TMS320VC5471 Fixed-Point Digital Signal Processor

Data Manual

Literature Number: SPRS180C
June 2001 – Revised December 2002

TEXAS INSTRUMENTS

IMPORTANT NOTICE

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

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

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

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

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

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

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265

REVISION HISTORY

REVISION	DATE	PRODUCT STATUS	HIGHLIGHTS
*	June 2001	Product Preview	Original
A	December 2001	Production Data	Updated characteristic data
B	July 2002	Production Data	Updated ARM SRAM read and write timing diagrams
C	December 2002	Production Data	Added industrial temperature specifications to the Recommended Operating Conditions and the Absolute Maximum Ratings (page 47). Updated mechanical drawing (page 87).

Contents

List of Figures

List of Tables

1 TMS320VC5471 Features

- **Dual CPU Processor Integrating a TMS320C54x™ DSP and an ARM7TDMI™ RISC MCU**
- **16-Bit Low-Power DSP With 72K x 16-bit Integrated SRAM Operates at up to 100 MHz**
- **Smart Power Management and Low-Power Modes for DSP and MCU Subsystems**
- **Integrated DSP Subsystem Peripherals**
  - **Two High-Speed, Full-Duplex Multichannel Buffered Serial Ports (McBSPs) Allowing the DSP Core to Interface Directly With CODECs**
  - **Six-Channel Direct Memory Access (DMA) Controller Enabling Six Independent Block Transfers With No Intervention From the DSP**
  - **ARM™ Port Interface (API) Provides 2K x 16-Bit Shared Memory Interface for Efficient Information Exchange Between the MCU Subsystem and the DSP Subsystem CPUs**
  - **External Memory Interface**
  - **Software-Programmable Wait-State Generator Capable of Extending External Bus Cycles By Up To 14 Machine Cycles**
  - **One Software-Programmable Hardware Timer For Control Operations**
  - **Programmable Phase-Locked Loop (PLL) Clock Generator**
- **ARM7TDMI RISC Microcontroller Core With 16K Bytes of Integrated SRAM and Enhanced Emulation Capabilities Operates at Up To 47.5 MHz**
- **Integrated MCU Subsystem Peripherals**
  - **Ethernet Interface Module With 10/100 Mb/s IEEE 802.3 Ethernet Media Access Controller (MAC)**
  - **Media Independent interface (MII) Port**
  - **Universal Asynchronous Receiver/ Transmitter (UART)**
  - **UART/IrDA Interface Which Supports the Slow Infrared (SIR) Protocol**
  - **Serial Peripheral Interface**
  - **Thirty-Six General-Purpose I/O Pins**
  - **Inter-Integrated Circuit (I^2C) Interface**
  - **Two General-Purpose Timers**
  - **One Watchdog Timer**
  - **Interrupt Handler**
  - **Interface to External Memory Supports Flash, SRAM, SDRAM, ROM**
  - **Flexible Clock Management for MCU Peripherals**
  - **Programmable Phase-Locked Loop (PLL) Clock Generator**
- **On-Chip Scan-Based Emulation Logic, IEEE Std 1149.1† (JTAG) Boundary Scan Logic of DSP and MCU Cores**
- **Supports Scan-Based Emulation of DSP and MCU Cores**
- **257-Ball MicroStar BGA™ (GHK Suffix) Package**

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

2 Introduction

This section describes the main features, gives a brief functional overview of the TMS320VC5471, lists the pin assignments, and provides a signal description table. The data manual also provides a detailed description section, electrical specifications, parameter measurement information, and mechanical data section describing the available packaging.

NOTE: This data manual is designed to be used in conjunction with the *TMS320C54x™ DSP Functional Overview* (literature number SPRU307) and the *TMS320VC547x CPU and Peripherals Reference Guide* (literature number SPRU038).

2.1 Description

The TMS320VC5471 integrates a DSP subsystem based on the TMS320C54x architecture and a RISC microcontroller subsystem based on the ARM7TDMI core as shown in Figure 2–1. The DSP subsystem includes 72K x 16-bit SRAM, a timer, a DMA controller, an external memory interface, and two McBSPs. The MCU subsystem includes three timers, general-purpose I/O, an external memory interface, and an Ethernet (10/100Base-T) interface with a media-independent interface (MII) port.

The TMS320VC5471 is implemented as two major subsystems that are highly independent. The DSP subsystem includes the following modules:

- TMS320C54x™ DSP core
- 72K x 16-bit internal SRAM organized as 32K x 16-bit of data SRAM and 40K x 16-bit of program SRAM.
- ARM port interface (API) to provide access by the MCU to 8K x 16 of the DSP's data SRAM
- Two multichannel buffered serial ports (McBSPs)
- Phase-locked loop (PLL)
- Timer
- Direct memory access (DMA) controller
- Programmable wait-state generator
- External memory interface

The MCU subsystem includes the following modules:

- ARM7TDMI CPU core (32/16-bit RISC processor) with extended emulation capabilities
- MCU memory interface for external SRAM, Flash, ROM, and SDRAM.
- On-chip 16K-byte (4K x 32) zero wait-state SRAM.
- MCU general-purpose I/Os (GPIOs), including support for an 8 x 8 keyboard.
- Three timers (two general-purpose, one watchdog)
- IrDA-compatible UART, supporting two modes
  - IrDA mode
  - UART mode without hardware flow control
- UART/Modem, with
  - hardware flow control support
  - autobaud function
- MCU subsystem interrupt handler
- MII port
- Ethernet 10/100Base-T interface
- Clock generator and control
- I^2C "master-only" interface
- Serial peripheral interface
- Phase-locked loop (PLL)

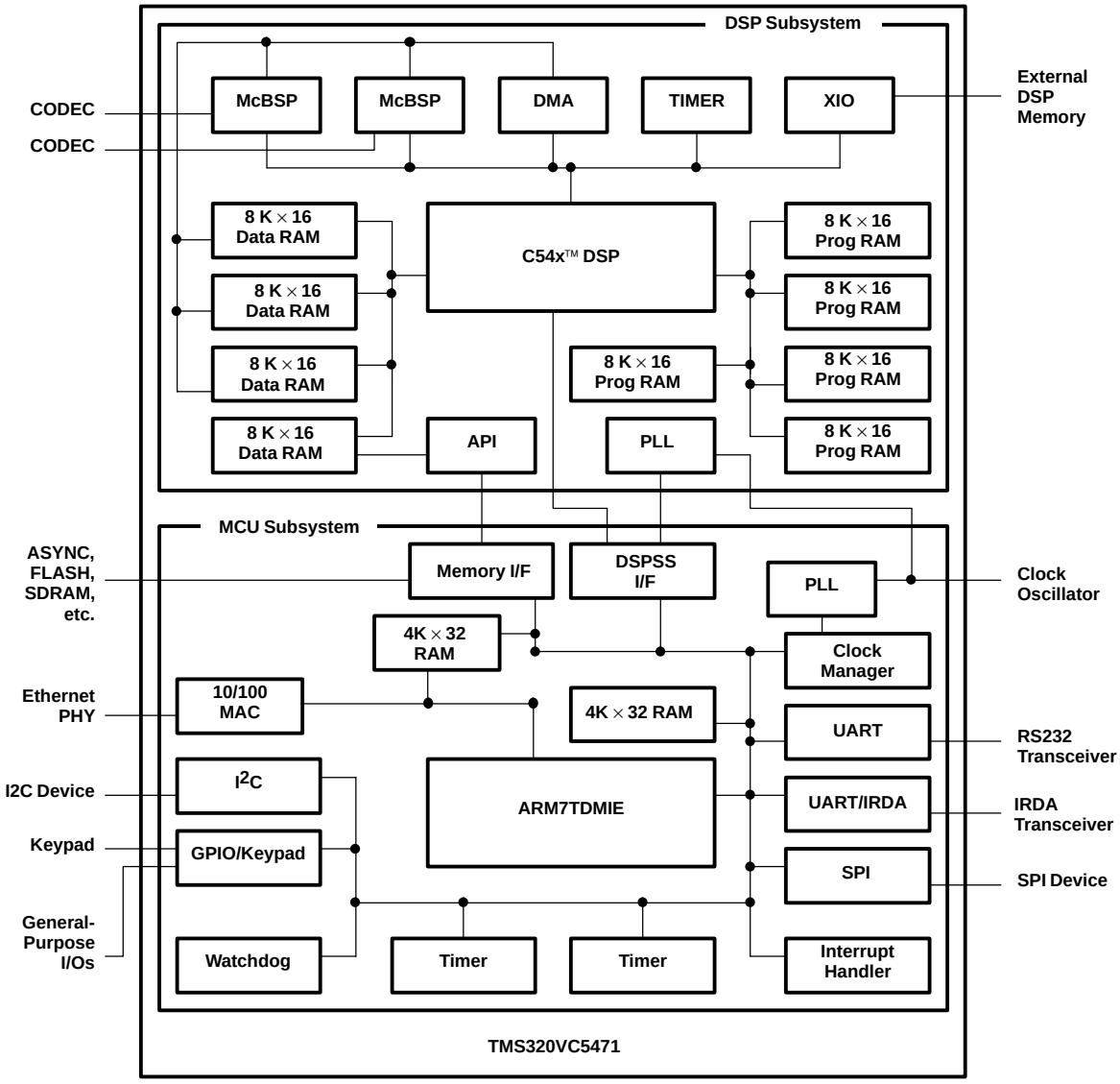


Figure 2–1. 5471 Functional Block Diagram

C54x is a trademark of Texas Instruments.

2.2 Pin Assignments

Figure 2–2 illustrates the ball locations for the 257-ball ball grid array (BGA) package and is used in conjunction with Table 2–1 to locate signal names and ball grid numbers.

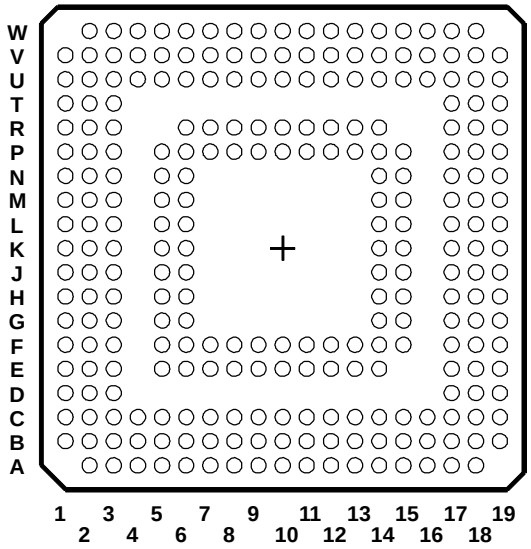


Figure 2–2. 257-Ball GHK Package (Bottom View)

Table 2–1. Pin Assignments for the GHK Package

BALL NO.	SIGNAL NAME	BALL NO.	SIGNAL NAME	BALL NO.	SIGNAL NAME	BALL NO.	SIGNAL NAME
B2	GPIO02	B1	GPIO03	D3	GPIO04	C2	GPIO05/ SDRAM_CKE
C1	GPIO06/ CLK16X_IRDA/SD_IRDA	D2	GPIO07/ TX_IRDA/TXIR_IRDA	D1	GPIO08/ RX_IRDA/RXIR_IRDA	E3	DV_{DD}
F5	GPIO09/ TX_MODEM	G6	V_{SS}	E2	GPIO10/ RX_MODEM	E1	GPIO11/ CTS_MODEM
F3	GPIO12/ RTS_MODEM	F2	GPIO13/ DCD_MODEM	G5	CV_{DD}	F1	GPIO18/ AUDIO_CLK
H6	GPIO19/ TIMER_OUT	G3	GPIO00	G2	GPIO14/ MCUEN2	G1	GPIO16/ SDA
H5	DV_{DD}	H3	GPIO17/ SCL	H2	V_{SS}	H1	KBGPIO14/ DSP_CLKOUT
J1	$\overline{\text{KBGPIO02}}$/ ARM_OPC	J2	$\overline{\text{KBGPIO03}}$/ ARM_MREQ	J3	$\overline{\text{KBGPIO04}}$/ ARM_EXEC	J5	$\overline{\text{KBGPIO06}}$/ ARM_FIQ
J6	$\overline{\text{KBGPIO07}}$/ ARM_IRQ	K1	KBGPIO13/ DSP_TOUT	K2	$\overline{\text{KBGPIO11}}$/ DSP_IAQ	K3	CV_{DD}
K5	KBGPIO05/ ARM_TBIT	K6	KBGPIO01/ DSP_XF	L1	KBGPIO15/ ARM_MCLK	L2	KBGPIO00
L3	V_{SS}	L6	$\overline{\text{KBGPIO12}}$/ DSP_MSC	L5	$\overline{\text{KBGPIO10}}$/ DSP_IACK	M1	KBGPIO08

NOTE: DV_{DD} is the 3.3 V power supply for the I/O pins while CV_{DD} is the 1.8 V power supply for the core CPU. V_{SS} is the ground for both the I/O pins and the core CPU.

Table 2–1. Pin Assignments for the GHK Package (Continued)

BALL NO.	SIGNAL NAME	BALL NO.	SIGNAL NAME	BALL NO.	SIGNAL NAME	BALL NO.	SIGNAL NAME
M2	KBGPIO09	M3	GPIO01	M6	GPIO15/ MCUEN1	M5	CLKX_SPI
N1	MCUDI/ EXTERN0	N2	MCUDO/ EXTERN1	N3	DV_{DD}	N6	MCUEN0
P1	CRS0	P2	COL0	N5	V_{SS}	P3	TXD00
R1	TXD01	P6	TXD03	R2	TXD02	P5	CV_{DD}
R3	TXEN0	T1	TCLK0	T2	TXER0	U1	RXER0
T3	DV_{DD}	U2	RCLK0	V1	RXDV0	U3	RXD01
V2	RXD00	W2	RXD02	U4	RXD03	V3	N/C
W3	N/C	V4	N/C	W4	N/C	U5	N/C
R6	V_{SS}	P7	N/C	V5	N/C	W5	N/C
U6	DV_{DD}	V6	N/C	R7	N/C	W6	N/C
P8	N/C	U7	N/C	V7	N/C	W7	N/C
R8	CV_{DD}	U8	N/C	V8	$\overline{EMU1}$	W8	$\overline{TRST}$
W9	TCK	V9	TDI	U9	V_{SS}	R9	TDO
P9	DV_{DD}	W10	TMS	V10	$\overline{EMU0}$	U10	REFCLK
R10	$\overline{RESET}$	P10	$\overline{RESET\_OUT}$	W11	ADD17	V11	ADD16
U11	ADD15	P11	ADD14	R11	ADD13	W12	ADD12/ SDRAM_A12
V12	ADD11/ SDRAM_A11	U12	V_{SS}	P12	ADD10/ SDRAM_A10	R12	DV_{DD}
W13	ADD01/ SDRAM_A01	V13	ADD09/ SDRAM_A09	U13	ADD08/ SDRAM_A08	P13	ADD07/ SDRAM_A07
W14	ADD06/ SDRAM_A06	V14	ADD05/ SDRAM_A05	R13	CV_{DD}	U14	ADD04/ SDRAM_A04
W15	ADD03/ SDRAM_A03	P14	V_{SS}	V15	ADD02/ SDRAM_A02	R14	DV_{DD}
U15	ADD00/ SDRAM_A00	W16	ADD18/ SDRAM_BA0	V16	ADD19 SDRAM_BA1	W17	$\overline{ADD20/ SDRAM\_WE}$
U16	$\overline{ADD21/ SDRAM\_CAS}$	V17	$\overline{ADD22/ SDRAM\_RAS}$	W18	CS4/ BIGEND	U17	DATA00
V18	DATA01	V19	DATA02	T17	V_{SS}	U18	DATA03
U19	DATA04	T18	DV_{DD}	T19	DATA05	R17	DATA06
P15	DATA07	N14	CV_{DD}	R18	DATA08	R19	DATA09
P17	DATA10	P18	DATA11	N15	V_{SS}	P19	DATA12
M14	DV_{DD}	N17	DATA13	N18	DATA14	N19	DATA15
M15	DATA16	M17	DATA17	M18	DATA18	M19	DATA19
L19	DATA20	L18	DATA21	L17	V_{SS}	L15	DATA22
L14	DATA23	K19	$\overline{BE0}/$ SDRAM_DQM0	K18	DV_{DD}	K17	$\overline{BE1}/$ SDRAM_DQM1
K15	$\overline{BE2}/$ SDRAM_DQM2	K14	$\overline{BE3}/$ SDRAM_DQM3	J19	$\overline{CS0}$	J18	$\overline{CS1}$
J17	$\overline{CS2}$	J14	$\overline{CS3}/$ ROMSIZE16	J15	V_{SS}	H19	$\overline{OE}$
H18	$\overline{WAIT}$	H17	DV_{DD}	H14	R/$\overline{W}$	H15	SDRAM_CLK
G19	$\overline{SDRAM\_CS}$	G18	DATA24	G17	CV_{DD}	G14	DATA25
F19	DATA26	F18	DATA27	G15	V_{SS}	F17	DATA28

NOTE: DV_{DD} is the 3.3 V power supply for the I/O pins while CV_{DD} is the 1.8 V power supply for the core CPU. V_{SS} is the ground for both the I/O pins and the core CPU.

