMGSA, DMGDA, and DMGCR). Autoinitialization allows:

- Continuous operation. Normally, the CPU would have to reinitialize the DMA immediately after the completion of the current block transfer; with the global reload registers, it can reinitialize these values for the next block transfer any time after the current block transfer begins.
- Repetitive operation. The CPU does not preload the global reload register with new values for each block transfer but only loads them on the first block transfer.

The DMA global reload register sets are sharred by all channels. However, select DMAs have been enhanced to expand the DMA global reload register set to provide each DMA channel its own DMA global reload register set. For example, the DMA global reload register set for channel 0 includes DMGSA0, DMGDA0, DMGCR0, and DMGFR0 while DMA channed 1 registers include DMGSA1, DMGDA1, DMGCR1, and DMGFR1.

Table 1–6. Devices Supporting Expanded DMA Global Reload Register Sets

	5402	5409	5410	5416	5420	5421	5441
Supported				√		√	√
Not supported	√	√	√		√		

1.5.4.5 DMA Transfer Counting

The DMA channel element count register (DMCTRx) and the frame count register (DMFRCx) contain bit fields that represent the number of frames and number of elements per frame to be transferred.

- Frame count. This 8-bit value defines the total number of frames in the block transfer. The maximum number of frames per block transfer is 128 (FRAME COUNT= 0ffh). The counter is decremented upon the last read transfer in a frame transfer. Once the last frame is transferred, the selected 8-bit counter is reloaded with the DMA global frame reload register (DMGFR) if the AUTOINIT is set to 1. A frame count of 0 (default value) means the block transfer contains a single frame.
- Element count. This 16-bit value defines the number of elements per frame. This counter is decremented after the read transfer of each

element. The maximum number of elements per frame is 65536 (DMCTRn = 0ffffh). In autoinitialization mode, once the last frame is transferred, the counter is reloaded with the DMA global count reload register (DMGCR).

1.5.4.6 DMA Transfer in Double-Word Mode

Double-word mode allows the DMA to transfer 32-bit words in any index mode. In double-word mode, two consecutive 16-bit transfers are initiated and the source and destination addresses are automatically updated following each transfer. In this mode, each 32-bit word is considered to be one element.

1.5.4.7 DMA Channel Index Registers

The particular DMA channel index register is selected by way of the SIND and DIND field in the DMA mode control register (DMMCRx). Unlike basic address adjustment, in conjunction with the frame index DMFRI0 and DMFRI1, the DMA allows different adjustment amount depending on whether or not the element transfer is the last in the current frame. The normal adjustment value (element index) is contained in the element index registers, DMIDX0 and DMIDX1. The adjustment value (frame index) for the end of the frame, is determined by the selected DMA frame index register, either DMFRI0 or DMFRI1.

The element index and the frame index affect address adjustment as follows:

- Element index. For all except the last transfer in the frame, element index determines the amount to be added to the DMA channel for the source/destination address register (DMSRCx/DMDSTx) as selected by the SIND/DIND bits.
- Frame index. If the transfer is the last in a frame, frame index is used for address adjustment as selected by the SIND/DIND bits. This occurs in both single-frame and multiframe transfer.

1.5.4.8 DMA Interrupts

The ability of the DMA to interrupt the CPU based on the status of the data transfer is configurable and is determined by the IMOD and DINM bits in the DMA channel mode control register (DMMCRn). The available modes are shown in Table 1–7.

Table 1–7. DMA Interrupts

Mode	DINM	IMOD	Interrupt
ABU (non-decrement)	1	0	At buffer full only
ABU (non-decrement)	1	1	At half buffer and buffer full
Multiframe	1	0	At block transfer complete (DMCTRn=DMSEFCn[7:0]=0)
Multiframe	1	1	At end of frame and end of block (DMCTRn=0)
Either	0	X	No interrupt generated
Either	0	X	No interrupt generated

1.5.5 Host-Port Interface (HPI)

1.5.5.1 Standard 8-Bit HPI

The host-port interface (HPI) is an 8-bit parallel port used to interface a host processor to the DSP device. Information is exchanged between the DSP device and the host processor through on-chip memory that is accessible by both the host and the DSP device. The DSP devices have access to the HPI control (HPIC) register and the host can address the HPI memory through the HPI address register (HPIA). Standard 8-bit HPI memory is a 2K-word DARAM block that resides at 1000h to 17FFh in data memory and can also be used as general-purpose on-chip data or program DARAM.

Data transfers of 16-bit words occur as two consecutive bytes with a dedicated pin (HBIL) indicating whether the high or low byte is being transmitted. Two control pins, HCNTL1 and HCNTL0, control host access to the HPIA, HPI data (with an optional automatic address increment), or the HPIC. The host can interrupt the DSP device by writing to HPIC. The DSP device can interrupt the host with a dedicated $\overline{\text{HINT}}$ pin that the host can acknowledge and clear.

The standard 8-bit HPI has two modes of operation, shared-access mode (SAM) and host-only mode (HOM). In SAM, the normal mode of operation, both the DSP device and the host can access HPI memory. In this mode, asynchronous host accesses are resynchronized internally and, in case of conflict, the host has access priority and the DSP device waits one cycle. The HOM capability allows the host to access HPI memory while the DSP device is in IDLE2 (all internal clocks stopped) or in reset mode. The host can therefore access the HPI RAM while the DSP device is in its optimal configuration in terms of power consumption.

The HPI control register has two data strobes, ($\overline{\text{HDS1}}$ and $\overline{\text{HDS2}}$), a read/write strobe (HR/$\overline{\text{W}}$), and an address strobe ($\overline{\text{HAS}}$) to enable a glueless interface to a variety of industry-standard host devices. The HPI is interfaced easily to hosts with multiplexed address/data bus, separate address and data buses, one data strobe and a read/write strobe, or two separate strobes for read and write.

The HPI supports high-speed back-to-back accesses.

- In the SAM, the HPI can transfer one byte every five DSP device periods—that is, 8 MBps with a 40-MIPS DSP, or 20 MBps with a 100-MIPS DSP. The HPI is designed so that the host can take advantage of this high bandwidth and run at frequencies up to $(f \times n) \div 5$, where n is the number of host cycles for an external access and f is the DSP device frequency.
- In HOM, the HPI supports high-speed back-to-back host accesses at one byte every 50 ns—that is, 20 MBps with a 40 MIPS or faster DSP.

1.5.5.2 Enhanced 8-Bit HPI (HPI8)

The enhanced 8-bit HPI (HPI8) is similar to the standard 8-bit HPI with the additional capability that data can be exchanged between the host processor and the DSP throughout the entire on-chip memory via the DMA.

The HPI8 can address all on-chip memory including extended memory pages. Extended memory addresses are defined by a 23-bit address. The HPI sets the upper six bits of the extended memory address by writing a one to the XHPIA bit in HPIC, then writing address bits A[22:16] into HPIA. The lower 16 bits of the extended memory address are set by writing a zero to XHPIA, followed by writing bits A[15:0] to HPIA. Similar to previous implementations of the HPI, after a write is performed to XHPIA or HPIA, a memory prefetch is initiated. The XHPIA bit is accessible only by the host.

The address and data strobe scheme implemented on the standard 8-bit HPI is also used on the enhanced 8-bit HPI.