Table 2–1. Pin Assignments for the GHK Package (Continued)

BALL NO.	SIGNAL NAME	BALL NO.	SIGNAL NAME	BALL NO.	SIGNAL NAME	BALL NO.	SIGNAL NAME
E19	DATA29	F14	DATA30	E18	DATA31	F15	DV_{DD}
E17	BDX0	D19	BFSX0	D18	BCLKX0	C19	BDR0
D17	V_{SS}	C18	BCLKR0	B19	BFSR0	C17	BDR1
B18	BFSR1	A18	BCLKR1	C16	BFSX1	B17	BCLKX1
A17	BDX1	B16	DSP_A00	A16	DSP_A01	C15	DSP_A02
E14	DV_{DD}	F13	V_{SS}	B15	DSP_A03	A15	DSP_A04
C14	DSP_A05	B14	DSP_A06	E13	DSP_A07	A14	DSP_A08
F12	V_{SS}	C13	DSP_A09	B13	DSP_A10	A13	DSP_A11
E12	DV_{DD}	C12	DSP_A12	B12	DSP_A13	A12	DSP_A14
A11	DSP_A15	B11	DSP_A16	C11	V_{SS}	E11	DSP_A17
F11	DSP_A18	A10	DSP_A19	B10	CV_{DD}	C10	DSP_D00
E10	DV_{DD}	F10	DSP_D01	A9	DSP_D02	B9	DSP_D03
C9	V_{SS}	F9	DSP_D04	E9	DSP_D05	A8	DSP_D06
B8	DSP_D07	C8	DV_{DD}	F8	DSP_D08	E8	DSP_D09
A7	DSP_D10	B7	DSP_D11	C7	V_{SS}	F7	DSP_D12
A6	DSP_D13	B6	DSP_D14	E7	DV_{DD}	C6	DSP_D15
A5	CV_{DD}	F6	$\overline{\text{DSP\_MSTRB}}$	B5	$\overline{\text{DSP\_DS}}$	E6	V_{SS}
C5	$\overline{\text{DSP\_INT0}}$	A4	$\overline{\text{DSP\_IOSTRB}}$	B4	$\overline{\text{DSP\_IS}}$	A3	$\overline{\text{DSP\_PS}}$
C4	DV_{DD}	B3	DSP_READY	A2	DSP_R/$\overline{W}$	C3	V_{SS}
E5	N/C						

NOTE: DV_{DD} is the 3.3 V power supply for the I/O pins while CV_{DD} is the 1.8 V power supply for the core CPU. V_{SS} is the ground for both the I/O pins and the core CPU.

2.3 Terminal Functions

Table 2–2 lists each terminal name, function, and operating mode(s) for the 5471 device. Some of the 5471 pins can be configured for one of two functions — a primary function and a secondary function. Table 2–3 provides a list of pins containing an internal pullup or pulldown function, including each pins functional pullup or pulldown value.

Table 2–2. Terminal Functions

TERMINAL NAME	I/O/Z†	DESCRIPTION
GPIO00–GPIO04	I/O	General-purpose input/output 00 through general-purpose input/output 04
GPIO05/SDRAM_CKE	I/O	General-purpose input/output 05 or SDRAM CKE signal
GPIO06/CLK16X_IRDA/SD_IRDA	I/O	General-purpose input/output 06 or 16x Serial Transmit Clock or IRDA Transceiver Shutdown
GPIO07/TX_IRDA/TXIR_IRDA	I/O	General-purpose input/output 07 or IRDA Transmit Data or IRDA Transmit Pulse
GPIO08/RX_IRDA/RXIR_IRDA	I/O	General-purpose input/output 08 or IRDA Receive Data or IRDA Receive Pulse
GPIO09/TX_MODEM	I/O	General-purpose input/output 09 or Modem TX Data Output
GPIO10/RX_MODEM	I/O	General-purpose input/output 10 or Modem RX Data Input
GPIO11/CTS_MODEM	I/O	General-purpose input/output 11 or Modem Clear-To-Send Input
GPIO12/RTS_MODEM	I/O	General-purpose input/output 12 or Modem Ready-To-Send Output
GPIO13/DCD_MODEM	I/O	General-purpose input/output 13 or Modem Carrier Detect Output
GPIO14/MCUEN2	I/O	General-purpose input/output 14 or SPI Enable Trigger bit 2
GPIO15/ MCUEN1	I/O	General-purpose input/output 15 or SPI Enable Trigger bit 1
GPIO16/SDA	I/O	General-purpose input/output 16 or I2C Serial Data
GPIO17/SCL	I/O	General-purpose input/output 17 or I2C Serial Clock
GPIO18/AUDIO_CLK	I/O	General-purpose input/output 18 or Audio Clock output
GPIO19/TIMER_OUT	I/O	General-purpose input/output 19 or MCU Timer 2 Output
KBGPIO00	I/O	Keyboard input/output 0
KBGPIO01/DSP_XF	I/O	Keyboard input/output 1or DSP XF output
KBGPIO02/$\overline{\text{ARM\_OPC}}$	I/O	Keyboard input/output 2 or active-low MCU opcode fetch output
KBGPIO03/$\overline{\text{ARM\_MREQ}}$	I/O	Keyboard input/output 3 or active-low MCU memory request output
KBGPIO04/$\overline{\text{ARM\_EXEC}}$	I/O	Keyboard input/output 4 or active-low MCU executed output
KBGPIO05/ARM_TBIT	I/O	Keyboard input/output 5 or MCU 16/32-bit instruction execution output
KBGPIO06/$\overline{\text{ARM\_FIQ}}$	I/O	Keyboard input/output 6 or active-low MCU fast interrupt output
KBGPIO07/$\overline{\text{ARM\_IRQ}}$	I/O	Keyboard input/output 7 or MCU normal interrupt output
KBGPIO08–KBGPIO09	I/O	Keyboard input/output bits 8–9‡
KBGPIO10/$\overline{\text{DSP\_IACK}}$	I/O	Keyboard input/output 10 or DSP interrupt acknowledge output‡
KBGPIO11/$\overline{\text{DSP\_IAQ}}$	I/O	Keyboard input/output 11 or active-low DSP instruction acquisition signal output‡
KBGPIO12/$\overline{\text{DSP\_MSC}}$	I/O	Keyboard input/output 12 or active-low DSP microstate complete signal output‡
KBGPIO13/DSP_TOUT	I/O	Keyboard input/output 13 or DSP timer output‡
KBGPIO14/ DSP_CLKOUT	I/O	Keyboard input/output 14 or DSP clock output‡
KBGPIO15/ARM_MCLK	I/O	Keyboard input/output 15 or MCU clock output‡
CLKX_SPI	O	SPI Serial clock
MCUDI/EXTERN0	I	SPI Input serial data/MCU EXTERN0 test signal input
MCUDO/EXTERN1	I/O	SPI Output serial data/MCU EXTERN1 test signal input

† I = input, O = output, Z = high-impedance
‡ The KBGPIO[15:8] pins include integrated pullup resistors.
§ The ARM7TDMI core can operate in either little- or big-endian mode. The 5471 configures the ARM7TDMI core's endianness by sampling this pin at the $\overline{\text{RESET}}$ rising edge.
¶ 5471 only supports MCU boot from a 16-bit or a 32-bit-wide memory implementation on $\overline{\text{CS0}}$. The 5471 configures the ARM7TDMI starting instruction size, and width of memory connected to $\overline{\text{CS0}}$, by sampling this pin at the $\overline{\text{RESET}}$ rising edge.

Table 2–2. Terminal Functions (Continued)

TERMINAL NAME	I/O/Z†	DESCRIPTION
MCUEN0	O	SPI enable trigger (edge/level, positive/negative)
CRS0	I	MII0 carrier sense
COL0	I	MII0 collision
TXD00–TXD03	O	MII0 TX data bit 0 through MII0 TX data bit 3
TXEN0	O	MII0 TX enable
TCLK0	I	MII0 TX clock
TXER0	O	MII0 TX error
RXER0	I	MII0 RX error
RCLK0	I	MII0 RX clock
RXDV0	I	MII0 RX data valid
RXD00 – RXD03	I	MII0 RX data bit 0 through MII0 RX data bit 3
$\overline{\text{EMU0}}$–$\overline{\text{EMU1}}$	I/O	Test emulation pins 0 and 1, active-low
$\overline{\text{TRST}}$	I	Test reset input, active-low. A pulldown resistor is suggested to allow both normal device operation and emulation.
TCK	I	Test Clock
TDI	I	Test Data Input
TDO	O	Test Data Output
TMS	I	Test Mode Select
REFCLK	I	Reference input clock
$\overline{\text{RESET}}$	I	Chip power-on reset, active-low§¶
$\overline{\text{RESET\_OUT}}$	O	Reset to external peripherals, active-low
ADD00/ SDRAM_A00 – ADD12/SDRAM_A12	O	MCU address bus bits 0 through 12 for Flash, SRAM, and SDRAM
ADD13–ADD17	O	MCU address bus bits 13 through 17 for Flash and SRAM
ADD18/ SDRAM_BA0	O	MCU address bus bit 18 for Flash and SRAM or SDRAM BA0 signal
ADD19/ SDRAM_BA1	O	MCU address bus bit 19 for Flash and SRAM or SDRAM BA1 signal
ADD20/ $\overline{\text{SDRAM\_WE}}$	O	MCU address bus bit 20 for Flash and SRAM or active-low SDRAM WE signal
ADD21/ $\overline{\text{SDRAM\_CAS}}$	O	MCU address bus bit 21 for Flash and SRAM or active-low SDRAM CAS signal
ADD22/ $\overline{\text{SDRAM\_RAS}}$	O	MCU address bus bit 22 for Flash and SRAM or active-low SDRAM RAS signal
CS4/ BIGEND	I/O	MCU Chip Select 4 (active-high).§ This pin is sampled at the $\overline{\text{RESET}}$ rising edge to determine the endian mode as defined below: 0 = Little Endian 1 = Big Endian For best results, an external pullup or pulldown should be installed to ensure proper sampling.
DATA00–DATA31	I/O	MCU data bus bits 0 through 31 for Flash, SRAM, and SDRAM
$\overline{\text{BE0}}$/ SDRAM_DQM0	O	MCU byte-enable for non-SDRAM cycles, or SDRAM byte-enable (for read) and mask (for write), active-low
$\overline{\text{BE1}}$/ SDRAM_DQM1	O	MCU byte-enable for non-SDRAM cycles, or SDRAM byte-enable (for read) and mask (for write), active-low
$\overline{\text{BE2}}$/ SDRAM_DQM2	O	MCU byte-enable for non-SDRAM cycles, or SDRAM byte-enable (for read) and mask (for write), active-low

† I = input, O = output, Z = high-impedance
‡ The KBGPIO[15:8] pins include integrated pullup resistors.
§ The ARM7TDMI core can operate in either little- or big-endian mode. The 5471 configures the ARM7TDMI core's endianness by sampling this pin at the $\overline{\text{RESET}}$ rising edge.
¶ 5471 only supports MCU boot from a 16-bit or a 32-bit-wide memory implementation on $\overline{\text{CS0}}$. The 5471 configures the ARM7TDMI starting instruction size, and width of memory connected to $\overline{\text{CS0}}$, by sampling this pin at the $\overline{\text{RESET}}$ rising edge.

Table 2–2. Terminal Functions (Continued)

TERMINAL NAME	I/O/Z†	DESCRIPTION
$\overline{\text{BE3}}$/ SDRAM_DQM3	O	MCU byte-enable for non-SDRAM cycles, or SDRAM byte-enable (for read) and mask (for write), active-low
$\overline{\text{CS0}}$	O	MCU chip select 0, active-low
$\overline{\text{CS1}}$	O	MCU chip select 1, active-low
$\overline{\text{CS2}}$	O	MCU chip select 2, active-low
$\overline{\text{CS3}}$/ ROMSIZE16	I/O	MCU chip select 3, active-low.¶ Input sampled at $\overline{\text{RESET}}$ rising edge to determine the width of the ARM processor during initialization as defined below: 0 = 16 bits 1 = 32 bits For best results, an external pullup or pulldown should be installed to ensure proper sampling.
$\overline{\text{OE}}$	O	MCU output-enable for peripherals connected to the MCU memory interface (not for SDRAM), active-low
$\overline{\text{WAIT}}$	I	MCU wait request, active-low. If this signal is unused, it must be pulled high with an external pullup resistor.
R/$\overline{\text{W}}$	O	Asynchrouous memory active-high read / active-low write signal (not for SDRAM)
SDRAM_CLK	O	SDRAM Clock
$\overline{\text{SDRAM\_CS}}$	O	SDRAM chip-select, active-low
BDX0	O/Z	McBSP0 transmit serial data
BFSX0	I/O/Z	McBSP0 transmit frame synchronization
BCLKX0	I/O/Z	McBSP0 transmit clock
BDR0	I	McBSP0 receive serial data
BCLKR0	I/O/Z	McBSP0 receive clock
BFSR0	I/O/Z	McBSP0 receive frame synchronization
BDR1	I	McBSP1 receive serial data
BFSR1	I/O/Z	McBSP1 receive frame synchronization
BCLKR1	I/O/Z	McBSP1 receive clock
BFSX1	I/O/Z	McBSP1 transmit frame synchronization
BCLKX1	I/O/Z	McBSP1 transmit clock
BDX1	O/Z	McBSP1 transmit data
DSP_A00–DSP_A19	O/Z	DSP Address bits 0 through 19
DSP_D00–DSP_D15	I/O/Z	DSP Data bus bits 0 through 15
$\overline{\text{DSP\_MSTRB}}$	O/Z	DSP memory strobe signal, active-low
$\overline{\text{DSP\_DS}}$	O/Z	DSP data space select, active-low
$\overline{\text{DSP\_INT0}}$	I	DSP external interrupt, active-low. If this signal is unused, it should be pulled high with an external pullup resistor.
$\overline{\text{DSP\_IOSTRB}}$	O/Z	DSP I/O strobe signal, active-low
$\overline{\text{DSP\_IS}}$	O/Z	DSP I/O space select, active-low
$\overline{\text{DSP\_PS}}$	O/Z	DSP program space select, active-low
DSP_READY	I	DSP data ready input (tie high if not used), active-high. If this signal is unused, it must be pulled high with an external pullup resistor.
DSP_R/$\overline{\text{W}}$	O/Z	DSP active-high read / active-low write signal

† I = input, O = output, Z = high-impedance
‡ The KBGPIO[15:8] pins include integrated pullup resistors.
§ The ARM7TDMI core can operate in either little- or big-endian mode. The 5471 configures the ARM7TDMI core's endianness by sampling this pin at the $\overline{\text{RESET}}$ rising edge.
¶ 5471 only supports MCU boot from a 16-bit or a 32-bit-wide memory implementation on $\overline{\text{CS0}}$. The 5471 configures the ARM7TDMI starting instruction size, and width of memory connected to $\overline{\text{CS0}}$, by sampling this pin at the $\overline{\text{RESET}}$ rising edge.

Table 2–2. Terminal Functions (Continued)

TERMINAL NAME	I/O/Z†	DESCRIPTION
V_{SS}		Ground
CV_{DD}		1.8-V supply for core logic
DV_{DD}		3.3-V supply for I/O cells
N/C		No Connection

† I = input, O = output, Z = high-impedance
‡ The KBGPIO[15:8] pins include integrated pullup resistors.
§ The ARM7TDMI core can operate in either little- or big-endian mode. The 5471 configures the ARM7TDMI core's endianness by sampling this pin at the $\overline{\text{RESET}}$ rising edge.
¶ 5471 only supports MCU boot from a 16-bit or a 32-bit-wide memory implementation on $\overline{\text{CS0}}$. The 5471 configures the ARM7TDMI starting instruction size, and width of memory connected to $\overline{\text{CS0}}$, by sampling this pin at the $\overline{\text{RESET}}$ rising edge.

Table 2–3. Internal Pullup/Pulldown Terminal List

TERMINAL NAME	I/O	BALL #	TYPE
DSP_D00	I/O/Z	C10	PS100, 100-µA pullup
DSP_D01	I/O/Z	F10	PS100, 100-µA pullup
DSP_D02	I/O/Z	A9	PS100, 100-µA pullup
DSP_D03	I/O/Z	B9	PS100, 100-µA pullup
DSP_D04	I/O/Z	F9	PS100, 100-µA pullup
DSP_D05	I/O/Z	E9	PS100, 100-µA pullup
DSP_D06	I/O/Z	A8	PS100, 100-µA pullup
DSP_D07	I/O/Z	B8	PS100, 100-µA pullup
DSP_D08	I/O/Z	F8	PS100, 100-µA pullup
DSP_D09	I/O/Z	E8	PS100, 100-µA pullup
DSP_D10	I/O/Z	A7	PS100, 100-µA pullup
DSP_D11	I/O/Z	B7	PS100, 100-µA pullup
DSP_D12	I/O/Z	F7	PS100, 100-µA pullup
DSP_D13	I/O/Z	A6	PS100, 100-µA pullup
DSP_D14	I/O/Z	B6	PS100, 100-µA pullup
DSP_D15	I/O/Z	C6	PS100, 100-µA pullup
KBGPIO08	I/O	M1	PS100, 100-µA pullup
KBGPIO09	I/O	M2	PS100, 100-µA pullup
KBGPIO10/$\overline{\text{DSP\_IACK}}$	I/O	L5	PS100, 100-µA pullup
KBGPIO11/$\overline{\text{DSP\_IAQ}}$	I/O	K2	PS100, 100-µA pullup
KBGPIO12/$\overline{\text{DSP\_MSC}}$	I/O	L6	PS100, 100-µA pullup
KBGPIO13/DSP_TOUT	I/O	K1	PS100, 100-µA pullup
KBGPIO14/DSP_CLKOUT	I/O	H1	PS100, 100-µA pullup
KBGPIO15/ARM_MCLK	I/O	L1	PS100, 100-µA pullup
GPIO16/SDA	I/O	G1	PS100, 100-µA pullup
$\overline{\text{CS4}}$/BIGEND	I/O	W18	PE100, 100-µA pulldown
$\overline{\text{CS3}}$/ROMSIZE16	I/O	J14	PS100, 100-µA pullup
RXD00	I	V2	PS020, 20-µA pullup
RXD01	I	U3	PS020, 20-µA pullup
RXD02	I	W2	PS020, 20-µA pullup
RXD03	I	U4	PS020, 20-µA pullup
RXDV0	I	V1	PS020, 20-µA pullup
RXER0	I	U1	PS020, 20-µA pullup
RCLK0	I	U2	PS020, 20-µA pullup
TCLK0	I	T1	PS020, 20-µA pullup
COL0	I	P2	PS020, 20-µA pullup
CRS0	I	P1	PS020, 20-µA pullup
$\overline{\text{EMU0}}$	I/O	V10	PS100, 100-µA pullup
$\overline{\text{EMU1}}$	I/O	V8	PS100, 100-µA pullup
TCK	I	W9	PE100, 100-µA pulldown
$\overline{\text{TRST}}$	I	W8	PE100, 100-µA pulldown
TMS	I/O	W10	PS100, 100-µA pullup
TDI	I/O	V9	PS100, 100-µA pullup

3 DSP Subsystem Functional Overview

The DSP subsystem is based on the TMS320C54x™ DSP core, on-chip memories and peripherals, and is code-compatible with other C54x products. The following description of the 5471 DSP subsystem is based on Figure 3–1.

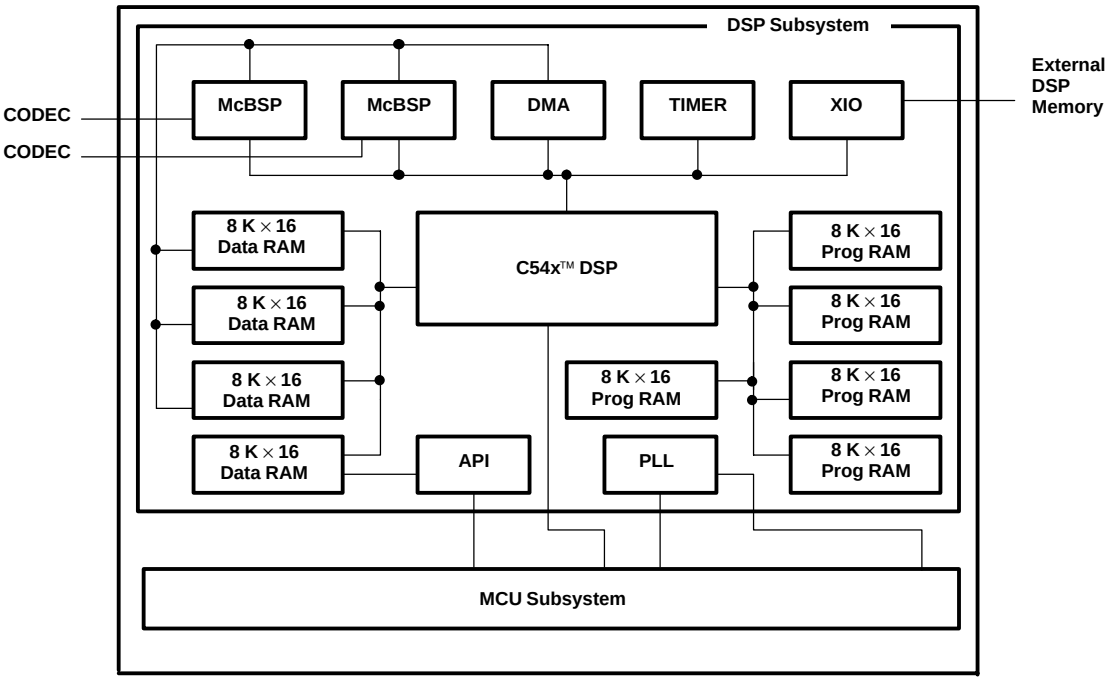


Figure 3–1. 5471 DSP Subsystem Functional Block Diagram

3.1 DSP Core

The 5471 DSP subsystem's fixed-point, digital signal processor (DSP) is based on an advanced modified Harvard architecture that has one program memory bus and three data memory buses. This processor provides an arithmetic logic unit (ALU) with a high degree of parallelism, application-specific hardware logic, on-chip memory, and additional on-chip peripherals. The basis of the operational flexibility and speed of this DSP is a highly specialized instruction set.

Separate program and data spaces allow simultaneous access to program instructions and data, providing the high degree of parallelism. Two read operations 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. In addition, the 5471 DSP subsystem includes the control mechanisms to manage repeated operations, function calls, and DSP interrupts.

The DSP core includes the following features:

- Low-power C54x™ DSP CPU, operating at up to 100 MHz
- Software-programmable wait-state generator with bank-switching wait-state logic
- External memory interface
  - Program space
  - Data space
  - I/O space
- Scan-based emulation logic

The 5471 DSP subsystem includes the DSP CPU core, a programmable phase-locked loop (PLL) for clock generation, an interface to external parallel devices, a timer, 72K words of RAM, two multichannel buffered serial interfaces, an interface to allow MCU access to part of the DSP subsystem memory map, and a JTAG interface.

3.1.1 DSP Memory Space and Buses

The 5471 subsystem has multiple memory spaces and four parallel buses that allow you to access both program and data simultaneously. Each of the four buses access different memory spaces for different aspects of the DSP operation:

- The DSP subsystem includes 72K words of on-chip RAM, as well as an extensive external memory range, which can be used to interface to a variety of memory types or peripherals.
- The program bus (PB) reads from program memory space, which contains the instructions to be executed.
- The write data bus (EB) writes into data memory space, which stores data used by the instructions and tables eventually used in execution. It also writes into I/O memory space.
- The two read data buses (CB and DB) read data from the data memory space and the DB data bus accesses I/O memory space. The I/O memory space interfaces to external memory-mapped peripherals and can serve as extra data storage space.

3.1.2 DSP Scan-Based Emulation Logic

The 5471 DSP subsystem includes a dedicated emulation port for in-circuit emulation. The emulation port is accessed directly by the Texas Instruments (TI) extended development system (XDS) hardware emulator and provides emulation.

3.2 DSP Memory

The 5471 device implements 72K words of on-chip RAM as follows:

- 40K words of program-space single-access RAM (SARAM)
- 16K words of data space dual-access RAM (DARAM)
- 16K words of data space single-access RAM (SARAM)

Each block of DARAM may perform up to two DSP accesses in one machine cycle. The DSP subsystem may also perform multiple accesses to separate memory blocks in one machine cycle. After a “normal” reset, the data-space RAM blocks in the addresses between 0x0000 and 0x7FFF are mapped into data memory space only, and the program space RAM blocks between 0x06000 and 0x0FFFF are mapped only to program space. The OVLY and DROM bits can affect this as shown in Figure 3–2 and Figure 3–3.

3.2.1 DSP On-Chip RAM

The DSP subsystem features 72K x 16-bit of on-chip RAM (two blocks of 8K x 16-bit DARAM and seven blocks of 8K x 16-bit SARAM). The DSP CPU can perform two accesses to a DARAM in one machine cycle (two reads in one cycle, or a read and a write in one cycle). It can also perform multiple accesses to separate memory blocks in one machine cycle.

After reset, the lower address range of the program space is mapped to external memory, the lower address range of the data space is mapped with on-chip RAM blocks. However, the OVLY bit in the PMST register can be used to map these RAM blocks into both program and data space.

3.2.2 Normal Mode DSP Memory Map

The normal mode DSP subsystem provides the memory map shown in Figure 3–2. This is the memory map that applies when the API boot mode feature is not enabled. The normal mode memory map applies any time that DSP register BSCR[4] (ABMDIS) is 1, or when MCU register DSP_REG[9] (DSP_APIBN) is 1.

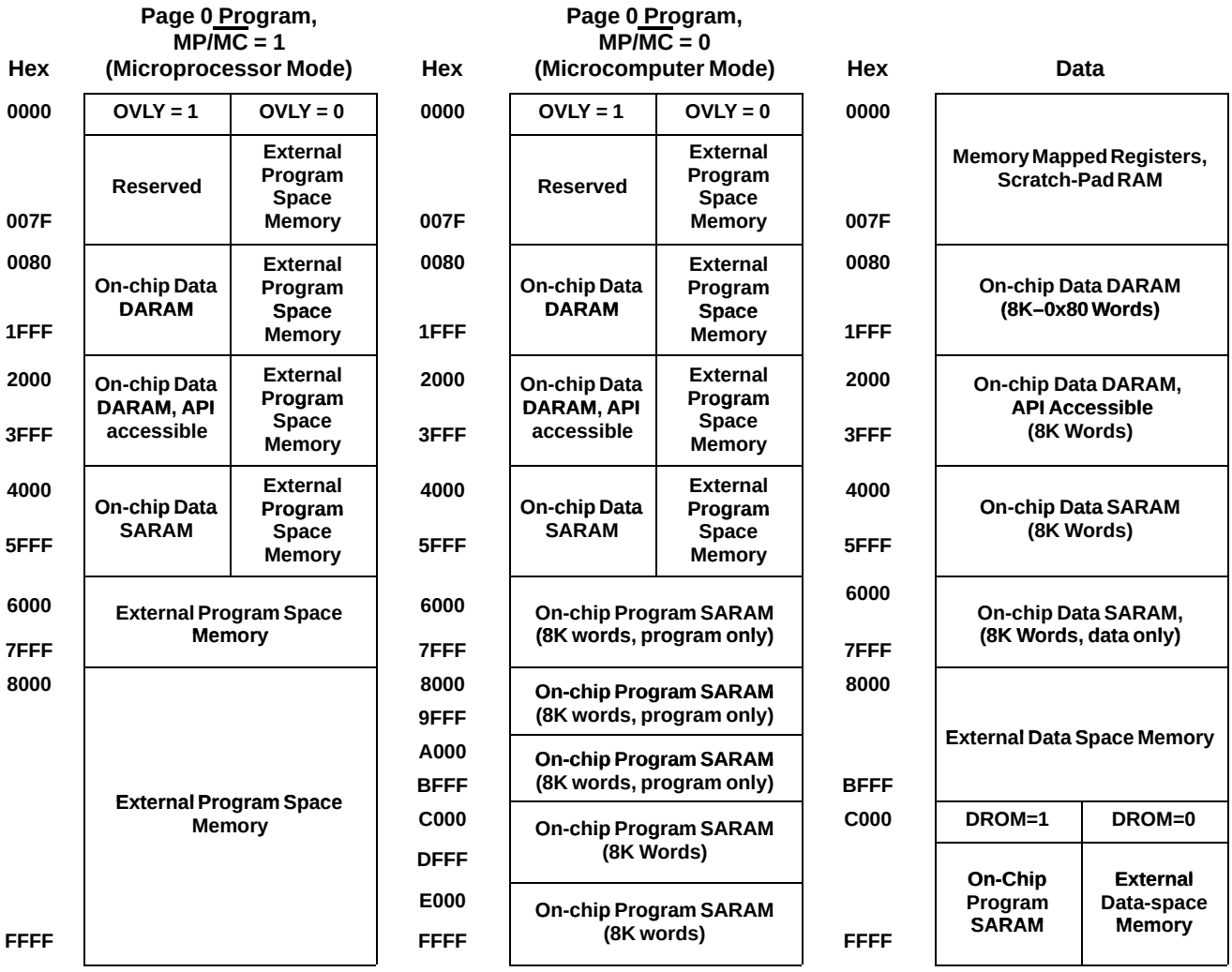


Figure 3–2. Memory Map for DSP Accesses (DSP_APIBN = 1 or ABMDIS = 1)

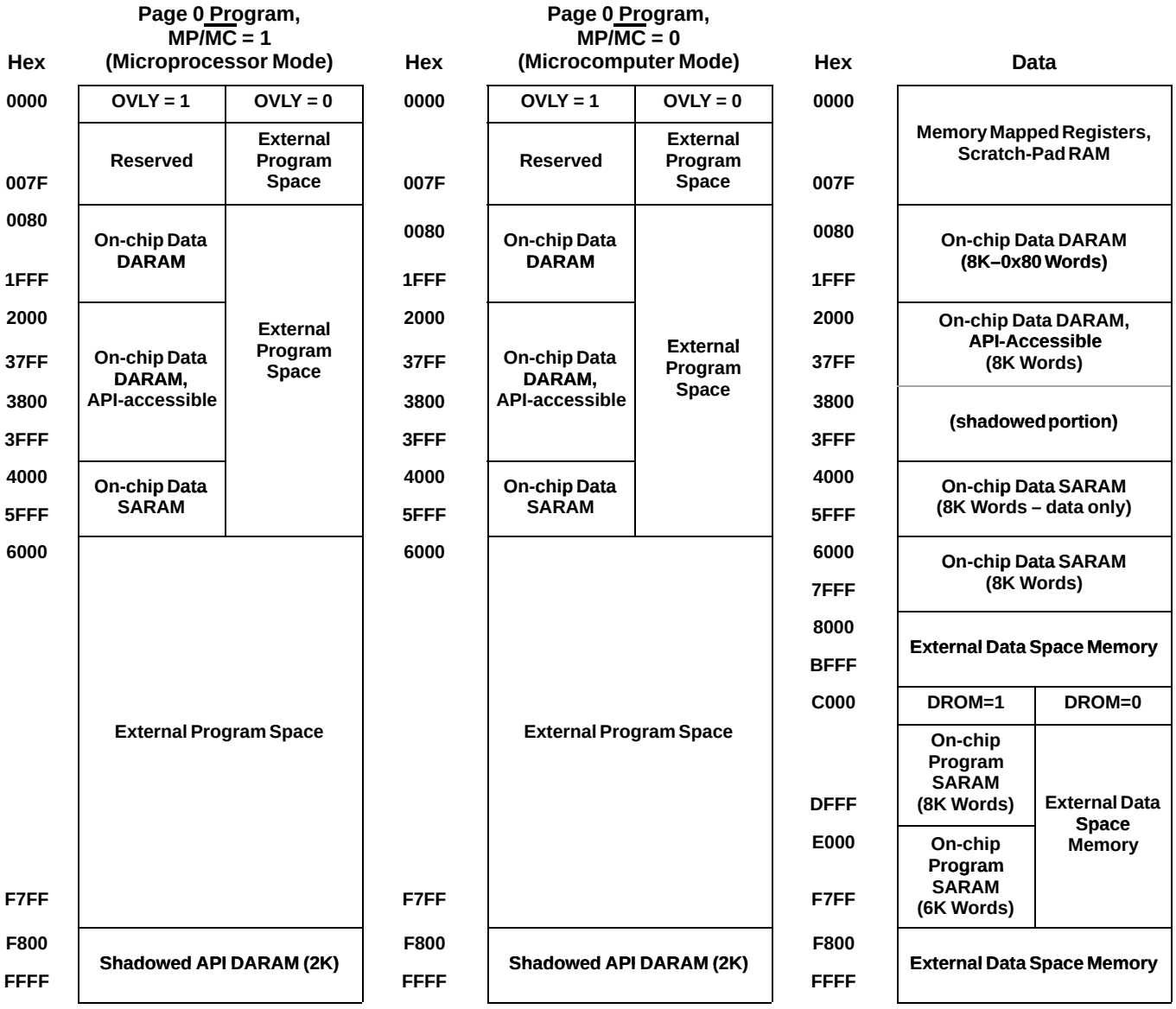
3.2.3 API Boot Mode

Under normal DSP subsystem reset conditions, the DSP will begin operation either from code that has been previously uploaded to internal program-space RAM (when in microcontroller mode) or from external memory on the XIO bus (microprocessor mode). Uploading the code to internal program-space RAM may be performed by the MCU. When the DSP subsystem is in API boot mode, the upper 2K words of the 8K-word API SARAM is shadowed so that it is found both in DSP data space at 0x3800–0x3FFF and in DSP program space at 0xF800 to 0xFFFF, which includes the DSP's reset vector.