All memory access on the HPI8 is in shared-access mode, meaning both the DSP and the host can access memory. Asynchronous host accesses are resynchronized internally and in the event that the CPU and the host both request access to the same memory block, the host has access priority. The HRDY pin provides handshaking to the host during memory access.

On the ’5410, the HPI8 also provides the capability to access memory during reset and power-down states. During reset, data or application code can be loaded via the HPI8 and the application can be initiated through the HPI option of the bootloader. During IDLE2/3 states, the HPI and the other six DMA channels continue to operate and all pending DMA events will complete before the DSP stops the clocks. The HPI has higher priority than the other six DMA channels. The HPI will continue to have access to memory in IDLE2/3 even after the DSP has stopped the internal clocks as long as X2/CLKIN is maintained. The HPI8 also remains active during emulation stop.

1.5.5.3 16-Bit HPI (HPI16)

The HPI16 is an enhanced 16-bit version of the ’C54x 8-bit HPI. The HPI16 is designed to allow a 16-bit host to access the DSP on-chip memory, with the host acting as the master of the interface. It should be noted that neither the CPU nor the DMA I/O spaces can be accessed using the HPI16.

Some features of the HPI16 include:

- 16-bit bidirectional data bus
- Multiple data strobes and control signals to allow glueless interfacing to a variety of hosts
- Multiplexed and nonmultiplexed address/data modes
- 18-bit address bus used in nonmultiplexed mode to allow access to all internal memory (including internal extended address pages)
- 18-bit address register used in multiplexed mode. Includes address autoincrement feature for faster accesses to sequential addresses.
- Interface to on-chip DMA module to allow access to entire internal memory space
- HRDY signal to hold off host accesses due to DMA latency
- Control register available in *multiplexed* mode only. Accessible by either host or DSP to provide host/DSP interrupts, extended addressing, and data prefetch capability.
- Maximum data rate: 28 MB/s at 100-MHz DSP clock rate (assuming multiplexed address/data with no DMA latency).

The HPI16 acts as a slave to a 16-bit host processor and allows access to the on-chip memory of the DSP. There are two modes of operation as determined by the HMODE signal: multiplexed mode and nonmultiplexed mode. In multiplexed mode, the HPI16 address and data buses are multiplexed onto a single bus. This provides a simple and glueless connection to multiplexed-bus processors. In nonmultiplexed mode, the HPI16 address and data buses are separate dedicated buses. In this mode, there is no access to the HPI16 control register (HPIC).

1.5.5.4 Combination Enhanced 8-Bit HPI (HPI8) and 16-Bit HPI (HPI16)

Some ’54x devices include both the HPI8 and the HPI16 (see Table 1–8). On these devices, only one HPI version can be used at a time and the selection is made by the HPI16 input pin. When the HPI16 pin is driven low, the HPI8 module is enabled; when the pin is driven high, the HPI16 module is enabled. These devices do not include an HMODE signal, so when the HPI16 module is enabled, only the nonmultiplexed mode of operation is supported.

Table 1–8. HPIs Supported

	C541	LC541	LC541B	C542	LC542	LC543	LC545A	LC546A	LC548	LC549	VC549	U/VC5402	U/VC5409	VC5410	VC5416	VC5420	VC5421	VC5441
Standard 8-bit HPI				√	√		√		√	√	√							
Enhanced 8-bit HPI												√	√	√	√			
16-bit HPI													√		√	√	√	√

1.5.6 Serial Ports

The ’54x devices provide high-speed, full-duplex serial ports that allow direct interface to other ’54x devices, codecs, and other devices in a system. There is a standard serial port, a time-division-multiplexed (TDM) serial port, a buffered serial port (BSP), and a multichannel buffered serial port (McBSP). Table 1–1 shows the availability of each of the serial port types in the ’54x family.

1.5.6.1 Standard Serial Port

The general-purpose serial port utilizes two memory-mapped registers for data transfer: the data-transmit register (DXR) and the data-receive register (DRR). Both of these registers can be accessed in the same manner as any other memory location. The transmit and receive sections of the serial port each have associated clocks, frame-synchronization pulses, and serial-shift registers; and serial data can be transferred either in bytes or in 16-bit words. Serial port receive and transmit operations can generate their own maskable transmit and receive interrupts (XINT and RINT), allowing serial-port transfers to be managed through software. The ’54x serial ports are double-buffered and fully static.

1.5.6.2 TDM Serial Port

The TDM port allows the device to communicate through time-division multiplexing with up to seven other ’54x devices with TDM ports. Time-division multiplexing is the division of time intervals into a number of subintervals with each subinterval representing a prespecified communications channel. The TDM port serially transmits 16-bit words on a single data line (TDAT) and destination addresses on a single address line (TADD). Each device can transmit data on a single channel and receive data from one or more of the eight channels, providing a simple and efficient interface for multiprocessing applications. A frame synchronization pulse occurs once every 128 clock cycles, corresponding to the transmission of one 16-bit word on each of the eight channels. Like the general-purpose serial port, the TDM port is double-buffered on both input and output data.

1.5.6.3 Buffered Serial Port (BSP)

The buffered serial port (BSP) consists of a full-duplex, double-buffered serial-port interface and an autobuffering unit (ABU). The serial port block of the BSP is an enhanced version of the standard serial port. The ABU allows the serial port to read/write directly to the ’54x internal memory using a dedicated bus independent of the CPU. This results in minimal overhead for serial port transactions and faster data rates.

When autobuffering capability is disabled (standard mode), serial port transfers are performed under software control through interrupts. In this mode, the ABU is transparent and the word-based interrupts (WXINT and WRINT) provided by the serial port are sent to the CPU as transmit interrupt (XINT) and receive interrupt (RINT). When autobuffering is enabled, word transfers are done directly between the serial port and the ’54x internal memory using ABU-embedded address generators.

The ABU has its own set of circular-addressing registers with corresponding address-generation units. Memory for the buffers resides in 2K words of the ’54x internal memory. The length and starting addresses of the buffers are user-programmable. A buffer-empty/buffer-full interrupt can be posted to the CPU. Buffering is easily halted by an autodisabling capability. Autobuffering capability can be enabled separately for transmit and receive sections. When autobuffering is disabled, operation is similar to that of the general-purpose serial port.

The BSP allows transfer of 8-, 10-, 12-, or 16-bit data packets. In burst mode, data packets are directed by a frame synchronization pulse for every packet. In continuous mode, the frame synchronization pulse occurs when the data transmission is initiated and no further pulses occur. The frame and clock strobes are frequency- and polarity-programmable. The BSP is fully static and operates at arbitrarily low clock frequencies. The BSP maximum operating frequency for ’54x devices up to 50 MIPS is CLKOUT. For higher-speed ’54x devices, the BSP maximum operating frequency is 50 Mbps at 20 ns.

Buffer Misalignment (BMINT) Interrupt (’549 only)

The BMINT interrupt is generated when a frame sync occurs and the ABU transmit or receive buffer pointer is not at the top of the buffer address. This is useful for detecting several potential error conditions on the serial interface, including extraneous and missed clocks, and frame sync pulses. A BMINT interrupt, therefore, indicates that one or more words may have been lost on the serial interface.

BMINT is useful for detecting buffer misalignment only when the buffer pointer(s) are initially loaded with the top-of-buffer address, and a frame of data contains the same number of words as the buffer length. These are the only conditions under which a frame sync occurring at a buffer address, other than the top of buffer, constitute an error condition. In cases where these conditions are met, a frame sync always occurs when the buffer pointer is at the top of buffer address, if the interface is functioning properly.