To make use of this mode, the MCU directly controls the DSP's reset and API boot mode signals. It holds the DSP in reset, enables API boot mode, and loads the DSP boot code via the API. Once it has loaded the code to the appropriate location in the API SARAM, the MCU releases the DSP from reset, and the DSP begins executing from the normal reset location, where the uploaded code now resides. Once the DSP code has completed any API boot activity, it may disable the memory remapping provided by the API boot mode by writing a 1 to BSCR[4] (ABMDIS). It is recommended that the MCU change the API boot mode input signal only when the DSP is held in reset.

3.2.4 API Boot-Mode DSP Memory Map

The memory map when the API boot mode feature is enabled is shown in Figure 3–3. API boot mode is enabled when DSP_REG[9] (DSP_APIBN) is 0 and BSCR[4] (ABMDIS) is 0.



† When DSP_REG[9] (DSP_APIBN) = 0 and BSCR[4] (ABMDIS) = 0, 2K words of the API DARAM are remapped to program-space, regardless of DSP_REG[10] (MP/MC) value. All other internal program-space RAMs are disabled in program space. Overlayable data-space RAMs may be dual-mapped to program-space via OVLY.

Figure 3–3. API Boot Mode Memory Map for DSP Accesses (DSP_APIBN = 0 and ABMDIS = 0)†

3.2.5 DSP Extended Program Memory

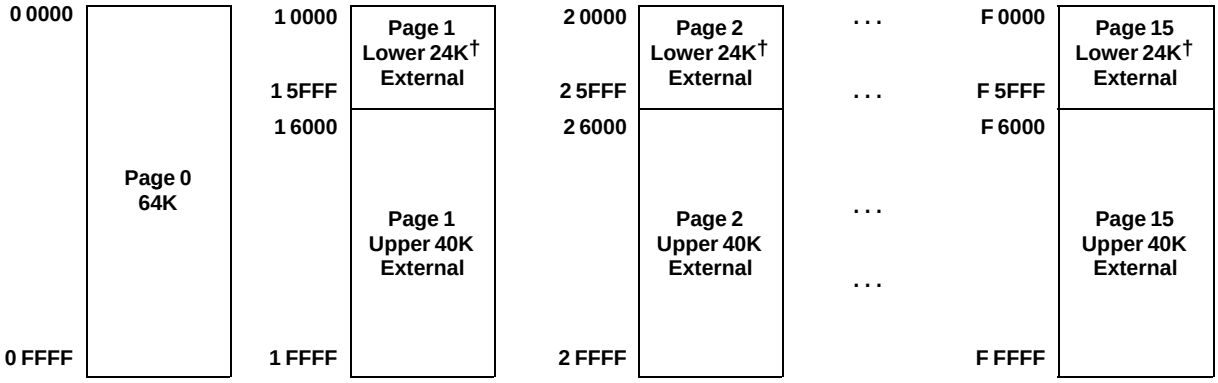
The DSP subsystem includes a memory paging scheme to extend the number of addressable program space locations from 64K to 1M words. The four extended address pins (DSP_A16 to DSP_A19) are used to address 15 pages of program memory. Each page includes 64K addressable locations. The extended program addresses are supported by eight instructions: FB[D], FBACC[D], FCALA[D], FCALL[D], FRET[D], FRETE[D], READA, and WRITA.

- 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
- 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 instructions, please refer to the *TMS320C54x DSP Reference Set, Volume 2: Mnemonic Instruction Set* (literature number SPRU172).

When the OVLY bit is set, each page of program memory is made up of two parts: a common block of 24K words maximum and a unique block of 40K words minimum. The common block is shared by all pages, and each unique block is accessible only through its assigned page.

The value of the program counter extension register (XPC) defines the page selection. At a hardware reset, the XPC is initialized to 0.



† Accesses to the lower 24K words of pages 1 through 15 are to external program memory only when the OVLY bit is cleared to 0. If the OVLY bit is set to 1, on-chip RAM is mapped to 0x0 to 0x5FFF of pages 1 through 15. Note external address pins are provided to support 15 external pages.

Figure 3–4. DSP Extended Program Memory Map

3.2.6 DSP 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 (RS) vector cannot be remapped because a hardware reset loads the IPTR with 1s. Therefore, the reset vector is always fetched at location FF80h in program space.

3.3 DSP Registers

The 5471 has 27 memory-mapped CPU registers, which are mapped in data memory space addresses 0h to 1Fh. Table 3–1 gives a list of CPU memory-mapped registers (MMRs) available on 5471. The device also has a set of memory-mapped registers associated with peripherals as shown in Table 3–2.

Table 3–1. CPU Memory-Mapped Registers

NAME	ADDRESS		DESCRIPTION
	DEC	HEX	
IMR	0	0	Interrupt mask register
IFR	1	1	Interrupt flag register
–	2–5	2–5	Reserved for testing
ST0	6	6	Status register 0
ST1	7	7	Status register 1
AL	8	8	Accumulator A low word (15–0)
AH	9	9	Accumulator A high word (31–16)
AG	10	A	Accumulator A guard bits (39–32)
BL	11	B	Accumulator B low word (15–0)
BH	12	C	Accumulator B high word (31–16)
BG	13	D	Accumulator B guard bits (39–32)
TREG	14	E	Temporary register
TRN	15	F	Transition register
AR0	16	10	Auxiliary register 0
AR1	17	11	Auxiliary register 1
AR2	18	12	Auxiliary register 2
AR3	19	13	Auxiliary register 3
AR4	20	14	Auxiliary register 4
AR5	21	15	Auxiliary register 5
AR6	22	16	Auxiliary register 6
AR7	23	17	Auxiliary register 7
SP	24	18	Stack pointer register
BK	25	19	Circular buffer size register
BRC	26	1A	Block repeat counter
RSA	27	1B	Block repeat start address
REA	28	1C	Block repeat end address
PMST	29	1D	Processor mode status (PMST) register
XPC	30	1E	Extended program page register
–	31	1F	Reserved

This lists the 5471 DSP subsystem peripheral registers and describes those that are specific to the 5471 or are different from the standard C54x registers. The standard C54x registers are defined in the *TMS320VC5409 Fixed-Point Digital Signal Processor* data sheet (literature number SPRS082) and the *TMS320VC5402 Fixed-Point Digital Signal Processor* data sheet (literature number SPRS079). The DSP subsystem peripheral mapping is shown in Table 3–2.

Table 3–2. Peripheral Memory-Mapped Registers

NAME	ADDRESS	DESCRIPTION	TYPE
DRR20	20h	Data receive register 2	McBSP #0
DRR10	21h	Data receive register 1	McBSP #0
DXR20	22h	Data transmit register 2	McBSP #0
DXR10	23h	Data transmit register 1	McBSP #0
TIM	24h	Timer register	Timer
PRD	25h	Timer period counter	Timer
TCR	26h	Timer control register	Timer
–	27h	Reserved	
SWWSR	28h	Software wait-state register	External Bus
BSCR	29h	Bank-switching control register	External Bus, API
–	2Ah	Reserved	
SWCR	2Bh	Software wait-state control register	External Bus
–	2Ch–37h	Reserved	
SPSA0	38h	McBSP0 subbank address register	McBSP #0
SPSD0	39h	McBSP0 subbank data register	McBSP #0
–	3Ah–3Bh	Reserved	
GPIOCR	3C	General-purpose I/O pins control register	GPIO
GPIOSR	3D	General-purpose I/O pins status register	GPIO
–	3E–3F	Reserved	
DRR21	40h	Data receive register 2	McBSP #1
DRR11	41h	Data receive register 1	McBSP #1
DXR21	42h	Data transmit register 2	McBSP #1
DXR11	43h	Data transmit register 1	McBSP #1
–	44h–47h	Reserved	
SPSA1	48h	McBSP1 subbank address register	McBSP #1
SPSD1	49h	McBSP1 subbank data register	McBSP #1
–	4Ah–53h	Reserved	
DMPREC	54h	DMA channel priority and enable control register	DMA
DMSA	55h	DMA subbank address register	DMA
DMSDI	56h	DMA subbank data register with autoincrement	DMA
DMSDN	57h	DMA subbank data register	DMA
CLKMD	58h	Clock mode register	PLL
–	59h–5Fh	Reserved	

Table 3–3. McBSP Control Registers and Subaddresses

McBSP0		McBSP1		SUB ADDRESS	DESCRIPTION
NAME	ADDRESS	NAME	ADDRESS		
SPCR10	39h	SPCR11	49h	00h	Serial port control register 1
SPCR20	39h	SPCR21	49h	01h	Serial port control register 2
RCR10	39h	RCR11	49h	02h	Receive control register 1
RCR20	39h	RCR21	49h	03h	Receive control register 2
XCR10	39h	XCR11	49h	04h	Transmit control register 1
XCR20	39h	XCR21	49h	05h	Transmit control register 2
SRGR10	39h	SRGR11	49h	06h	Sample rate generator register 1
SRGR20	39h	SRGR21	49h	07h	Sample rate generator register 2
MCR10	39h	MCR11	49h	08h	Multichannel register 1
MCR20	39h	MCR21	49h	09h	Multichannel register 2
RCERA0	39h	RCERA1	49h	0Ah	Receive channel enable register partition A
RCERB0	39h	RCERB1	49h	0Bh	Receive channel enable register partition B
XCERA0	39h	XCERA1	49h	0Ch	Transmit channel enable register partition A
XCERB0	39h	XCERB1	49h	0Dh	Transmit channel enable register partition B
PCR0	39h	PCR1	49h	0Eh	Pin control register

3.3.1 Bank-Switching Control Register

The 5471 bank-switching control register (BSCR) controls both the bank-switching wait-state generation functions that are consistent with the 5409, and several 5471-specific features. These special features include control of configuration of some DSP subsystem external memory interface functionality and some aspects of the interface whereby the MCU may access a portion of the DSP subsystem RAM. The BSCR is shown in Figure 3–5 and the bits are described in Table 3–4.

The BSCR register also gives the DSP some control over the ARM programming interface, the mechanism which enables the MCU to access portions of the DSPs internal RAM. Included are the API mode (APIMODE), an interrupt to the MCU (HINT), and DSP memory map selection (ABMDIS).

15 12	11	10 5	4	3	2	1	0
BNKCMP	PS–DS	Reserved	ABM DIS	HINT	API MODE	Reserved	EXIO
R/W-1111	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R = Read, W = Write

Figure 3–5. Bank-Switching Control Register (BSCR)

Table 3–4. Bank-Switching Control Register (BSCR) Bit Fields

BIT NO.	BIT NAME	RESET VALUE	FUNCTION
15–12	BNKCMP	1111	Bank compare. Determines the external bank size. BNKCMP is used to mask the four MSBs of an address. For example, if BNKCMP = 1111b, the four MSBs (bits 15–12) are compared, resulting in a bank size of 4K words. Bank sizes of 4K words to 64K words are allowed. Table 3–5 shows the relationship between BNKCMP and the address range.
11	PS–DS	1	Program read – data read access. Inserts an extra cycle between consecutive accesses of program read and data read, or data read and program read. – PS–DS = 0: No extra cycles are inserted by this feature – PS–DS = 1: One extra cycle is inserted between consecutive data and program reads.
10–5	Reserved	0	These bits are reserved and will retain written values.
4	ABMDIS	0	API boot mode disable. This bit will force the DSP out of API boot mode. This bit has no effect when the DSP_APIBN is high. When DSP_APIBN is low, a 0 in this bit enables API boot mode and a 1 disables API boot mode. Note that a DSP subsystem reset will clear this bit and thus re-enable the DSP_APIBN port. – ABMDIS = 0: API boot mode is under the control of DSP_APIBN – ABMDIS = 1: DSP_APIBN is ignored and DSP subsystem is kept out of API boot mode.
3	HINT	0	Host processor interrupt. This bit enables/disables an interrupt to the MCU from the DSP. At reset, the HINT bit is cleared. To send a proper interrupt to the MCU, the DSP must write a 1 to HINT, delay an appropriate time, then write a 0 to HINT to create a pulse of an appropriate duration on the interrupt signal to the MCU. – HINT = 0: Interrupt signal to the MCU is not active – HINT = 1: Interrupt signal to the MCU is active
2	APIMODE	0	HOM/SAM enable. This bit enables/disables the API host-only mode. – APIMODE = 0: The shared-access mode (SAM) is enabled. Both the DSP and the MCU may access the API RAM. – APIMODE = 1: The host-only mode (HOM) is enabled. The MCU may access the API RAM; the DSP may not access the API RAM. DSP writes to the API RAM are ignored, and DSP reads from the API RAM return undefined values. DSP IDLE modes have no effect on MCU accesses to the API memory when in host-only mode.
1	Reserved	0	These bits are reserved and will retain written values.
0	EXIO	0	External bus interface off. The EXIO bit controls the external bus-off function. – EXIO = 0: The external bus interface functions as usual. – EXIO = 1: The address bus, data bus, and control signals become inactive after completing the current bus cycle. Table 3–6 lists the states of the external interface signals when the interface is disabled. Note that the DROM, MP/$\overline{MC}$, and OVLY bits in the PMST and the HM bit of ST1 cannot be modified when the interface is disabled.

3.3.2 Programmable Bank-Switching Wait-States

The bank-switching features of the programmable bank-switching logic of the 5471 DSP subsystem is functionally equivalent to that of the 5409. This feature automatically inserts one cycle when accessing across memory-bank boundaries within program or data memory space. A bank-switching wait-state can also be automatically inserted when accessing across the data space to program space boundary. These features are programmed via the BNKCMP and PS–DS bits in the BSCR.

The relationship between BNKCMP, the address bits to be compared, and the bank size is summarized in Table 3–5. The BNKCMP values that are not listed in the table are not allowed.

Table 3–5. Relationship Between BNKCMP and Bank Size

BNKCMP				MSBs TO COMPARE	BANK SIZE (16-BIT WORDS)
Bit 15	Bit 14	Bit 13	Bit 12		
0	0	0	0	None	64K
1	0	0	0	15	32K
1	1	0	0	15–14	16K
1	1	1	0	15–13	8K
1	1	1	1	15–12	4K

The external memory interface can be disabled when not used. This feature can be used to reduce the power consumption of the 5471 DSP subsystem. Table 3–6 lists the states of the external interface pins, when the interface is disabled (EXIO=1).

Table 3–6. State of Signals When External Bus Interface is Disabled (EXIO = 1)

SIGNAL	STATE	SIGNAL	STATE
DSP_A(19–0)	Previous State	$\overline{\text{DSP\_MSTRB}}$	High Level
DSP_D(15–0)	High Impedance	$\overline{\text{DSP\_IOSTRB}}$	High Level
$\overline{\text{DSP\_PS}}$	High Level	DSP_R/$\overline{\text{W}}$	High Level
$\overline{\text{DSP\_DS}}$	High Level	KBGPIO12/$\overline{\text{DSP\_MSC}}$†	High Level
$\overline{\text{DSP\_IS}}$	High Level	KBGPIO11/$\overline{\text{DSP\_IAQz}}$‡	High Level

† Only applies when KBGPIO12 is enabled to output $\overline{\text{DSP\_MSC}}$.

‡ Only applies when KBGPIO11 is enabled to output $\overline{\text{DSP\_IAQ}}$

3.3.3 Processor Mode Status Register

The 5471 DSP subsystem processor mode status register (PMST) differs from the typical C54x design in that the PMST CLKOUT bit has no effect on the CLKOUT port of the subsystem. This port is always clocking when the DSP subsystem's PLL is active.

3.4 DSP Peripherals

The following section describes how the 5471 DSP subsystem peripherals differ from standard peripherals as found on the TMS320VC5409.

3.4.1 Multichannel Buffered Serial Port (McBSP)

The 5471 provides high-speed, full-duplex serial ports that are similar to those found on the 5409 device. They allow direct interface to other 54x devices, codecs, and other devices in a system. There are two multichannel buffered serial ports (McBSPs) within the DSP subsystem.

3.4.1.1 Deriving Sample Clock from an External Clock

To accommodate applications that require an external reference clock to be divided down to create the frame sync clock and internal sample rate clock, the 5471 McBSP allows 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 TMS320C5000™ DSP platform devices. The new bit layout of the PCR is shown in Figure 3–6. For a description of the remaining bits, see *TMS320C54x DSP Reference Set, Volume 5: Enhanced Peripherals* (literature number SPRU302).

15	14	13	12	11	10	9	8
Reserved		XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM
RW		RW	RW	RW	RW	RW	RW
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SCLKME	CLKS STAT	DX STAT	DR STAT	FSXP	FSRP	CLKXP	CLKRP
RW	RW	RW	RW	RW	RW	RW	RW

LEGEND: R = Read, W = Write

Figure 3–6. Pin Control Register (PCR)

The selection of the sample rate generator (SRG) clock input source is made by the combination of the CLKSM and SCLKME bit values as shown in Table 3–7.

Table 3–7. Sample Rate Generator Clock Source Selection

SCLKME	CLKSM	SRG CLOCK SOURCE
0	0	Reserved
0	1	DSP_CPU clock
1	0	BCLKR pin
1	1	BCLKX pin

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 since the BCLKR pin is used in input mode.

TMS320C5000 is a trademark of Texas Instruments.

3.4.2 DSP Direct Memory Access (DMA) Controller

The DSP subsystem includes a six-channel DMA controller, for performing data transfers independent of the DSP. There are several limitations that the 5471 DMA controller places on its DMA transfers. The DMA controller has no access to the RAMs that are normally in program space, cannot access RAMs that are implemented in the MCU subsystem, and cannot access the RAM block in DSP data space that is connected to the API interface. The DMA controller cannot transfer between the McBSP DRR and DXR registers, nor can it transfer between McBSP DRR or DXR registers and external resources. The DMA controller cannot perform 32-bit accesses to external resources.

3.4.2.1 DMA Controllers View of the DSP Memory Map

Note that the DMA controller has access only to the data space DARAM at 0x0080 to 0x1FFF and the data space SARAMs that are at 0x4000 to 0x7FFF. It does not have access to any of the SARAMs in program space between 0x6000 and 0xFFFF, or to the API DARAM at 0x2000 to 0x3FFF. The API module has access to the 8K x 16 DARAM block at data space 0x2000 to 0x3FFF. Note also that the DMA controller can access external data space devices with extended addressing, while the DSP can only access page 0 of external data space.

3.4.2.2 DSP DMA External Access

The DMA supports external accesses to extended program space, extended data space, and extended I/O space.

- No more than one channel may be programmed for external writes at any time
- No more than one channel may be programmed for external reads at any time
- Only single-word transfers are supported for external accesses.
- The DMA does not support transfers from peripherals to external memory.
- The DMA does not support transfers from external memory to the peripherals.

3.4.2.3 DSP DMA Transfers in Doubleword Mode (Internal Only)

Doubleword mode allows the DMA to transfer 32-bit words in any index mode. In doubleword 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.

3.4.2.4 DSP DMA Interrupts

The ability of the DMA to interrupt the DSP 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 3–8.

Table 3–8. DMA Interrupts

MODE	DINM	IMOD	INTERRUPT
ABU (non-decrement)	1	0	At full buffer only
ABU (non-decrement)	1	1	At half buffer and full buffer
Multi-Frame	1	0	At block transfer complete (DMCTRn = DMSEFCn[7:0] = 0)
Multi-Frame	1	1	At end of frame and end of block (DMCTRn = 0)
ABU or Multi-Frame	0	X	No interrupt generated
ABU or Multi-Frame	0	X	No interrupt generated

3.4.2.5 DMA controller synchronization events

The internal transfers associated with each DMA channel can be synchronized to one of several events. The DSYN bit field of the DMSEFCn register selects the synchronization event for a channel. The list of possible events and the DSYN values are shown in Table 3–9.

Table 3–9. DMA Synchronization Events

DSYN VALUE	DMA SYNCHRONIZATION EVENT
0000b	No synchronization used
0001b	McBSP0 receive event
0010b	McBSP0 transmit event
0011b	Reserved
0100b	Reserved
0101b	McBSP1 receive event
0110b	McBSP1 transmit event
0111b	Reserved
1000b	Reserved
1001b	Reserved
1010b	Reserved
1011b	Reserved
1100b	Reserved
1101b	Timer interrupt event
1110b	External interrupt 0
1111b	Reserved

3.4.2.6 DSP DMA Channel Interrupt Selection

The DMA controller can generate a DSP interrupt for each of the six channels. However, due to a limit on the number of internal DSP interrupt inputs, channels 0, 1, 2, and 3 are multiplexed with other interrupt sources. DMA channels 0, 1, 2, and 3 share an interrupt line with the receive and transmit portions of the McBSP. When the DSP is reset, the interrupts from these three DMA channels are deselected. The INTSEL bit field in the DMPREC register can be used to select these interrupts, as shown in Table 3–10.

Table 3–10. DMA Channel Interrupt Selection

INTSEL Value	IMR/IFR[6]	IMR/IFR[7]	IMR/IFR[10]	IMR/IFR[11]
00b (reset)	Reserved	Reserved	BRINT1	BXINT1
01b	Reserved	Reserved	DMAC2	DMAC3
10b	DMAC0	DMAC1	DMAC2	DMAC3
11b	Reserved			

3.4.3 ARM Port Interface (API)

Information is exchanged between the MCU and the DSP through the on-chip shared API memory. The API memory is a 8K $\times$ 16-bit word DARAM (dual-access RAM) block. The API memory can also be used by the DSP as general-purpose data or program DARAM. In this circuit, only the DSP memory has DARAM.

The API has two modes of operation selected by the bank switching control register (BSCR): shared-access mode (SAM) and host-only mode (HOM). In SAM, both the DSP and the MCU can access the API memory. In SAM, asynchronous host accesses from the MCU are resynchronized internally. If the DSP and the MCU try to perform an access at the same time, the MCU has access priority and the DSP waits one cycle. SAM can be run when the DSP is in IDLE1 mode. In HOM, only the MCU can access the API memory. When HOM is selected, API memory blocks are disabled for DSP access, no value can be read from nor written to those memory locations.

SAM is the default configuration when the DSP exits from a reset phase. SAM is normally selected whenever the DSP is in normal operating mode or in IDLE 1. HOM is normally selected before the DSP is placed in IDLE2 or IDLE3.

3.4.4 DSP External Memory Interface

The 5471 external memory interface is largely identical to that of the 5409. Significant differences include the lack of a MP/$\overline{MC}$ pin, and multiplexing of $\overline{DSP\_MSC}$, $\overline{DSP\_IAQ}$, $\overline{DSP\_IACK}$, and DSP_CLKOUT functions on KBGPIO pins.

3.4.5 DSP Software-Programmable Wait-State Generator

The 5471 DSP subsystem's software wait-state generator is compatible with the 5409. Wait-states programmed in the SWWSR register apply to both DSP-generated and DMA-generated cycles to external memory peripherals on the DSP's XIO bus.

3.4.6 DSP Timer

The 5471 DSP timer is identical to that found on the 5409.

3.4.7 DSP Clocking

The DSP subsystem is capable of operating in DIV mode and PLL (normal) mode. The MCU may program the initial DSP initial clocking mode, and DSP code may change to other clocking modes. The DSP controls the clocking mode of operation via the settings in the CLKMD register.

The DSP subsystem's PLL module, when operating in PLL (normal) mode, provides a multiplied version of the REFCLK input clock as the DSP subsystem clock. The PLL can multiply by integer values between 1 and 15, by 0.5*n (for integer values of n between 1 and 15, inclusive), or by 0.25*n (for integer values of n between 1 and 15, inclusive). To ensure that the PLL locks to the input frequency, the input clock frequency must meet the minimum frequency as shown for each set of scaling values in Table 3–11.

Table 3–11. DSP Clock Scaler Values and Minimum REFCLK Frequencies

k (SCALER)	MINIMUM REFCLK FREQUENCY (MHZ)
k = n, for integer values of n between 1 and 15, inclusive	5
k = 0.5 * n, for integer values of n between 1 and 15, inclusive	10
k = 0.25 * n, for integer values of n between 1 and 15, inclusive	20

When in DIV mode, the PLL minimum input frequencies do not apply.

The PLL_DSP output frequency is dynamically selected by the contents of the memory-mapped DSP_CLKMD peripheral register. The initial settings for this register are controlled by a memory-mapped register, DSP_REG, on the RISC processor. The PLL should not be programmed to exceed the DSP subsystem's maximum operating frequency as defined in the parametric portion of this data manual.

3.5 DSP Power Management

The DSP subsystem has three power-down modes, which are activated by the IDLE1, IDLE2, and IDLE3 instructions. In these modes, the C54x DSP core enters a dormant state and dissipates considerably less power than in normal operation.

The IDLE1 mode halts all DSP activities except the DSP system clock. Because the system clock remains applied to the DSP subsystem peripheral modules, the DSP peripheral circuits continue operating and the DSP_CLKOUT pin remains active. Thus, peripherals such as serial ports and timers can take the DSP out of its power-down state.

The IDLE2 mode halts the DSP subsystem peripherals as well as the DSP core, but the DSP subsystem's phase-locked loop clock multiplier will remain active to allow for a rapid resume from IDLE2. Because the DSP subsystem peripherals are stopped in this mode, they cannot be used to generate the interrupt to wake up the C54x as with IDLE1. However, power is significantly reduced because the device is completely stopped. To terminate IDLE2, activate either $\overline{\text{RESET}}$ or $\overline{\text{DSP\_INT0}}$ with a pulse that meets the requirements defined in the parametric section of this data manual. Note that $\overline{\text{RESET}}$ assertion will cause the MCU subsystem to hold the DSP subsystem in reset.

The IDLE3 mode functions like IDLE2 but it also halts the DSP phase-locked loop (PLL) circuit. IDLE3 is used to achieve the lowest possible DSP power consumption. This mode reduces power dissipation more than IDLE2. Furthermore, the IDLE3 state allows you to reconfigure the DSP PLL externally if the system requires the C54x to operate at a lower speed to save power. To terminate IDLE3, activate $\overline{\text{RESET}}$ or $\overline{\text{DSP\_INT0}}$ with a pulse that meets the requirements defined in the parametric section of this data manual. Note that $\overline{\text{RESET}}$ assertion will cause the MCU subsystem to hold the DSP subsystem in reset.

3.6 DSP Interrupts

The DSP subsystem interrupts are mapped as shown in Table 3–12.

Table 3–12. DSP Interrupt Mapping

NAME	ADDRESS		PRIORITY	FUNCTION
	DEC	HEX		
RS, SINTR	0	00	1	Reset
SINT16	4	04	2	Software interrupt #16
SINT17	8	08	—	Software interrupt #17
SINT18	12	0C	—	Software interrupt #18
SINT19	16	10	—	Software interrupt #19
SINT20	20	14	—	Software interrupt #20
SINT21	24	18	—	Software interrupt #21
SINT22	28	1C	—	Software interrupt #22
SINT23	32	20	—	Software interrupt #23
SINT24	36	24	—	Software interrupt #24
SINT25	40	28	—	Software interrupt #25
SINT26	44	2C	—	Software interrupt #26
SINT27	48	30	—	Software interrupt #27
SINT28	52	34	—	Software interrupt #28
SINT29	56	38	—	Software interrupt #29
SINT30	60	3C	—	Software interrupt #30
INT0, SINT0	64	40	3	External user interrupt 0
SINT1	68	44	4	Software interrupt #1
SINT2	72	48	5	Software interrupt #2
TINT, SINT3	76	4C	6	Timer interrupt
BRINT0, SINT4	80	50	7	McBSP #0 receive interrupt (default)
BXINT0, SINT5	84	54	8	McBSP #0 transmit interrupt (default)
DMAC0, SINT6	88	58	9	DMA channel 0, Software interrupt #6
DMAC1, SINT7	92	5C	10	DMA channel 1, Software interrupt #7
SINT8	96	60	11	Software interrupt #8
AINT, SINT9	100	64	12	API interrupt
BRINT1, DMAC2, SINT10	104	68	13	McBSP #1 receive interrupt (default)
BXINT1, DMAC3, SINT11	108	6C	14	McBSP #1 transmit interrupt (default)
DMAC4,SINT12	112	70	15	DMA channel 4 interrupt (default)
DMAC5,SINT13	116	74	16	DMA channel 5 interrupt (default)
–	120–127	78–7F	—	Reserved

The bits of the interrupt flag register (IFR) and interrupt mask register (IMR) are arranged as shown in Figure 3–7. The function of each bit is described in Table 3–13.

15	14	13	12	11	10	9	8
Reserved		DMAC5	DMAC4	BXINT1/ DMAC3	BRINT1/ DMAC2	AINT	Reserved

7	6	5	4	3	2	1	0
DMAC1	DMAC0	BXINT0	BRINT0	TINT	Reserved		INT0

Figure 3–7. IFR and IMR Registers

Table 3–13. IFR and IMR Register Bit Fields

BIT		FUNCTION
NUMBER	NAME	
15–14	–	Reserved
13	DMAC5	DMA channel 5 interrupt flag/mask bit
12	DMAC4	DMA channel 4 interrupt flag/mask bit
11	BXINT1/DMAC3	McBSP1 transmit interrupt flag/mask bit
10	BRINT1/DMAC2	McBSP1 receive interrupt flag/mask bit
9	AINT	ARM-to-DSP interrupt flag/mask
8	–	Reserved
7	DMAC1	DMA channel 1 interrupt flag/mask bit
6	DMAC0	DMA channel 0 interrupt flag/mask bit
5	BXINT0	McBSP0 transmit interrupt flag/mask bit
4	BRINT0	McBSP0 receive interrupt flag/mask bit
3	TINT	Timer interrupt flag/mask bit
2–1	–	Reserved
0	INT0	External interrupt 0 flag/mask bit

4 MCU Subsystem Functional Overview

The 5471 MCU subsystem includes TI's emulation-enhanced ARM7TDMI microcontroller core and several peripherals including SPI and I^2C interfaces, UARTs, timers, general-purpose input/output, and external memory interface. The MCU subsystem provides 4K x 32 bits of general-purpose RAM and 4K x 32 bits of Ethernet packet RAM. The following description of the 5471 MCU subsystem is based on Figure 4–1.

4.1 MCU Core

The MCU subsystem uses Texas Instruments' emulation-enhanced ARM71TDMIE core, a derivative of the ARM Limited ARM7TDMI core. The ARM7TDMI processor core is a member of the ARM7 Thumb™ family. It is a low power 32-bit RISC processor incorporating the Thumb 16-bit compressed instruction set. This microprocessor executes 32-bit or 16-bit instructions and on 32-, 16-, or 8-bit data. The excellent code density achieved with Thumb leads to system cost reductions by reducing the required memory size and achieving 32-bit system performance from 16-bit wide memories.

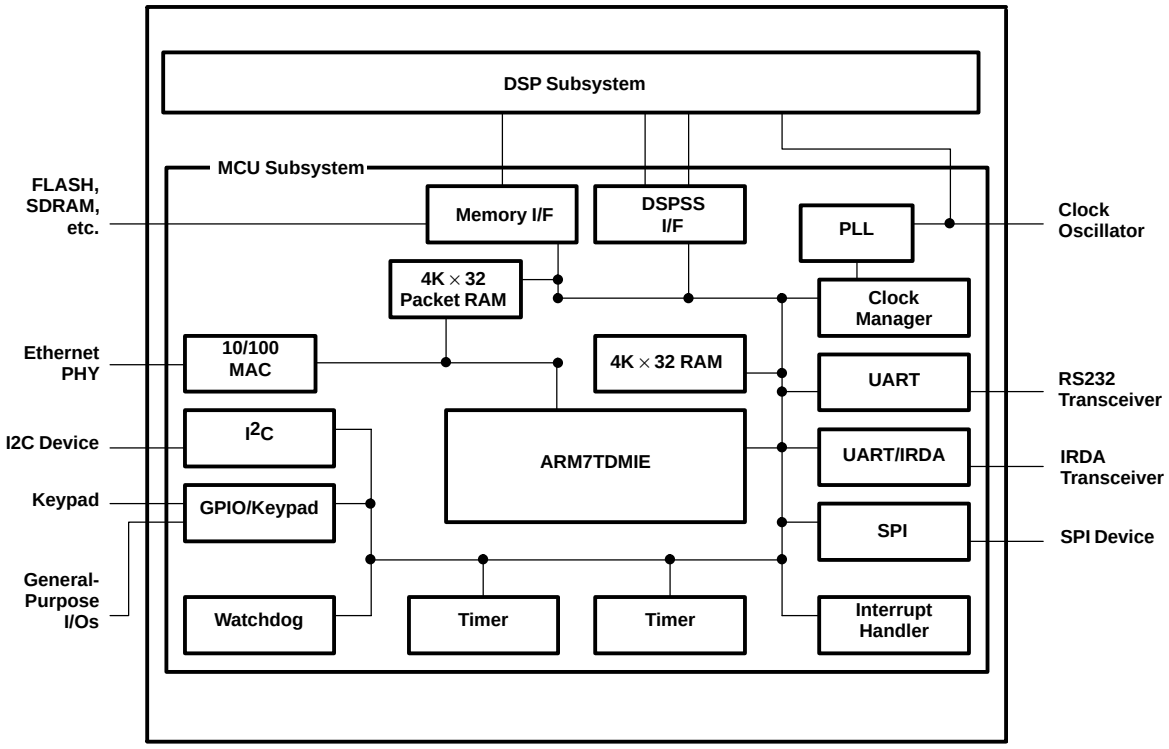


Figure 4–1. 5471 MCU Subsystem Functional Block Diagram

4.1.1 ARM7TDMI Emulation Features

The MCU subsystem's ARM7TDMI core has been enhanced by TI with additional debugging features that are available to emulation tools which understand TI's ARM7TDMIE core. ARM7TDMI emulation tools which are not specifically created with support for the ARM7TDMIE core's emulation extensions are unable to take advantage of these debugging features.