If BMINT is enabled under conditions other than those stated above, interrupts may be generated under circumstances other than actual buffer misalignment. In these cases, BMINT should generally be masked in the IMR register so that the processor will ignore this interrupt.

BMINT is available when operating autobuffering mode with continuous transfers, the FIG bit cleared to 0, and external serial clocks or frames.

The BSP0 and BSP1 BMINT bits in the IMR and IFR registers are bits 12 and 13, respectively (bit 15 is the MSB). The interrupt vector locations of IMR and IFR are 070h and 074h, respectively.

1.5.6.4 Multichannel Buffered Serial Port (McBSP)

The ’54x devices provide high-speed, full-duplex, multichannel buffered serial ports that allow direct interface to other ’54x devices, codecs, and other devices in a system. The multichannel buffered serial ports (McBSPs) are

based on the standard serial port interface found on other ’54x devices. Like its predecessors, the McBSP provides:

- Full-duplex communication
- Double-buffer data registers which allow a continuous data stream
- Independent framing and clocking for receive and transmit

In addition, the McBSP has the following capabilities:

- Direct interface to:
  - T1/E1 framers
  - MVIP switching-compatible and ST-BUS compliant devices
  - IOM-2 compliant devices
  - AC97-compliant devices
  - IIS-compliant devices
  - Serial peripheral interface
- Multichannel transmit and receive of up to 128 channels
- A wide selection of data sizes including 8, 12, 16, 20, 24, or 32 bits
- μ-law and A-law companding
- Programmable polarity for both frame synchronization and data clocks
- Programmable internal clock and frame generation

The McBSPs consist of separate transmit and receive channels that operate completely independently. The external interface of each McBSP consists of the following pins:

- BCLKX — Transmit reference clock
- BDX — Transmit data
- BFSX — Transmit frame sync
- BCLKR — Receive reference clock
- BDR — Receive data
- BFSR — Receive frame sync
- BCLKS — External clock reference for the programmable clock generator

The first six pins listed are identical to the previous serial port interfaces on the ’C5000 family of DSPs. The BCLKS pin is an additional signal to provide a clock reference to the McBSP programmable clock generator. As a compatibility option, BCLKS is not implemented on some packages.

On the transmitter, transmit frame synchronization and clocking are indicated by the BFSX and BCLKX pins, respectively. The CPU or DMA can initiate transmission of data by writing to the data transmit register (DXR). Data written to DXR is shifted out on the BDX pin through a transmit shift register (XSR). This structure allows DXR to be loaded with the next word to be sent while the transmission of the current word is in progress.

On the receiver, receive frame synchronization and clocking are indicated by the BFSR and BCLKR pins, respectively. The CPU or DMA can read received data from the data receive register (DRR). Data received on the BDR pin is shifted into a receive shift register (RSR) and then buffered in the receive buffer register (RBR). If DRR is empty, the RBR contents are copied into DRR.

If not, RBR holds the data until DRR is available. This structure allows storage of the two previous words while the reception of the current word is in progress.

To maintain pin compatibility with previous devices, not all 54x devices with McBSPs implement the BCLKS pin. For this reason, select 54x devices allow either the receive clock pin (BCLKR) or the transmit clock pin (BCLKX) to be configured as the input clock to the sample rate generator. This enhancement is enabled through two register bits: pin control register (PCR) bit 7 – enhanced sample clock mode (SCLKME), and sample rate generator register 2 (SRGR2) bit 13 – McBSP sample rate generator clock mode (CLKSM). SCLKME is an addition to the PCR contained in the McBSPs on previous ’C5000 devices. The selection of the sample rate generator (SRG) clock input source is made by the combination of the CLKSM and SCLKME bit values.

Table 1–9. External Clock Source For Sample Rate Generator Supported

	5402	5409	5410	5416	5420	5421	5441
CLKS pins			√				
BCLKX/BCLKR configurable				√		√	√
Not supported	√	√			√		

When either of the bidirectional pins, BCLKR or BCLKX, is configured as the clock input, its output buffer is automatically disabled. For example, with SCLKME = 1 and CLKSM = 0, the BCLKR pin is configured as the SRG input. In this case, both the transmitter and receiver circuits can be synchronized to the SRG output by setting the PCR bits (9:8) for CLKXM = 1 and CLKRM = 1. However, the SRG output is only driven onto the BCLKX pin because the BCLKR output is automatically disabled.

The CPU and DMA can move data to and from the McBSPs and can synchronize transfers based on McBSP interrupts, event signals, and status flags. The DMA is capable of handling data movement between the McBSPs and memory with no intervention from the CPU.

In addition to the standard serial port functions, the McBSP provides programmable clock and frame sync generation. Among the programmable functions are:

- Frame sync pulse width
- Frame period
- Frame sync delay
- Clock reference (internal vs. external)
- Clock division
- Clock and frame sync polarity

The on-chip companding hardware allows compression and expansion of data in either μ-law or A-law format. When companding is used, transmit data is encoded according to the specified companding law and received data is decoded to 2s complement format.

The McBSP allows multiple channels to be independently selected for the transmitter and receiver. When the multiple channels are selected, each frame represents a time-division multiplexed (TDM) data stream. In using TDM data

streams, the DSP CPU can be programmed to process as many data streams as necessary for the specific application. Thus, to save memory and bus bandwidth, multichannel selection allows independent enabling of particular channels for transmission and reception. Up to a maximum of 32 channels in a 128-channel bit stream can be enabled or disabled. Select devices have been enhanced to allow the enabling or disabling of up to 128 channels in a 128-channel bit stream. Devices supporting this enhancement are listed in Table 1–10.

Table 1–10. Devices Supporting Up To 128 Channels of TDM

	5402	5409	5410	5416	5420	5421	5441
Up to 128 channels supported				√		√	√
Up to 32 channels supported	√	√	√		√		

The clock-stop mode (CLKSTP) in the McBSP provides compatibility with the serial peripheral interface protocol. Clock-stop mode works with only single-phase frames and one word per frame. The word sizes supported by the McBSP are programmable for 8-, 12-, 16-, 20-, 24-, or 32-bit operation. When the McBSP is configured to operate in serial peripheral interface mode, both the transmitter and the receiver operate together as a master or as a slave.

The McBSP is fully static and operates at arbitrarily low clock frequencies. The maximum frequency is CPU clock frequency divided by 2.

1.5.7 General-Purpose I/O (GPIO) Pins

The ’54x family of devices provide general-purpose I/O pins that can be read or written through software control. All devices support two GPIO pins.

- BIO – A general input upon which conditional instructions can be based.
- XF – An external flag output that can be driven low or high under software control.

BIO and XF are often used for handshaking functions. In addition to the above described pins, other GPIO pins are available on selected devices. Some GPIO pins are multiplexed with the McBSP/HPI pin functions and some GPIO pins are dedicated. The multiplexed pins can be used for a GPIO function or a McBSP/HPI function under software control. However, the dedicated GPIO pins are always used for general-purpose I/O. See Table 1-1 for the availability of GPIO pins on each device.

1.5.8 Hardware Timer

The ’54x devices feature a 16-bit timing circuit with a four-bit prescaler. The timer counter is decremented by one at every CLKOUT cycle. Each time the counter decrements to zero, a timer interrupt is generated. The timer can be stopped, restarted, reset, or disabled by specific status bits.

1.5.9 Clock Generator

There are two basic options for clock generation on the ’54x family of devices: divide-by-two and PLL. In the first option, the CPU clock is generated by

dividing the input clock provided as X2/CLKIN by two. The second option uses a phase-locked loop circuit to generate a CPU clock that is a multiple of the frequency of the input clock. The PLL method allows a high-frequency internal CPU clock to be generated from a low-frequency external clock. Maintaining a low-frequency clock off chip reduces system power consumption, reduces clock-generated electromagnetic interference (EMI), and facilitates the use of less expensive external crystals or oscillators. The desired clock options are initially selected with the clock mode (CLKMD) pins.

The clock options available on the ’54x family vary depending on device. However, all ’54x devices provide the divide-by-two clock capability. On devices that provide a hardware PLL, the desired multiplication factor is chosen by the state of the clock mode pins only.

1.5.9.1 Hardware PLL

There are two types of hardware PLL providing different sets of multiplication factors. The option-one hardware PLL provides divide-by-two operation and multiplication factors of 1, 1.5, 2, or 3. The option-two hardware PLL provides divide-by-two operation and multiplication factors of 1, 4, 4.5, or 5.

1.5.9.2 Software PLL

The software PLL is programmable and the clock multiplication factor can be changed under software control. The initial clock mode setting is determined by the state of the clock mode pins and then the PLL can be programmed to change the clock mode. The software PLL provides multiplication factors ranging from 0.25 to 16. The software PLL also provides a built-in programmable lock-delay counter that prevents the PLL from clocking the CPU until a predefined lock-up time has elapsed, thereby ensuring that the PLL has had enough time to lock onto the input clock.

The input clock to the DSP is provided at the X2/CLKIN pin and can be produced by an external clock source (i.e., an integrated circuit oscillator), or can be produced by the on-chip oscillator on the DSP with as little as three external components: a crystal or ceramic resonator, and two resistors.

The PLL options available on each of the ’54x family DSPs are included in the peripherals section of Table 1–1. For more detailed information on the operation of the clock options and the hardware or software PLL, consult the *TMS320C54x DSP Reference Set, Volume 1: CPU and Peripherals* (literature number SPRU131).

1.6 Instruction Set Summary

This section summarizes the syntax used by the mnemonic assembler and the associated instruction set opcodes for the ’54x DSP devices (see Table 1–11). For detailed information on instruction operation, see the *TMS320C54x DSP Reference Set, Volume 2: Mnemonic Instruction Set* (literature number SPRU172); and for detailed information on the algebraic assembler, see the *TMS320C54x DSP Reference Set, Volume 3: Algebraic Instruction Set* (literature number SPRU179).

Table 1–11. ’54x Instruction Set Opcodes

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Arithmetic Instructions						
ABDST *Xmem*, *Ymem*	Absolute distance	1/1	1110	0011	XXXX	YYYY
ABS *src* [, *dst*]	Absolute value of ACC	1/1	1111	01SD	1000	0101
ADD *Smem, src*	Add operand to ACC	1/1	0000	000S	IAAA	AAAA
ADD *Smem,* **TS**, *src*	Add (shifted by TREG[5:0]) operand to ACC	1/1	0000	010S	IAAA	AAAA
ADD *Smem,* **16**, *src* [, *dst*]	Add (shifted by 16 bits) operand to ACC	1/1	0011	11SD	IAAA	AAAA
ADD *Smem* [, *SHIFT*], *src* [, *dst*]	Add shifted operand to ACC (2-word opcode)	2/2	0110 0000	1111 11SD	IAAA 000S	AAAA HIFT
ADD *Xmem, SHFT*, *src*	Add shifted operand to ACC	1/1	1001	000S	XXXX	SHFT
ADD *Xmem, Ymem, dst*	Add dual operands, shift result by 16	1/1	1010	000D	XXXX	YYYY
ADD *#lk* [, *SHFT*], *src* [, *dst*]	Add shifted long-immediate value to ACC	2/2	1111	00SD	0000	SHFT
ADD *#lk*, **16**, *src* [, *dst*]	Add (shifted by 16 bits) long-immediate to ACC	2/2	1111	00SD	0110	0000
ADD *src* [, *SHIFT*], [, *dst*]	Add ACC(s) (A/B), then shift result	1/1	1111	01SD	000S	HIFT
ADD *src,* **ASM** [, *dst*]	Add ACC(s) (A/B), then shift result by ASM value	1/1	1111	01SD	1000	0000
ADDC *Smem, src*	Add to accumulator with carry	1/1	0000	011S	IAAA	AAAA
ADDM *#lk*, *Smem*	Add long-immediate value to memory	2/2	0110	1011	IAAA	AAAA
ADDS *Smem, src*	Add to ACC with sign-extension suppressed	1/1	0000	001S	IAAA	AAAA
DADD *Lmem, src* [, *dst*]	Double/dual add to accumulator	1/1	0101	00SD	IAAA	AAAA
DADST *Lmem*, *dst*	Double/dual add/subtract of T, long operand	1/1	0101	101D	IAAA	AAAA
DELAY *Smem*	Memory delay	1/1	0100	1101	IAAA	AAAA
DRSUB *Lmem*, *src*	Double/dual 16-bit subtract from long word	1/1	0101	100S	IAAA	AAAA
DSADT *Lmem, dst*	Double/dual, subtract/add of T, long operand	1/1	0101	111D	IAAA	AAAA
DSUB *Lmem*, *src*	Double-precision/dual 16-bit subtract from ACC	1/1	0101	010S	IAAA	AAAA

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

Table 1–11. ’54x Instruction Set Opcodes (Continued)

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Arithmetic Instructions (Continued)						
DSUBT *Lmem*, *dst*	Double/dual, subtract/subtract of T, long operand	1/1	0101	110D	IAAA	AAAA
EXP *src*	Accumulator exponent	1/1	1111	010S	1000	1110
FIRS *Xmem*, *Ymem*, *pmad*	Symmetrical finite impulse response filter	2/3	1110	0000	XXXX	YYYY
LMS *Xmem*, *Ymem*	Least mean square	1/1	1110	0001	XXXX	YYYY
MAC[**R**] *Smem*, *src*	Multiply by TREG, add to ACC, round if specified	1/1	0010	10RS	IAAA	AAAA
MAC[**R**] *Xmem*, *Ymem*, *src* [, *dst*]	Multiply dual, add to ACC, round if specified	1/1	1011	0RSD	XXXX	YYYY
MAC *#lk*, *src* [, *dst*]	Multiply TREG by long-immediate, add to ACC	2/2	1111	00SD	0110	0111
MAC *Smem*, *#lk*, *src* [, *dst*]	Multiply by long-immediate value, add to ACC	2/2	0110	01SD	IAAA	AAAA
MACA[**R**] *Smem* [, *B*]	Multiply by ACCA, add to ACCB [round]	1/1	0011	01R1	IAAA	AAAA
MACA[**R**] **T**, *src* [, *dst*]	Multiply TREG by ACCA, add to ACC [round]	1/1	1111	01SD	1000	100R
MACD *Smem*, *pmad*, *src*	Multiply by program memory, accumulate/delay	2/3	0111	101S	IAAA	AAAA
MACP *Smem*, *pmad*, *src*	Multiply by program memory, then accumulate	2/3	0111	100S	IAAA	AAAA
MACSU *Xmem*, *Ymem*, *src*	Multiply signed by unsigned, then accumulate	1/1	1010	011S	XXXX	YYYY
MAS[**R**] *Smem, src*	Multiply by T, subtract from ACC [round]	1/1	0010	11RS	IAAA	AAAA
MAS[**R**] *Xmem*, *Ymem*, *src* [, *dst*]	Multiply dual, subtract from ACC [round]	1/1	1011	1RSD	XXXX	YYYY
MASA *Smem* [, *B*]	Multiply operand by ACCA, subtract from ACCB	1/1	0011	0011	IAAA	AAAA
MASA[**R**] **T**, *src* [, *dst*]	Multiply ACCA by T, subtract from ACC [round]	1/1	1111	01SD	1000	101R
MAX *dst*	Accumulator maximum	1/1	1111	010D	1000	0110
MIN *dst*	Accumulator minimum	1/1	1111	010D	1000	0111