Thumb is a registered trademark of ARM Limited.

The enhanced emulation features provide the following debugging capabilities:

- Single processor (MCU only) and multiprocessor (MCU and DSP) debug
- High-level language and assembly debug (run, halt, step...)
- Real-time (MCU continuously running) or non-real-time (MCU stopped) debug options
- Supports both 32- and 16-bit ARM7TDMI modes
- Endianess transparency
- Unlimited breakpoints via opcode replacement (software breakpoint)
- Two hardware breakpoints (one configurable as software breakpoint) with maskable cycle type, address and data compare
- Two external breakpoint events ($\overline{\text{EMU0}}$, $\overline{\text{EMU1}}$ pins)
- Internal events generate external triggers
- Benchmarking/profiling capability

4.1.2 MCU Memory Space

The MCU memory space includes internal RAM, internal peripherals, and areas for access to external memories and peripherals connected to the $\overline{\text{CS0}}$–$\overline{\text{CS3}}$ and CS4 pins, and external SDRAM. The MCU memory map is summarized in Table 4–1.

Table 4–1. MCU Memory Space

NAME	START ADDRESS	STOP ADDRESS	SIZE IN BYTES	DATA ACCESS
$\overline{\text{CS0}}$	0000:0000	007F:FFFF	8M	8/16/32
$\overline{\text{CS1}}$	0080:0000	00FF:FFFF	8M	8/16/32
$\overline{\text{CS2}}$	0100:0000	017F:FFFF	8M	8/16/32
$\overline{\text{CS3}}$	0180:0000	01FF:FFFF	8M	8/16/32
CS4	0200:0000	027F:FFFF	8M	8/16/32
Reserved	0280:0000	0FFF:FFFF		
SDRAM_$\overline{\text{CS}}$	1000:0000	11FF:FFFF	32M	8/16/32
Reserved	1200:0000	FFBF:FFFF		
Internal SRAM	FFC0:0000	FFC0:3FFF	16K	8/16/32
Reserved	FFC0:4000	FFCF:FFFF		
EIM SRAM	FFD0:0000	FFD0:3FFF	16K	8/16/32
Reserved	FFD0:4000	FFDF:FFFF		
API RAM	FFE0:0000	FFE0:3FFF	16K	16/32
Reserved	FFE0:4000	FFF3:FFFF		
API registers	FFE4:0000	FFE4:0001	2	16
Reserved	FFE4:0002	FFFE:FFFF		
EIM	FFFF:0000	FFFF:07FF	2K	32
UART_IRDA	FFFF:0800	FFFF:0FFF	2K	32
UART	FFFF:1000	FFFF:17FF	2K	32
I2C	FFFF:1800	FFFF:1FFF	2K	32
SPI	FFFF:2000	FFFF:27FF	2K	32
GPIO	FFFF:2800	FFFF:28FF	256	32
KGPIO	FFFF:2900	FFFF:29FF	256	32
Timer0	FFFF:2A00	FFFF:2AFF	256	32
Timer1	FFFF:2B00	FFFF:2BFF	256	32
Timer2	FFFF:2C00	FFFF:2CFF	256	32
INTH	FFFF:2D00	FFFF:2DFF	256	32
MEMINT	FFFF:2E00	FFFF:2EFF	256	32

Table 4–1. MCU Memory Space (Continued)

NAME	START ADDRESS	STOP ADDRESS	SIZE IN BYTES	DATA ACCESS
CLKM	FFFF:2F00	FFFF:2FFF	256	32
SDRAMIF	FFFF:3000	FFFF:30FF	256	32
Reserved	FFFF:3100	FFFF:31FF		
ARM_PLL	FFFF:3200	FFFF:32FF	256	32
Reserved	FFFF:3300	FFFF:FFFF		

4.2 MCU Memory Interface

The MCU memory interface connects the MCU to the internal and external memories and peripherals via a 32-bit-wide data bus. This bus supports MCU access sizes of 8 bits, 16 bits, and 32 bits. All peripheral control registers are implemented as 32-bit devices and should only be accessed using 32-bit operations.

The MCU memory interface allows endian configuration. Generally, a system should be configured so that all peripherals operate in the same endian mode. In cases where mixed endianness is used, the software engineer must carefully control accesses to resources which are configured for the endianness that is opposite to that of the MCU.

The MCU memory interface also provides management of external accesses. External devices supported include ROM (Flash), SRAM, and SDRAM. The external data bus is a 32-bit bidirectional bus. The following access management features for MCU accesses to external memory and peripherals are provided:

- Five external chip-select signals, each with a dedicated 8M-byte range of the MCU memory map.
- MCU access duration management (wait-state insertion) to enable the connection of slow memory devices, and support for intercycle delays to prevent external data bus contention.
- MCU read and write access size adaptation for MCU accesses to external non-SDRAM memory or peripherals with 32-bit, 16-bit, and 8-bit data bus width.
- MCU read and write access size adaptation for MCU accesses to external SDRAM memory with 32-bit or 16-bit data bus width.

The MCU memory ranges associated with each external memory chip-select ($\overline{CS0}$–$\overline{CS3}$, CS4) may be configured in little-endian or big-endian mode. The MCU processor is also configured to operate in big/little endian mode based on the state of “BIGEND” during power-up/reset time. MCU code that accesses a resource that is configured with a different endianness than the MCU must perform the appropriate address calculations (depending on the data size used to write the data, and the data size being used to read the data).

4.2.1 MCU Memory Interface Wait-States

MCU accesses to internal peripherals and internal memories are generally performed at 0 wait state. SDRAM refresh cycles delay any MCU access. MCU accesses to the API RAM require multiple clocks as determined by the programming of the API_REG register.

Timing for MCU accesses to external SDRAM memory is controlled by SDRAM interface registers which control CAS latency, RAS latency, back-to-back timing, refresh rate, etc. External memories or peripherals controlled by $\overline{CS0}$–$\overline{CS3}$ and CS4 may be programmed to provide various cycle timings. Additionally, the $\overline{WAIT}$ input pin may be used to insert wait-states under external hardware control.

4.2.2 MCU API Interface

The API interface, which provides MCU access to a small portion of DSP memory, provides a 16-bit data path to the API RAM in the DSP subsystem. All 32-bit transactions are divided into two 16-bit API transactions. The API interface supports back-to-back access with programmable insertion of wait state to ensure correct synchronization of signals between the MCU subsystem and the DSP subsystem.

4.2.3 MCU SDRAM Memory Interface

The SDRAM memory interface main features are:

- It operates with the MCU memory interface (MEMINT) so that SDRAM memories can be used on the same board with Flash and/or SRAM.
- Supports 16- and 32-bit-wide data bus to SDRAM parts
- Can address up to 256M bits of SDRAM
- Operates at the MCU clock speed (47.5 MHz)
- Flexible programming of SDRAM timing parameters
- Supports four open SDRAM pages
- Minimal burst operations supported. (In 16-bit-wide mode, a burst of two is supported.)

The SDRAM interface must be properly initialized before the SDRAM memory can be used. The following steps are needed for initializing the SDRAM Interface:

- Program SDRAM_REG for data bus size and the number of dummy cycles used after access to slower memory device.
- Program SDRAM parameters in SDRAM_CONFIG, SDRAM_REF_COUNT, and SDRAM_INIT_CONF
- Write a ‘1’ in the INIT bit of the SDRAM_CNTL register
- Wait until a ‘1’ is read inside the READY bit of the SDRAM_CNTL register before accessing the SDRAM.

NOTE: By writing a ‘1’ in the INIT bit of the SDRAM_CNTL register, the initialization state machine of the SDRAM interface module will do the following: Wait for the number of cycles that is programmed in INIT_NOP_MAX_CNT (should be 100 µs or greater). Send the specified number of refresh commands that have been programmed into the INIT_REF_MAX_CNT register.

The SDRAM load-mode-register command is used to configure the SDRAM device mode. Parameters used in the load-mode-register command are based on the values programmed in the SDRAM interface registers.

4.3 MCU Peripherals

The following section describes the 5471 MCU subsystem peripherals.

4.3.1 MCU Ethernet Interface Module

The 5471 Ethernet interface implements an IEEE802.3/Ethernet MAC which supports 10- and 100-Mbit/s data rates, along with buffer memory for Ethernet traffic. The 16-KB memory-mapped packet memory is used for temporary storage of transmit (TX) and receive (RX) packets and various control information. This packet memory is isolated from the MCU subsystem memory bus, so that Ethernet data does not affect MCU subsystem memory access performance. Figure 4–2 is a functional block diagram of the Ethernet interface.

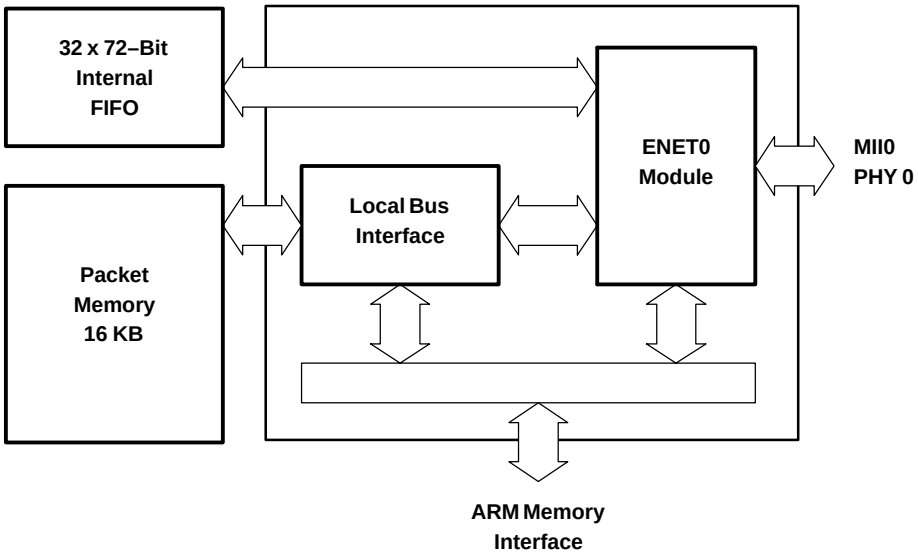


Figure 4–2. Ethernet Interface Block Diagram

The Ethernet interface has two packet channels. One of these channels is associated with the Ethernet media access controller (MAC) port (ENET0), and one virtual port connected to the logical link control layer (sub-layer for the data link layer) of the MCU software (called MCU port). The Ethernet interface handles the packet routing between the two packet channels.

The MAC is a full-featured, configurable 802.3 controller. The half/full duplex configuration is controlled by setting the DUPLEX bit in the mode register. The MAC receive block implements all the required IEEE 802.3 standards for either 10Mbit/s operation or 100Mbit/s operation. The MAC receive block receives data from the MII, handles receive path control, and identification of errors. The MAC transmit block takes the data from the buffer memory and serializes it, and provides it to the Media Independent Interface (MII) port, following the IEEE 802.3 standards requirements for either 10 Mbit/s operation or 100 Mbit/s operation. It implements identification of transmit errors, and transmits the data to the MII.

When the MAC receives an Ethernet packet, the packet information is temporarily stored in a local receive FIFO. The receive FIFO is provided to decouple system memory (packet memory) access timing from packet timing. This prevents overrun conditions caused by MCU accesses to the packet memory. In addition to buffering, the receive FIFO allows for network retries, runts, and flow control. Similarly, there is a transmit FIFO for the MAC. Each of these FIFOs is 32x72, to allow buffering of a full packet plus address and control and status information. The Ethernet interface automatically transfers data between the packet memory and the FIFOs as necessary.

Movement of data between the FIFOs and the packet memory is directed via a data structure in the packet memory called the descriptor rings (DR). In general, the DR data structures provide flexible packet management allowing variable memory buffer sizes and locations. The DR also provides a means to synchronize shared ownership of data buffers in memory between the Ethernet interface and the MCU. Each ring consists of a variable number of descriptors which can be chained to handle long packets in multiple data buffer areas. The location of the descriptor rings in memory is programmable using Ethernet interface configuration registers. One descriptor ring is for transmit operations and the other is for receive operations.

The packet memory provides two circular queues for Ethernet packets: one queue for received packets and one queue for transmit packets for the port (ENET0). Each queue is a list of descriptors which includes control and status information, and a pointer to packet data. When the MCU software identifies a packet which must be moved from one queue to another, only the descriptors (pointing to the packet data) are copied, i.e., data packet is passed by reference. Each descriptor is linked to a packet buffer. To keep track of free buffers, a free buffer entry is added after each packet buffer. This allows a buffer to be referenced by two descriptors of two different queues, which is useful when receiving broadcast or multicast packets.

The Ethernet interface accesses each descriptor circular queue in strict sequential fashion. If the Ethernet interface wants to process a descriptor entry to transmit a packet, it will check the ownership bit. If the MCU owns the entry, the controller will periodically poll the entry waiting for ownership to be passed to the ENET module. If ownership is not passed before the buffer empties, a transmit underflow will occur. Likewise, for receive operations, the controller will poll and wait for the next descriptor entry to become available. If ownership is not passed before the buffer fills, a receive overflow will occur. The user may set the poll time by loading the 16-bit poll interval register, which defines the number of MCU subsystem clocks between polls. The Ethernet interface does not check the length of a chained descriptor. MCU software should avoid creating linked descriptors that violate Ethernet rules.

It is the responsibility of MCU software to initialize the packet memory structure and circular queues, and to ensure coherence with the configuration values programmed in the Ethernet interface registers.

4.3.1.1 EIM Operation

The MCU software has the responsibility to initialize the Ethernet interface module (EIM).

- Packet buffer memory initialization
  - The MCU software has to initialize the descriptor rings structures. It is a requirement that each descriptor points to an existing buffer and that each buffer is only referenced once. Interrupts have to be enabled on all descriptors. All descriptors have to be owned by their own port (OWN=1). Buffer usage entries have to be initialized to zero.
- ENET0 initialization
  - If the ENET module is not used it could be disabled by clearing permanently the ENABLE bit of the MODE register.
  - The mode, backoff seed, backoff count, TX flow go, flow control, VTYPE, ring poll interval registers have to be set according to the operating configuration.
  - The TDBA and RDBA registers have to be set according to the mapping done in the packets memory.
  - The logical address hash filter register has to be initialized to zero.
  - The ENET has to be set in all pass mode in the address mode enable register. If MCU logical addressing is desired, the enable logical address bit has to be set to one as well.
- EIM start-up
  - The ENET has to be started by setting the ENABLE bit in its MODE register.

- Operation
  - MCU software has to manage RX/TX events (by interrupt or polling) on MCU queues. Packets could be processed directly in EIM packet memory or could be transferred to general-purpose MCU memory to release descriptors as soon as possible.
  - If multicast subscribing/unsubscribing is desired on MCU port, the MCU software has to compute a new value of the logical address hash filter and reprogram it on the ENET.
- ENET DMA operations are limited to the 16 KB packet RAM. ENET cannot see the whole MCU memory space. Bits 31–16 of addresses are not significant.
- ENET interrupts are grouped into one EIM interrupt. Interrupt status is accessible through the EIM status register.

4.3.2 MCU Universal Asynchronous Receiver/Transmitter (UART) Interfaces

The UART modules perform serial-to-parallel conversion on data characters received and parallel-to-serial conversion on data characters transmitted by the processor.

4.3.2.1 UART Modes

UART mode supports data transmission and data reception via two 64-word-deep FIFOs. Five different parity types are enabled, with three different stop bit formats. The word length is between 5 and 8, and line break can be generated and detected. The baud rate is generated from the MCU subsystem clock by a programmable divisor. All transmitting parameters can be detected in reception mode by an auto-bauding mechanism that recognizes speed, word size, parity, and the number of stop bits. Software flow control is available in the UART/IrDA when in UART mode. The UART/modem supports both software and hardware flow control when in UART mode. Hardware flow control can reduce the MCU software overhead required to process data interrupts.

UART/Modem Mode

The UART/modem provides an autobauding mechanism. All operations are controllable either via a software interface or using hardware flow control signals.