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

Table 1–11. '54x Instruction Set Opcodes (Continued)

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Arithmetic Instructions (Continued)						
MPY[**R**] *Smem*, *dst*	Multiply TREG by operand, round if specified	1/1	0010	00RD	IAAA	AAAA
MPY *Xmem*, *Ymem*, *dst*	Multiply dual data-memory operands	1/1	1010	010D	XXXX	YYYY
MPY *Smem*, #*lk*, *dst*	Multiply operand by long-immediate operand	2/2	0110	001D	IAAA	AAAA
MPY #*lk*, *dst*	Multiply TREG value by long-immediate operand	2/2	1111	000D	0110	0110
MPYA *Smem*	Multiply single data-memory operand by ACCA	1/1	0011	0001	IAAA	AAAA
MPYA *dst*	Multiply TREG value by ACCA	1/1	1111	010D	1000	1100
MPYU *Smem*, *dst*	Multiply unsigned	1/1	0010	010D	IAAA	AAAA
NEG *src* [, *dst*]	Negate accumulator	1/1	1111	01SD	1000	0100
NORM *src* [, *dst*]	Normalize	1/1	1111	01SD	1000	1111
POLY *Smem*	Evaluate polynomial	1/1	0011	0110	IAAA	AAAA
RND *src* [*, dst*]	Round accumulator	1/1	1111	01SD	1001	1111
SAT *src*	Saturate accumulator	1/1	1111	010S	1000	0011
SQDST *Xmem*, *Ymem*	Square distance	1/1	1110	0010	XXXX	YYYY
SQUR *Smem, dst*	Square single data-memory operand	1/1	0010	011D	IAAA	AAAA
SQUR A, *dst*	Square ACCA high	1/1	1111	010D	1000	1101
SQURA *Smem, src*	Square and accumulate	1/1	0011	100S	IAAA	AAAA
SQURS *Smem, src*	Square and subtract	1/1	0011	101S	IAAA	AAAA
SUB *Smem, src*	Subtract operand from accumulator	1/1	0000	100S	IAAA	AAAA
SUB *Smem,* **TS**, *src*	Shift by TREG[5:0], then subtract from ACC	1/1	0000	110S	IAAA	AAAA
SUB *Smem,* **16**, *src* [, *dst*]	Shift operand 16 bits, then subtract from ACC	1/1	0100	00SD	IAAA	AAAA
SUB *Smem* [, *SHIFT*], *src* [, *dst*]	Shift operand, then subtract from ACC (2-word opcode)	2/2	0110 0000	1111 11SD	IAAA 001S	AAAA HIFT
SUB *Xmem, SHFT, src*	Shift operand, then subtract from ACC	1/1	1001	001S	XXXX	SHFT
SUB *Xmem, Ymem, dst*	Shift dual operands by 16, then subtract	1/1	1010	001D	XXXX	YYYY

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

Table 1–11. ’54x Instruction Set Opcodes (Continued)

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Arithmetic Instructions (Continued)						
SUB *#lk* [, *SHFT*], *src* [, *dst*]	Shift long-immediate, then subtract from ACC	2/2	1111	00SD	0001	SHFT
SUB *#lk,* **16**, *src* [, *dst*]	Shift long-immediate 16 bits, subtract from ACC	2/2	1111	00SD	0110	0001
SUB *src* [, *SHIFT*], [, *dst*]	Subtract shifted ACC from ACC	1/1	1111	01SD	001S	HIFT
SUB *src,* **ASM** [, *dst*]	Subtract ACC shifted by ASM from ACC	1/1	1111	01SD	1000	0001
SUBB *Smem, src*	Subtract from accumulator with borrow	1/1	0000	111D	IAAA	AAAA
SUBC *Smem, src*	Subtract conditionally	1/1	0001	111S	IAAA	AAAA
SUBS *Smem, src*	Subtract from ACC, sign-extension suppressed	1/1	0000	101S	IAAA	AAAA
Control Instructions						
B[**D**] *pmad*	Branch unconditionally with optional delay	2/4,2‡	1111	00Z0	0111	0011
BACC[**D**] *src*	Branch to address in ACC, optional delay	1/6,4‡	1111	01ZS	1110	0010
BANZ[**D**] *pmad*, *Sind*	Branch on AR(ARP) not zero, optional delay	2/4§,2¶,2‡	0110	11Z0	IAAA	AAAA
BC[**D**] *pmad*, *cond* [, *cond* [, *cond*]]	Branch conditionally, optional delay	2/5§,3¶,3‡	1111	10Z0	CCCC	CCCC
CALA[**D**] *src*	Call subroutine at address in ACC, optional delay	1/6,4‡	1111	01ZS	1110	0011
CALL[**D**] *pmad*	Call unconditionally, optional delay	2/4,2¶	1111	00Z0	0111	0100
CC[**D**] *pmad*, *cond* [, *cond* [, *cond*]]	Call conditionally, optional delay	2/5§,3¶,3‡	1111	10Z1	CCCC	CCCC
FB[**D**] *extpmad*	Far branch unconditionally (optional delay)	2/4,2‡	1111	10Z0	1KKK	KKKK
FBACC[**D**] *src*	Far branch to address in ACC, optional delay	1/6,4‡	1111	01ZS	1110	0110
FCALA[**D**] *src*	Far call to address in ACC, optional delay	1/6,4‡	1111	01ZS	1110	0111
FCALL[**D**] *extpmad*	Far call unconditionally, optional delay	2/4,2‡	1111	10Z1	1KKK	KKKK
FRAME *K*	Stack pointer immediate offset	1/1	1110	1110	KKKK	KKKK
FRET[**D**]	Far return (FRETD is for delayed return)	1/6,4‡	1111	01Z0	1110	0100

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

Table 1–11. ’54x Instruction Set Opcodes (Continued)

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Control Instructions (Continued)						
FRETE[**D**]	Far return, enable interrupts, optional delay	1/6,4‡	1111	01Z0	1110	0101
IDLE *K*	Idle until interrupt	1/4	1111	01NN	1110	0001
INTR *K*	Software interrupt	1/3	1111	0111	110K	KKKK
MAR *Smem*	Modify auxiliary register	1/1	0110	1101	IAAA	AAAA
NOP	No operation	1/1	1111	0100	1001	0101
POPD *Smem*	Pop top of stack to data memory	1/1	1000	1011	IAAA	AAAA
POPM *MMR*	Pop top of stack to memory-mapped register	1/1	1000	1010	IAAA	AAAA
PSHD *Smem*	Push data-memory value onto stack	1/1	0100	1011	IAAA	AAAA
PSHM *MMR*	Push memory-mapped register onto stack	1/1	0100	1010	IAAA	AAAA
RC[**D**] *cond* [, *cond* [, *cond*]]	Return conditionally, optional delay	1/5§,3¶,3‡	1111	11Z0	CCCC	CCCC
RESET	Software reset	1/3	1111	0111	1110	0000
RET[**D**]	Return, optional delay	1/5,3‡	1111	11Z0	0000	0000
RETE[**D**]	Return and enable interrupts, optional delay	1/5,3‡	1111	01Z0	1110	1011
RETF[**D**]	Return fast and enable interrupts, optional delay	1/3,1‡	1111	01Z0	1001	1011
RPT *Smem*	Repeat next instruction, count is in operand	1/1	0100	0111	IAAA	AAAA
RPT *#K*	Repeat next instruction, count is short immediate	1/1	1110	1100	KKKK	KKKK
RPT *#lk*	Repeat next instruction, count is long immediate	2/2	1111	0000	0111	0000
RPTB[**D**] *pmad*	Block repeat, optional delay	2/4,2‡	1111	00Z0	0111	0010
RPTZ *dst, #lk*	Repeat next instruction and clear accumulator	2/2	1111	000D	0111	0001
RSBX *N, SBIT*	Reset status-register bit	1/1	1111	01N0	1011	SBIT
SSBX *N, SBIT*	Set status-register bit	1/1	1111	01N1	1011	SBIT
TRAP *K*	Software interrupt	1/3	1111	0100	110K	KKKK

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

Table 1–11. ’54x Instruction Set Opcodes (Continued)

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Control Instructions (Continued)						
XC *n*, *cond* [, *cond* [, *cond*]]	Execute conditionally	1/1	1111	11N1	CCCC	CCCC
I/O Instructions						
PORTR *PA*, *Smem*	Read data from port	2/2	0111	0100	IAAA	AAAA
PORTW *Smem, PA*	Write data to port	2/2	0111	0101	IAAA	AAAA
Load/Store Instructions						
CMPS *src*, *Smem*	Compare, select and store maximum	1/1	1000	111S	IAAA	AAAA
DLD *Lmem*, *dst*	Long-word load to accumulator	1/1	0101	011D	IAAA	AAAA
DST *src*, *Lmem*	Store accumulator in long word	1/2	0100	111S	IAAA	AAAA
LD *Smem*, *dst*	Load accumulator with operand	1/1	0001	000D	IAAA	AAAA
LD *Smem*, **TS**, *dst*	Shift operand by TREG[5:0], then load into ACC	1/1	0001	010D	IAAA	AAAA
LD *Smem,* **16**, *dst*	Shift operand by 16 bits, then load into ACC	1/1	0100	010D	IAAA	AAAA
LD *Smem* [, *SHIFT*], *dst*	Shift operand, then load into ACC (2-word opcode)	2/2	0110 0000	1111 110D	IAAA 010S	AAAA HIFT
LD *Xmem*, *SHFT*, *dst*	Shift operand, then load into ACC	1/1	1001	010D	XXXX	SHFT
LD *#K, dst*	Load ACC with short-immediate operand	1/1	1110	100D	KKKK	KKKK
LD *#lk* [, *SHFT*], *dst*	Shift long-immediate, then load into ACC	2/2	1111	000D	0010	SHFT
LD *#lk*, **16**, *dst*	Shift long-immediate 16 bits, load into ACC	2/2	1111	000D	0110	0010
LD *src*, **ASM** [, *dst*]	Shift ACC by value in ASM register	1/1	1111	01SD	1000	0010
LD *src* [, *SHIFT*] [, *dst*]	Shift accumulator	1/1	1111	01SD	010S	HIFT
LD *Smem,* **T**	Load TREG with single data-memory operand	1/1	0011	0000	IAAA	AAAA
LD *Smem,* **DP**	Load DP with single data-memory operand	1/3	0100	0110	IAAA	AAAA
LD *#k9,* **DP**	Load DP with 9-bit operand	1/1	1110	101K	KKKK	KKKK
LD *#k5,* **ASM**	Load ACC shift-mode register with 5-bit operand	1/1	1110	1101	000K	KKKK

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

Table 1–11. '54x Instruction Set Opcodes (Continued)

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Load/Store Instructions (Continued)						
LD *#k3,* **ARP**	Load ARP with 3-bit operand	1/1	1111	0100	1010	0KKK
LD *Smem,* **ASM**	Load operand bits 4–0 into ASM register	1/1	0011	0010	IAAA	AAAA
LD *Xmem*, *dst* **|| MAC**[**R**] *Ymem* [, *dst_*]	Parallel load, multiply/accumulate [round]	1/1	1010	10RD	XXXX	YYYY
LD *Xmem*, *dst* **|| MAS**[**R**] *Ymem* [, *dst_*]	Parallel load, multiply/subtract [round]	1/1	1010	11RD	XXXX	YYYY
LDM *MMR, dst*	Load memory-mapped register	1/1	0100	100D	IAAA	AAAA
LDR *Smem*, *dst*	Load memory value in ACC high with rounding	1/1	0001	011D	IAAA	AAAA
LDU *Smem*, *dst*	Load unsigned memory value	1/1	0001	001D	IAAA	AAAA
LTD *Smem*	Load TREG and insert delay	1/1	0100	1100	IAAA	AAAA
SACCD *src, Xmem, cond*	Store accumulator conditionally	1/1	1001	111S	XXXX	COND
SRCCD *Xmem*, *cond*	Store block-repeat counter conditionally	1/1	1001	1101	XXXX	COND
ST T, *Smem*	Store TREG	1/1	1000	1100	IAAA	AAAA
ST TRN, *Smem*	Store TRN	1/1	1000	1101	IAAA	AAAA
ST *#lk*, *Smem*	Store long-immediate operand	2/2	0111	0110	IAAA	AAAA
STH *src*, *Smem*	Store accumulator high to data memory	1/1	1000	001S	IAAA	AAAA
STH *src*, **ASM**, *Smem*	Shift ACC high by ASM, store to data memory	1/1	1000	011S	IAAA	AAAA
STH *src*, *SHFT*, *Xmem*	Shift ACC high, then store to data memory	1/1	1001	101S	XXXX	SHFT
STH *src* [, *SHIFT*], *Smem*	Shift ACC high, then store to data memory (2-word opcode)	2/2	0110 0000	1111 110S	IAAA 011S	AAAA HIFT
ST *src*, *Ymem* **|| ADD** *Xmem*, *dst*	Store ACC with parallel add	1/1	1100	00SD	XXXX	YYYY
ST *src*, *Ymem* **|| LD** *Xmem*, *dst*	Store ACC with parallel load into accumulator	1/1	1100	10SD	XXXX	YYYY
ST *src*, *Ymem* **|| LD** *Xmem*, **T**	Store ACC with parallel load into TREG	1/1	1110	01S0	XXXX	YYYY
ST *src*, *Ymem* **|| MAC**[**R**] *Xmem*, *dst*	Parallel store and multiply ACC [round]	1/1	1101	0RSD	XXXX	YYYY