The UART/Modem provides five external pin signals (multiplexed with GPIO pins):

- GPIO09/TX_MODEM: Output - transmit data
- GPIO10/RX_MODEM: Input - receive data
- GPIO11/CTS_MODEM: Input - clear to send
- GPIO12/RTS_MODEM: Output - request to send
- GPIO13/DCD_MODEM: Output - data carrier detected

The UART/modem features:

- Line-break generation and detection
- Interrupt system control
- Loop-back capabilities for internal test
- The autobauding supports speeds between 1200 and 115.2 Kbits/s.
- Hardware flow control (DCD, RTS, CTS)

UART/IrDA Mode

In the UART/IrDA mode, an Infrared Data Association (IrDA) protocol encoder/decoder manages the RXIR and TXIR signals to an IrDA transceiver. It includes the slow infrared (SIR) protocol in order to be connected with an infrared transmitter to any external data peripherals with an IrDA compliant data interface.

The UART/IrDA provides three external pin signals (multiplexed with GPIO pins):

- GPIO07/TX_IRDA/TXIR_IRDA: Output – transmit data
- GPIO08/RX_IRDA/RXIR_IRDA: Input – receive data
- GPIO06/CLK16X_IRDA/SD_IRDA: Output – shut down the IR transceiver

UART/IrDA IrDA mode features:

- IrDA 1.0 SIR support allows serial communication at baud rates up to 115.2 Kbits/s.
- Sending a single infrared pulse signals a zero. A one is signaled by not sending any pulse. The width of the pulse can be either 1.6 μs or 3/16th of a single bit time.
- In IrDA 1.0 SIR mode, the device operation is similar to the operation in UART mode. However, the data transfer operations are normally performed in half-duplex. The modem control and status signals are not used.
- Frame formatting: addition of variable xBOF characters and EOF characters
- Uplink/downlink CRC generation/detection
- Asynchronous transparency (automatic insertion of break character)
- Eight characters status FIFO available to monitor frame length and frame errors
- Variable frame length for RX and TX IrDA frame.

The format of the serial data is similar to the UART/modem data format. Each data word is sent with a zero value start bit, followed by 8 data bits, and ending with at least one stop bit with a binary value of one.

In SIR mode, data transfer takes place between MCU and peripheral devices at speeds up to 115200 bits/s. A SIR transmit frame consists of start flags (either a single C0h, multiple C0hs, or a single C0h preceded by a number of FFh flags), followed by frame data supplied by the MCU, then a CRC-16 word, and a terminating stop flag (C1h). The transmit state machine attaches start flags, CRC-16, and stop flags to the transmit data provided by the MCU. If necessary, it also applies SIR data transparency.

NOTE: BLR[4] is used to select whether C0h or FFh start patterns are to be used when multiple start flags are required.

SIR transparency is carried out if the outgoing data (between the start and stop flags) contains C0h, C1h, or 7Dh. If one of these characters is about to be transmitted, then the SIR state machine sends an escape character (7Dh) first, then inverts the fifth bit of the real data to be sent, and sends this data immediately after the 7Dh character.

The SIR receiver state machine recovers the receive clock, removes the start flags, removes any transparency from the incoming data, and determines frame boundary with reception of the stop flag. It also checks for errors such as: frame abort (7Dh character followed immediately by a C1h stop flag, without transparency), CRC error, and frame-length error. Once a frame is received, the state machine generates an interrupt to the MCU. The MCU reads the LSR (status register) to check for receive errors, and processes any valid receive data.

Note that the UART/IrDA is capable of operating in full-duplex, but that the SIR standard disallows full-duplex operation.

The infrared output in SIR mode can implement either 1.6 μs or 3/16-bit encoding, which is selected by ACREG[7]. In 1.6-μs encoding, the infrared pulse width is 1.6 μs. In 3/16-encoding, the infrared pulse width is 3/16th of a bit duration (1/baud-rate).

4.3.2.2 FIFO Operation

The transmit FIFO is written via the THR register. Transmit FIFO status is reflected in the LSR.

The receive FIFO is read via the RHR register. The receive FIFO tracks reception of break, framing errors, and parity errors on a byte-by-byte basis. The LSR bits which reflect receive status show the status for the byte that is at the top of the FIFO. Reads of LSR do not advance the Receive FIFO pointer.

LSR[7] is set when there is an error anywhere in the RX FIFO and is cleared only when there are no more errors remaining in the FIFO.

4.3.2.3 UART Clocking

The UART clock divisors must be set based on the MCU subsystem clock rate and the desired baud rate. The UART_DIV_115K register is a prescaler which divides the MCU subsystem clock down to a rate usable by the UART state machines. The UART_DIV_BITRATE register is the divider which provides the bit clock to the UART from the pre-scaled clock.

UART_DIV_BITRATE = MCU clock frequency / UART_DIV_115K / desired bit rate

For autobaud function on the UART/modem, the UART_DIV_115K value must result in a 115200-Hz clock rate:

UART_DIV_115K = MCU clock frequency / 115200 Hz.

4.3.3 MCU Serial Peripheral Interface (SPI)

The SPI is a bidirectional 3-line interface dedicated to the transfer of data to and from external devices offering a 3-line serial interface.

The SPI interface is fully-duplexed and is configurable from 1 to 32 bits, providing three enable signals programmable either as positive or negative edge- or level-sensitive.

This serial port is based on a looped shift-register which allows for both transmit (parallel-in, serial-out) and receive (serial-in, parallel-out) modes. The serial port is fully controlled by the MCU memory interface (data write, data read, and activation of serialization operations).

The serial clock period (T_{CLKX_SPI}) is derived from the MCU subsystem clock, based on a prescale divider (PTV):

SPI Clock Rate = MCU Clock Rate / (4 * PTV)

where PTV is 2, 4, 8, or 16.

4.3.4 MCU General-Purpose I/O

The 5471 provides 36 general-purpose I/Os (GPIOs) configurable in read or write mode by internal registers. The GPIOs are divided into two groups, GPIO(19:0) and KBGPIO(15:0). KBGPIOs are keyboard GPIO pins, which are implemented similarly to the GPIO(19:0) pins, although some KBGPIO pins have internal pullup resistors. Some of the GPIO and KBGPIO pins share functionality with signals from other modules. Register bits configure whether the pin is used for the normal GPIO or KBGPIO functionality, or for the alternate functionality.

Each GPIO is associated with 6 configuration/status bits whose description is given in Table 4–2 and Table 4–3.

Table 4–2. GPIO Control/Status Bits

BIT NAME	DESCRIPTION
io	I/O bit: Writable when I/O is configured as an output (cio=0) Reads value on I/O pin when I/O is configured as an input (cio=1)
cio	Configure I/O: 0: output 1: input (default)
gpio_irqA	See Table 4–3
gpio_irqB	See Table 4–3
ddio	Delta detect bit: If gpio is configured as an output (cio=0), always reads as '0' If gpio is configured as an input (cio=1), reads a '1' if io has changed since ddio was last cleared
gpio_en	Selects register for muxed GPIOs: 0: other signal 1: gpio (default) Non-shared GPIOs are always available at the i/o pin independent of the value of gpio_en

Table 4–3. GPIO_IRQ Bits Definition

gpio_irqB	gpio_irqA	FUNCTION
0	0	The associated device pin does not cause assertion of a GPIO interrupt.
0	1	A rising edge on the associated device pin causes assertion of a GPIO interrupt.
1	0	A falling edge on the associated device pin causes assertion of a GPIO interrupt.
1	1	Any change on the associated device pin causes assertion of a GPIO interrupt.

The configuration/status bits are accessible through 12 memory-mapped registers: GPIO_IO, KBGPIO_IO, GPIO_CIO, KBGPIO_CIO, GPIO_IRQA, KBGPIO_IRQA, GPIO_IRQB, KBGPIO_IRQB, GPIO_DDIO, KBGPIO_DDIO, GPIO_EN and KBGPIO_EN.

The pins KBGPIO(15:8) have on-chip pullup resistors, making them suitable for use as keypad row inputs. To configure the 5471 for an 8x8 keypad [with keypad columns connected to KBGPIO(7:0) and rows connected to KBGPIO(15:8)], use the following settings:

- KBGPIO(15:8) pins are configured as inputs (cio=1) for row lines
- KBGPIO(7:0) pins are configured as outputs (cio=0) for column lines

Keypad scanning can be done by setting the output values on all of the column KBGPIO pins to 0 and enabling KBGPIO(15:8) interrupts. When a KBGPIO(15:8) interrupt is seen, all but one of the the KGBPIO column output values are be changed to 1, and then the KBGPIO row pins are read. If the KBGPIO pin row input value is all 1s, then there are no keys in the column currently driven to 0 that are set. Any bit in the KGBPIO pin row input value that is 0 indicates a closed switch. Next, select a different column by writing a new value to KBGPIO column pins with just one output pin value 0 and the remainder 1. Repeat the KBGPIO row read and check for closed switches. Repeat this for each column in the keyboard array. Once the keypad has been read, the interrupt service routine should set all column outputs back to 0 so that the next key press may be sensed.

4.3.5 MCU Inter-Integrated Circuit (I^2C) Interface

The master I^2C interface module provides an interface between the MCU subsystem bus and the I^2C pins, allowing the MCU to control external peripheral devices connected to the I^2C pins. Fundamentally, the I^2C interface module is a parallel-to-serial and serial-to-parallel converter. The parallel data received from the MCU for transmit has to be converted to a suitable serial form for external peripheral devices on the I^2C Bus™. Conversely, the serial data received from the I^2C bus has to be converted to a suitable parallel form for passing to the MCU.

The I^2C interface features are:

- Operates as an I^2C master only; cannot be configured as an I^2C slave
- Supports 7-bit I^2C device address
- Supports an 8-bit I^2C device subaddress
- Master write to slave receiver in single or multiple mode (data loop)
- Provides a 16-byte transmit FIFO to reduce MCU overhead
- Supports simple read cycle, read combined cycle
- A 3-bit programmable prescale internal clock divider and 7-bit programmable SCL clock divider to support a wide range of input clock frequencies. The I^2C SCL clock frequency are:
  - I^2C Standard Mode: 100 kHz
  - I^2C Fast Mode: 400 kHz
- 3-bit programmable spike filter to provide noise filtering on the I^2C data input signal

The module does not support:

- I^2C bus 10-bit addressing
- I^2C bus CBUS compatibility
- Multi-Master I^2C
- Clock synchronization as a handshake: slave devices are NOT allowed to hold the SCL line LOW to force the master (5471) into a wait state until the slave is ready for the next transfer.

4.3.6 MCU Timers

The 5471 MCU subsystem implements three 16-bit timers configurable either in “auto reload” or in “count down to zero and stop” modes. Each timer can generate an interrupt to the MCU when it counts down to zero.

The TIMER0 may be configured as either a watchdog counter or as a general-purpose timer. The two others (TIMER1 and TIMER2) are general-purpose timers.

TIMER2 may be used to provide timing to an external device via the shared pin GPIO19/TIMER_OUT. When this pin is configured to provide TIMER_OUT, the pin will provide a low pulse when the TIMER2 interrupt occurs.

4.3.6.1 General-Purpose Timers

TIMER1 and TIMER2, are general-purpose timers, and can be programmed to provide specific timing via a clock prescale value (PTV) and a starting count value (LOAD_TIM). The timer counts down to zero from the LOAD_TIM value, changing value every n MCU subsystem clocks, depending on the PTV value. The time between interrupts to the MCU interrupt handler is defined as:

$$t_{int} = t_{clk} * (LOAD\_TIM + 1) * 2^{(PTV + 1)}$$

If the counter’s auto-reload (AR) bit is set when the counter reaches zero, the timer will auto-reload the LOAD_TIM value into its counter and start counting again.

The counter will only run when its start (ST) bit is set. The PTV and AR bits of the CNTL_TIMER register and LOAD_TIM must not be programmed when the timer is running. The TIMERn current counter value may be read via the VALUE_TIMn register.

I^2C Bus is a trademark of Philips Electronics N. V. Corporation.

4.3.6.2 TIMER0 as a Watchdog Timer

By default, TIMER0 is configured as a watchdog timer. The watchdog is designed to detect user programs stuck in infinite loops resulting in loss of program control or “runaway” programs. When configured as a watchdog counter, TIMER0 will reset the MCU subsystem if it counts down to zero. The MCU’s reset initialization code may examine the WATCHDOG_STAT register to determine if a MCU reset was caused by the Watchdog.

The user program should write periodically into LOAD_TIM register before the counter reaches 0 in order to reload the timer with this new value. A LOAD_TIM write is taken into account only if the new loaded value is different from the previous one. When the watchdog counter reaches 0, it resets the MCU core and all the MCU subsystem peripherals.

On power up, the watchdog is enabled and the value loaded into LOAD_TIM register is set to the maximum value (0xFFFF) and PTV is set to seven. This gives to user a time of 16,777,216 * t_{clk} to switch to normal timer mode or to set a different watchdog timer time out to Timer0.

If a programmer does not want to have this default watchdog functionality, he must write a specific sequence into a dedicated register (TIMER_MODE) in order to configure it as a general-purpose timer.

Disabling TIMER0 Watchdog Function

Timer0 may be configured as a general-purpose timer by writing a predefined sequence (0xF5 followed by 0xA0) in the WATCHDOG_DIS bits of the CNTL_TIMER0 register. If the watchdog disable function sees a write to CNTL_TIMER0 with WATCHDOG_DIS set to 0xF5, it begins watching for another write to CNTL_TIMER0 with WATCHDOG_DIS set to 0xA0. If WATCHDOG_DIS in the next write is 0xA0, TIMER0 is set for general-purpose mode. If the next write is instead 0xF5, TIMER0 remains in watchdog mode and the disable function begins watching again for 0xF5.

Re-enabling TIMER0 Watchdog Function

To switch TIMER0 from general-purpose timer mode to watchdog timer mode, write a ‘1’ in the WATCHDOG bit of the CNTL_TIMER0 register. When WATCHDOG is set to 1, the LOAD_TIM value is forced to 0xFFFF. Due to internal sequencing, the MCU code should wait at least three timer clock periods before re-initializing LOAD_TIM or PVT to values that meet system watchdog timing needs.

4.3.7 MCU Interrupt Handler

The MCU subsystem interrupt handler prioritizes and masks interrupts from up to 16 MCU subsystem interrupt sources (IRQ0–15). It also steers these interrupts to the MCU’s two interrupt inputs: $\overline{\text{ARM\_IRQ}}$ (low-priority interrupt request) and $\overline{\text{ARM\_FIQ}}$ (fast interrupt request). It receives interrupts from both internal modules and the external chip environment. External interrupts to the MCU subsystem interrupt handler may be provided through GPIO and/or KBGPIO pins.

Each interrupt source has an interrupt level register (ILR) which defines the priority of the corresponding interrupt. When multiple interrupts are pending, the interrupt with the highest-priority interrupt level register value is sent to the MCU first. In the case of multiple pending interrupts with identical ILR priority values (with no other pending interrupts with higher priority), the highest interrupt number is sent to the MCU first.

Each interrupt source can be individually routed to one of the two input interrupt lines of the MCU, by properly setting the FIQ bit in the interrupt’s ILR_IRQ_n register.

The following paragraph lists the typical IRQ sequence. The FIQ sequence is similar.

- As the interrupt handler sees unmasked interrupt inputs go active, it sets the corresponding IT_REG bit.
- At this time, there are three possible cases:
  - If the NEW_INT_AGR bit in the INT_CTRL_REG is set, a previous interrupt has not yet been completely serviced, and the new interrupt cannot be sent to the MCU.
  - If NEW_INT_AGR is 0 and there is only one interrupt marked as active in the IT_REG, that interrupt number is selected as the interrupt to be passed to the MCU. This selected interrupt number is stored in SIR_IRQ_REG.
  - If NEW_INT_AGR is 0 and there are more than one interrupt marked as active in the IT_REG, the interrupt handler selects the highest-priority interrupt based on ILR_IRQ_n register priority values and, where priority values are identical, interrupt numbers. This selected interrupt number is stored in SIR_IRQ_REG.
- The interrupt is sent to the MCU via its IRQ port.
- The MCU begins execution at the IRQ vector. MCU code reads SIR_IRQ_REG to determine which interrupt source caused the interrupt. MCU code branches to the appropriate Interrupt Service Routine.
- The MCU software must write a “1” to the INT_CTRL_REG register NEW_IRQ_AGR bit in order to reset the IRQ output and SIR_IRQ_REG register. This enables the interrupt handler to send new interrupts to the MCU via the IRQ signal.

MCU subsystem interrupt numbers are shown in Table 4–4.