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

Table 1–11. ’54x Instruction Set Opcodes (Continued)

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Load / Store Instructions (Continued)						
ST *src*, *Ymem* **|| MAS**[**R**] *Xmem*, *dst*	Parallel store, multiply, and subtract	1 / 1	1101	1RSD	XXXX	YYYY
ST *src*, *Ymem* **|| MPY** *Xmem*, *dst*	Parallel store and multiply	1 / 1	1100	11SD	XXXX	YYYY
ST *src*, *Ymem* **|| SUB** *Xmem*, *dst*	Parallel store and subtract	1 / 1	1100	01SD	XXXX	YYYY
STL *src*, *Smem*	Store ACC low to data memory	1 / 1	1000	000S	IAAA	AAAA
STL *src*, **ASM**, *Smem*	Shift ACC low by ASM, store to data memory	1 / 1	1000	010S	IAAA	AAAA
STL *src*, *SHFT*, *Xmem*	Shift ACC low, then store to data memory	1 / 1	1001	100S	XXXX	SHFT
STL *src* [, *SHIFT*], *Smem*	Shift ACC low, then store to data memory (2-word opcode)	2 / 2	0110 0000	1111 110S	IAAA 100S	AAAA HIFT
STLM *src*, *MMR*	Store accumulator low to memory	1 / 1	1000	100S	IAAA	AAAA
STM *#lk*, *MMR*	Store ACC low into memory-mapped register	2 / 2	0111	0111	IAAA	AAAA
STRCD *Xmem*, *cond*	Store TREG conditionally	1 / 1	1001	1100	XXXX	COND
Logical Instructions						
AND *Smem*, *src*	AND single data-memory operand with ACC	1 / 1	0001	100S	IAAA	AAAA
AND *#lk* [, *SHFT*], *src* [, *dst*]	Shift long-immediate operand, AND with ACC	2 / 2	1111	00SD	0011	SHFT
AND *#lk*, **16**, *src* [, *dst*]	Shift long-immediate 16 bits, AND with ACC	2 / 2	1111	00SD	0110	0011
AND *src* [, *SHIFT*], [, *dst*]	AND accumulator(s), then shift result	1 / 1	1111	00SD	100S	HIFT
ANDM *#lk*, *Smem*	AND memory with long-immediate operand	2 / 2	0110	1000	IAAA	AAAA
BIT *Xmem*, *BITC*	Test bit	1 / 1	1001	0110	XXXX	BITC
BITF *Smem*, *#lk*	Test bit field specified by immediate value	2 / 2	0110	0001	IAAA	AAAA
BITT *Smem*	Test bit specified by TREG	1 / 1	0011	0100	IAAA	AAAA
CMPL *src* [, *dst*]	Complement accumulator	1 / 1	1111	01SD	1001	0011

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

Table 1–11. ’54x Instruction Set Opcodes (Continued)

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Logical Instructions (Continued)						
CMPM *Smem*, *#lk*	Compare memory with long-immediate operand	2/2	0110	0000	IAAA	AAAA
CMPR *CC*, *ARx*	Compare auxiliary register with AR0	1/1	1111	01CC	1010	1ARX
OR *Smem, src*	OR single data-memory operand with ACC	1/1	0001	101S	IAAA	AAAA
OR *#lk* [, *SHFT*], *src* [, *dst*]	Shift long-immediate operand, then OR with ACC	2/2	1111	00SD	0100	SHFT
OR *#lk,* **16**, *src* [, *dst*]	Shift long-immediate 16 bits, then OR with ACC	2/2	1111	00SD	0110	0100
OR *src* [*, SHIFT*], [, *dst*]	OR accumulator(s), then shift result	1/1	1111	00SD	101S	HIFT
ORM *#lk, Smem*	OR memory with constant	2/2	0110	1001	IAAA	AAAA
ROL *src*	Rotate accumulator left	1/1	1111	010S	1001	0001
ROLTC *src*	Rotate accumulator left using TC	1/1	1111	010S	1001	0010
ROR *src*	Rotate accumulator right	1/1	1111	010S	1001	0000
SFTA *src*, *SHIFT* [, *dst*]	Shift accumulator arithmetically	1/1	1111	01SD	011S	HIFT
SFTC *src*	Shift accumulator conditionally	1/1	1111	010S	1001	0100
SFTL *src*, *SHIFT* [, *dst*]	Shift accumulator logically	1/1	1111	00SD	111S	HIFT
XOR *Smem, src*	XOR operand with ACC	1/1	0001	110S	IAAA	AAAA
XOR *#lk* [, *SHFT*], *src* [, *dst*]	Shift long-immediate, then XOR with ACC	2/2	1111	00SD	0101	SHFT
XOR *#lk,* **16**, *src* [, *dst*]	Shift long-immediate 16 bits, then XOR with ACC	2/2	1111	00SD	0110	0101
XOR *src* [, *SHIFT*] [, *dst*]	XOR accumulator(s), then shift result	1/1	1111	00SD	110S	HIFT
XORM *#lk, Smem*	XOR memory with constant	2/2	0110	1010	IAAA	AAAA
Move Instructions						
MVDD *Xmem*, *Ymem*	Move within data memory, X/Y addressing	1/1	1110	0101	XXXX	YYYY
MVDK *Smem*, *dmad*	Move data, destination addressing	2/2	0111	0001	IAAA	AAAA
MVDM *dmad*, *MMR*	Move data to memory-mapped register	2/2	0111	0010	IAAA	AAAA
MVDP *Smem*, *pmad*	Move data to program memory	2/4	0111	1101	IAAA	AAAA

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

Table 1–11. ’54x Instruction Set Opcodes (Continued)

Mnemonic Syntax	Description	Words/ Cycles†	Opcode MSB			LSB
Move Instructions (Continued)						
MVKD *dmad*, *Smem*	Move data with source addressing	2/2	0111	0000	IAAA	AAAA
MVMD *MMR*, *dmad*	Move memory-mapped register to data	2/2	0111	0011	IAAA	AAAA
MVMM *MMRx*, *MMRy*	Move between memory-mapped registers	1/1	1110	0111	MMRX	MMRY
MVPD *pmad*, *Smem*	Move program memory to data memory	2/3	0111	1100	IAAA	AAAA
READA *Smem*	Read data memory addressed by ACCA	1/5	0111	1110	IAAA	AAAA
WRITA *Smem*	Write data memory addressed by ACCA	1/5	0111	1111	IAAA	AAAA

† Values for words and cycles assume the use of DARAM for data. Add one word and one cycle when using long-offset indirect addressing or absolute addressing with a single data-memory operand.
‡ Delayed Instruction
§ Condition true
¶ Condition false

1.7 Development Support

Texas Instruments (TI) offers an extensive line of development tools for the ’54x generation of DSPs, including tools to evaluate the performance of the processors, generate code, develop algorithm implementations, and fully integrate and debug software and hardware modules.

The following products support development of ’54x-based applications:

Software Development Tools:
Assembler/Linker
Simulator
Optimizing ANSI C compiler
Application algorithms
C/Assembly debugger and code profiler

Hardware Development Tools:
Extended development system (XDS™) emulator (supports ’54x multiprocessor system debug)
’54x EVM (Evaluation Module)
’54x DSK (DSP Starter Kit)

The *TMS320 Family Development Support Reference Guide* (SPRU011) contains information about development support products for all TMS320™ family member devices, including documentation. Refer to this document for further information about TMS320 documentation or any other TMS320 support products from Texas Instruments. There is an additional document, the *TMS320 Third-Party Support Reference Guide* (SPRU052), which contains information about TMS320-related products from other companies in the industry. To receive copies of TMS320 literature, contact the Literature Response Center at 800/477-8924.

1.7.1 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all TMS320 devices and support tools. Each TMS320 member has one of three prefixes: TMX, TMP, or TMS. Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS). This development flow is defined below.

Device development evolutionary flow:

TMX Experimental device that is not necessarily representative of the final device’s electrical specifications

TMP Final silicon die that conforms to the device’s electrical specifications but has not completed quality and reliability verification

TMS Fully-qualified production device

XDS and TMS320 are trademarks of Texas Instruments.

Support tool development evolutionary flow:

TMDX Development support product that has not yet completed Texas Instruments internal qualification testing.

TMDS Fully qualified development support product

TMX and TMP devices and TMDX development support tools are shipped against the following disclaimer:

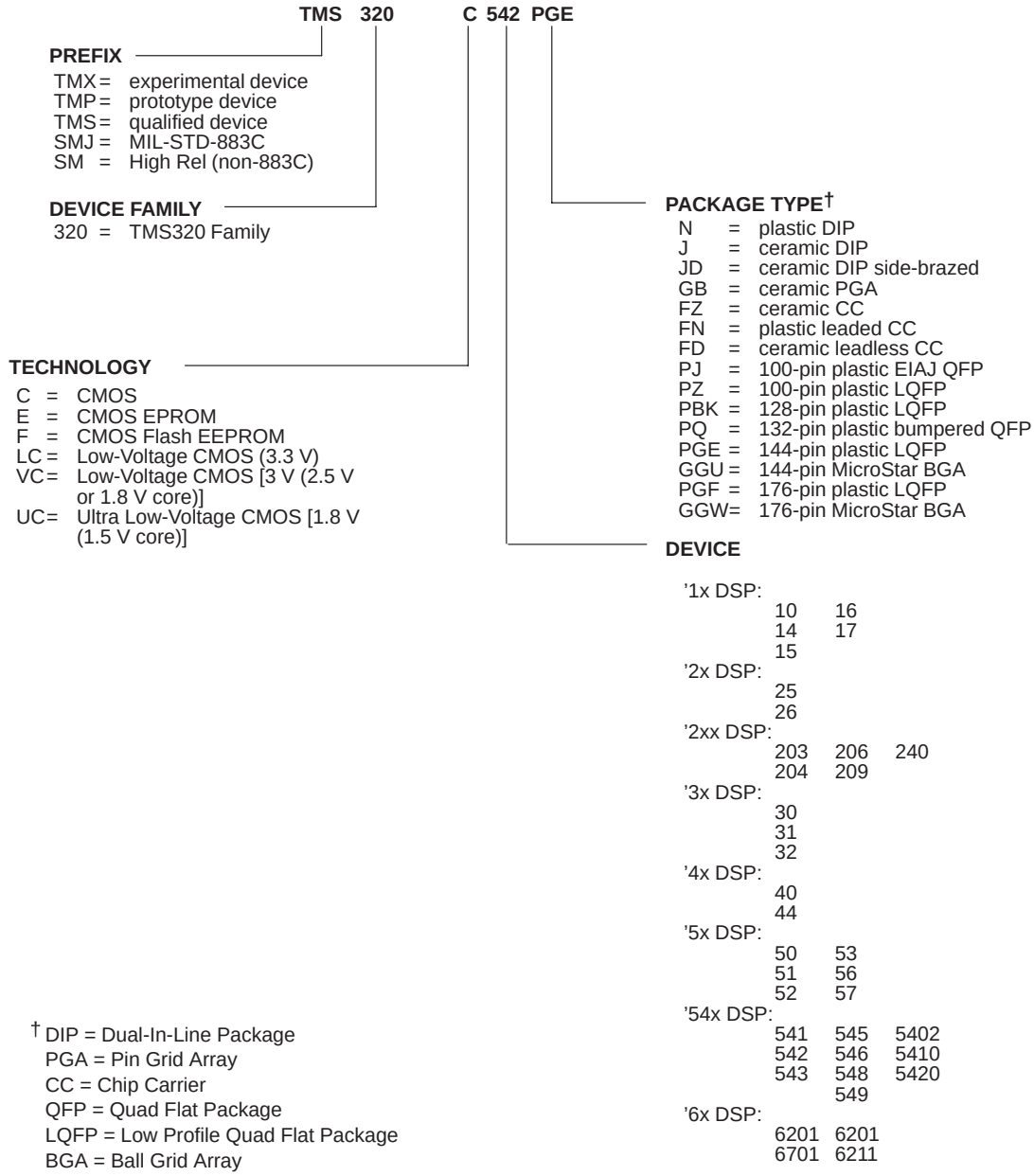
“Developmental product is intended for internal evaluation purposes.”

TMS devices and TMDS development support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI’s standard warranty applies.

Predictions show that prototype devices (TMX or TMP) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, PZ, PGE, PBK, or GGU) and temperature range (for example, L). Figure 1–3 provides a legend for reading the complete device name for any TMS320 family member.

Figure 1–3. TMS320 DSP Device Nomenclature



1.8 Documentation Support

Extensive documentation supports all TMS320 family generations of devices from product announcement through applications development. The following types of documentation available to support the design and use of the C5000 family of DSPs:

- Family Functional Overview (such as this document)
- Device-specific data sheets
- Complete User Guides
- Development support tools
- Hardware and software application reports

The four-volume *TMS320C54x DSP Reference Set* (literature number SPRU210) consists of:

- *Volume 1: CPU and Peripherals* (literature number SPRU131)
- *Volume 2: Mnemonic Instruction Set* (literature number SPRU172)
- *Volume 3: Algebraic Instruction Set* (literature number SPRU179)
- *Volume 4: Applications Guide* (literature number SPRU173)

The reference set describes in detail the ’54x TMS320 products currently available and the hardware and software applications, including algorithms, for fixed-point TMS320 devices.

For general background information on DSPs and TI devices, see the three-volume publication *Digital Signal Processing Applications with the TMS320 Family* (literature numbers SPRA012, SPRA016, and SPRA017).

A series of DSP textbooks is published by Prentice-Hall and John Wiley & Sons to support digital signal processing research and education. The TMS320 newsletter, *Details on Signal Processing*, is published quarterly and distributed to update TMS320 customers on product information.

Information regarding TI DSP products is also available on the Worldwide Web at *http://www.ti.com* uniform resource locator (URL).

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its products 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.

Customers are responsible for their applications using TI components.

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

TI assumes no liability for applications assistance or customer product design. TI does not 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 products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, license, warranty or endorsement thereof.

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. Representation or reproduction of this information with alteration voids all warranties provided for an associated TI product or service, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Resale of TI's 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, is an unfair and deceptive business practice, and TI is not responsible nor liable for any such use.

Also see: Standard Terms and Conditions of Sale for Semiconductor Products. www.ti.com/sc/docs/stdterms.htm

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265