Table 4–4. MCU Peripheral Interrupt Mapping

IRQ REQUEST	MCU INTERRUPT SOURCE
IRQ0	Watchdog TIMER0 interrupt
IRQ1	TIMER1 interrupt
IRQ2	TIMER2 interrupt
IRQ3	GPIO0 interrupt
IRQ4	Ethernet interface interrupt
IRQ5	KBGPIO(7:0) interrupt
IRQ6	UART interrupts
IRQ7	UART_IRDA interrupt
IRQ8	KBGPIO(15:8) interrupt
IRQ9	GPIO03
IRQ10	GPIO02
IRQ11	I^2C interrupt
IRQ12	GPIO1 interrupt
IRQ13	SPI interrupt
IRQ14	GPIO(19:4) interrupt
IRQ15	API interrupt

4.4 MCU Power-Down Modes

The MCU can be shut down and awaken by setting the appropriate bit into the CLKM_REG and WAKEUP_REG. It is the software’s responsibility to put the ARM_CLOCK in bypass or low-frequency mode before the MCU is shut down. The MCU is woken by the first interrupt received from one of the peripherals.

The MCU subsystem peripherals can be individually shut down and awaken by setting the corresponding bits in the CLKM_REG and WAKEUP_REG register. It is the software’s responsibility to power down or wake up individual peripherals.

4.5 MCU Peripheral Clock Management

The 5471 device is clocked by the external clock input, REFCLK. The 5471 REFCLK input does not provide a clock oscillator, so it must be driven by a square-wave input signal that meets V_{IH} and V_{IL} requirements. Both subsystems (DSP and MCU) derive their clocks from REFCLK using software-programmable PLLs. Immediately following reset or power up, the contents of the respective CLKMD registers for both subsystems depend on the status of the programmable internal ports on their respective PLLs. For the MCU subsystem, these internal ports are hardwired so that the programmable capability of default mode is not available. For the DSP subsystem, the programmable internal ports to the DSP PLL are connected to the output of a register (DSP_REG) that is controlled by the RISC processor. This allows the DSP PLL default values to be programmable under the control of the MCU subsystem. The clock management scheme is shown in Figure 4–3.

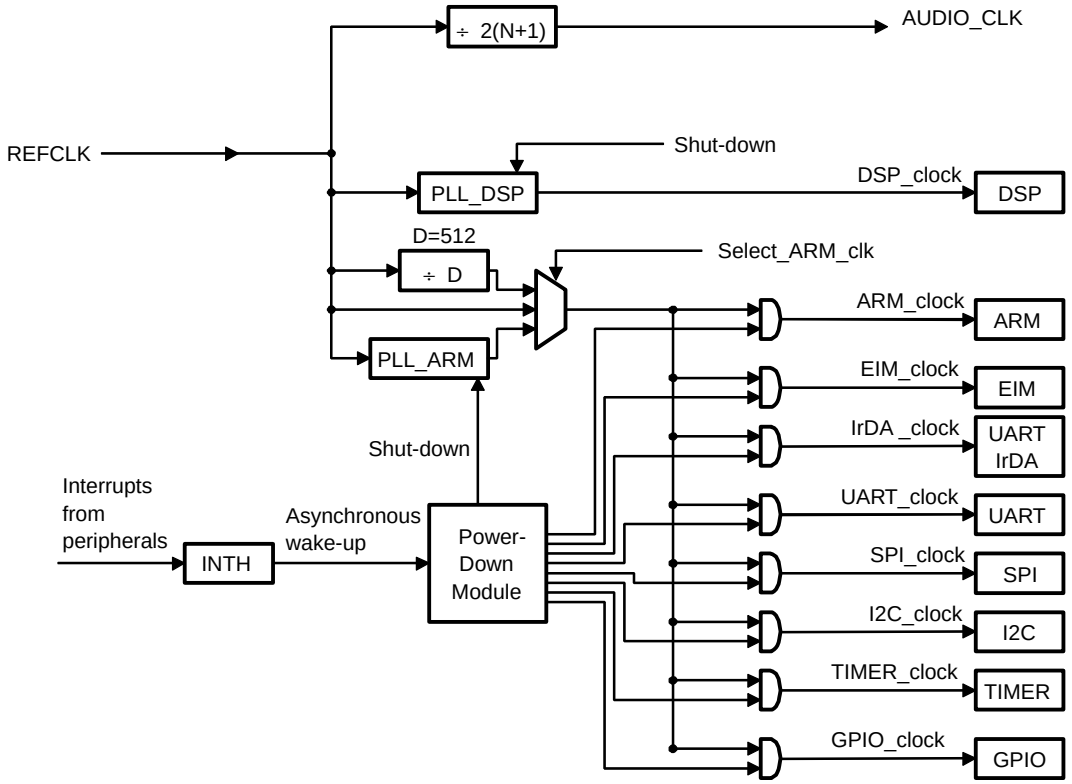


Figure 4–3. Clock Management Module

The MCU subsystem operates in one of three modes: PLL (normal) mode, DIV mode, or power-down mode. In power-down mode, the input clock frequency is divided by a large value (512 – 1023). Configuring the operation of the MCU subsystem clock module in power-down mode and using divisor values below 512 will result in stopping the clock.

Similar to the DSP clock, the MCU subsystem clock is also derived from the scaled version of the input clock, REFCLK. The PLL used by the RISC has the same look and feel as the DSP PLL. For this reason, the RISC processor uses the same minimum input-clock-frequency restriction and scaling values as the DSP.

4.6 Initialization

The RESET input is the master reset that resets both the DSP and MCU subsystems. The RESET_OUT signal can be used to reset external peripherals under control of the MCU. Table 4–5 describes how reset events are propagated to the MCU subsystem, the DSP subsystem, and the RESET_OUT pin.

Table 4–5. Reset Management

Activated Reset Signal	MCU Reset	DSP Reset	External Reset (RESET_OUT)
RESET	Yes	Yes	Yes
RST_CMD watchdog	Yes	No	No
DSP software reset (in register CLKM_CNTL_RST)	No	Yes	No
RESET_OUT (software controlled by bit in register CLKM_CNTL_RST)	No	No	Yes

CLKM_CNTL_RST is a control register mapped in the MCU memory space at address 0xFFFF2F10.

When the MCU subsystem is reset, the MCU program counter begins execution at address 0x00000000.

When the DSP subsystem is reset, the DSP program counter begins execution from 0xFF80.

Each MCU subsystem peripheral may be independently held in reset via control bits in the MCU subsystem register RESET_REG. By default, these bits are set as soon as the MCU subsystem reset is activated, which leaves all MCU subsystem peripherals active.

4.6.1 Initial MCU Code

Since the MCU begins execution from address 0x00000000 after MCU subsystem reset, the initialization code for the MCU subsystem must be provided by the memory device connected to the CS0 pin. This code may be either 32-bit or 16-bit (Thumb) ARM code. The value of the CS3/ROMSIZE16 pin is sampled at the rising edge of RESET to determine the mode that the ARM7TDMI core will execute.

CS3/ROMSIZE16 INPUT VALUE AT RESET RISING EDGE	ARM7TDMI INITIAL MODE
0	16-bit instruction (Thumb) mode
1	32-bit instruction mode

Similarly, the MCU subsystem initializes the ARM7TDMI for big-endian or little-endian operation based on the value sampled on the CS4_BIGEND pin at the rising edge of the RESET signal.

CS4/BIGEND INPUT VALUE AT RESET RISING EDGE	ARM7TDMI INITIAL MODE
0	Little-endian
1	Big-endian

4.6.2 DSP Boot Mode

When the DSP comes out of reset, it begins execution from 0xFF80. It can execute code either from external memory or from internal memory, depending on the value of the DSP_MPNMC bit in the MCU's DSP_PLL_REG register. Once the appropriate mode is set, and, if necessary, code is downloaded via the API module, the MCU may release the DSP subsystem from reset by clearing the DSP_RESET bit in the MCU subsystem's CLKM _CNTL _RST register.

For the case where DSP execution starts from internal memory, the MCU should download the appropriate DSP code via the API interface (before releasing the DSP from reset). This is done by properly setting the DSP_MPNMC and DSP_APIBN bits in the DSP_PLL_REG MCU memory-mapped register, and then downloading the code via the API interface. The DSP_APIBN bit, when 0, causes the 2K-word block of DSP data memory at 0x3800–0x3FFF to appear also in DSP program space beginning at 0xF800.

Table 4–6 shows the combinations of the DSP_PLL_REG register DSP_MPNMC and DSP_APIBN bits that control where the DSP fetches code immediately following release of DSP reset.

Table 4–6. DSP Boot Memory

DSP_MPNMC	DSP_APIBN	DSP BOOT MEMORY AT DSP PROGRAM SPACE ADDRESS 0xFF80
0	0	Shadow of the API-accessible on-chip data-space RAM
0	1	On-chip program-only RAM which is not accessible to the API module
1	0	Shadow of the API-accessible on-chip Data-space RAM
1	1	External DSP memory

4.6.3 Emulation Support

The 5471 provides flexible emulation capabilities. It supports both TI's emulation tools and tools developed by third parties.

Since the emulation tools may understand both the C54x™ DSP core and the ARM7TDMIE core, or just the C54x™ DSP core, just the ARM7TDMIE core, or just the ARM7TDMI core, multiple emulation modes are required. To allow configuration of the internal JTAG chains used by the various emulators, the 5471 samples the $\overline{\text{EMU1}}$ and $\overline{\text{EMU0}}$ pins at $\overline{\text{TRST}}$ rising edge. Based on the values of these pins, the internal scan chain is configured for different emulation modes. These modes are summarized in Table 4–7.

Table 4–7. Emulation Mode Selection

Values at $\overline{\text{TRST}}$ Rising Edge		SCAN CHAIN ORDER
$\overline{\text{EMU0}}$	$\overline{\text{EMU1}}$	
0	0	Only the ARM7TDMIE core is in the chain. (This is also the mode for tools which only understand the ARM7TDMI core.)
0	1	Only the C54x™ DSP core is in the chain
1	0	Reserved
1	1	ARM7TDMIE core followed by C54x™ DSP core followed by ASIC Logic test access port (TAP) controller

Some tools that only understand one type of processor may be configured for use with 5471 even when other devices are on the ASIC scan chain. This is done by instructing the tools to place the other “unrecognized” devices on the JTAG scan chain in their “BYPASS” modes. While the exact configuration methods are beyond the scope of this document, the key is to understand the length of the instruction registers for the TAP controllers so that the correct BYPASS instructions can be issued. For the 5471, this methodology would only apply to the 5471 emulation mode where all three TAP controllers are in the JTAG scan chain ($\overline{EMU1}$ = $\overline{EMU0}$ = ’1’ at $\overline{TRST}$ rising edge). The JTAG TAP controller instruction register lengths are listed in Table 4–8.

Table 4–8. JTAG TAP Controller Instruction Register Lengths

INSTRUCTION REGISTER LENGTH	JTAG TAP CONTROLLER
4 bits	ARM7TDMIE core
8 bits	C54x™ DSP core
8 bits	ASIC Logic TAP Controller

5 Documentation Support

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

- *TMS320VC547x CPU and Peripherals Reference Guide* (literature number SPRU038)
- *TMS320C54x™ DSP Functional Overview* (literature number SPRU307)
- Device-specific data sheets
- Complete user's guides
- Development support tools
- Hardware and software application reports

The five-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)
- *Volume 5: Enhanced Peripherals* (literature number SPRU302)

The reference set describes in detail the TMS320C54x™ DSP products currently available and the hardware and software applications, including algorithms, for fixed-point TMS320™ DSP family of devices.

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

TMS320 and C5000 are trademarks of Texas Instruments.

6 Electrical Specifications

This section provides the absolute maximum ratings and the recommended operating conditions for the device.

6.1 Absolute Maximum Ratings

The list of absolute maximum ratings are specified over operating case temperature. Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability. All supply voltage (core and I/O) values are with respect to V_{SS}.

Supply voltage I/O range, DV_{DD} . . . . . –0.5 V to 4 V
Supply voltage core range, CV_{DD} . . . . . –0.5 V to 2 V
Input voltage range, V_I . . . . . –0.5 V to 4.1 V
Output voltage range, V_O . . . . . –0.5 V to 4.1 V
Operating case temperature range, T_C (Commercial) . . . . . 0°C to 85°C
T_C (Industrial) . . . . . –40°C to 85°C
Storage temperature range, T_{stg} . . . . . –65°C to 150°C

6.2 Recommended Operating Conditions

			MIN	NOM	MAX	UNIT
DV_{DD}	Device supply voltage, I/O (Level shifter)†		3	3.3	3.6	V
CV_{DD}	Device supply voltage, core†		1.71	1.8	1.95	V
V_{SS}	Supply voltage, GND			0		V
V_{IH}	High-level input voltage, I/O		0.6*DV_{DD}			V
V_{IL}	Low-level input voltage				0.3*DV_{DD}	V
I_{OH}	High-level output current	DSP_R/W, DSP_PS, DSP_IS, DSP_DS, DSP_IOSTRB, DSP_MSTRB, DSP_A[19:0], DSP_D[15:0], KBGPIO01/DSP_XF, KBGPIO10/DSP_IACK, KBGPIO14/DSP_CLKOUT			–8	mA
		All other outputs not including power/ground/DSP signals.			–4	
I_{OL}	Low-level output current	DSP_R/W, DSP_PS, DSP_IS, DSP_DS, DSP_IOSTRB, DSP_MSTRB, DSP_A[19:0], DSP_D[15:0], KBGPIO01/DSP_XF, KBGPIO10/DSP_IACK, KBGPIO14/DSP_CLKOUT			8	mA
		All other outputs not including power/ground/DSP signals.			4	
T_C	Operating case temperature	Commercial	0		85	°C
		Industrial	–40		85	

† Texas Instrument DSPs do not require specific power sequencing between the core supply and the I/O supply. However, systems should be designedto ensure that neither supply is powered up for extended periods of time if the other supply is below the proper operating voltage. Excessive exposure to these conditions can adversely affect the long-term reliability of the devices. System-level concerns such as bus contention may require supply sequencing to be implemented. In this case, the core supply should be powered up at the same time as or prior to the I/O buffers, and then powered down after the I/O buffers.

6.3 Electrical Characteristics Over Recommended Operating Case Temperature (Unless Otherwise Noted)

PARAMETER			TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		I_{OH} = RATED	0.8*DV_{DD}			V
V_{OL}	Low-level output voltage		I_{OL} = RATED			0.22*DV_{DD}	V
I_{IZ}	Input current for outputs in high impedance	CS4/BIGEND	DV_{DD} MAX, V_I = V_{SS} to DV_{DD}	–300		10	µA
		$\overline{EMU}$[1:0], GPIO16/SDA, $\overline{CS3}$/ROMSIZE16, KBGPIO12/DSP_MSC, KBGPIO13/$\overline{DSP\_TOUT}$, KBGPIO11/$\overline{DSP\_IAQ}$, KBGPIO10/DSP_IACK, KBGPIO14/DSP_CLKOUT, KBGPIO15/ARM_MCLK, KBGPIO[9:8], DSP_D[15:0], DSP_A[19:0]	DV_{DD} MAX, V_I = V_{SS} to DV_{DD}	–10		310	
		All other inputs	DV_{DD} MAX, V_I = V_{SS} to DV_{DD}	–10		10	
I_I	Input current	TDI, TMS	DV_{DD} MAX, V_I = V_{SS} to DV_{DD}	–10		310	µA
		CRS0, COL0, TCLK0, RXER0, RCLK0, RXDV0, RXD0[3:0]		–10		110	
		$\overline{TRST}$, TCK		–300		10	
		All other input-only pins		–10		10	
I_{DDC}	Supply current, core MCU (CV_{DD})		CV_{DD} MAX, DV_{DD} MAX, $f_{(DSP\_CLKOUT)}$ = 100 MHz, $f_{(ARM\_MCLK)}$ = 47.5 MHz, T_C = 25 °C		70†		mA
I_{DDP}	Supply current, I/Os (DV_{DD})		CV_{DD} MAX, DV_{DD} MAX, $f_{(DSP\_CLKOUT)}$ = 100 MHz, $f_{(ARM\_MCLK)}$ = 47.5 MHz, T_C = 25 °C		50†		mA
C_i	Input capacitance				5		pF
C_o	Output capacitance				5		pF

† The current measurements were taken with the following nominal conditions: CV_{DD} = 1.8V, DV_{DD} = 3.3V, temperature = 25C, with the DSP executing program code from internal SRAM consisting of 50% NOP / 50% MACD instructions @ 100 MHz, and the ARM executing dhrystone program code from external SRAM @ 47.5 MHz. Actual operating current varies with program being executed and external pin usage.

6.4 Package Thermal Resistance Characteristics

Table 6–1 provides the thermal resistance characteristics for the recommended package type used on the TMS320VC5471 DSP.

Table 6–1. Thermal Resistance Characteristics

PARAMETER	GHK PACKAGE	UNIT
$R_{\Theta JA}$	34	°C/W
$R_{\Theta JC}$	9.5	°C/W

6.5 Timing Parameter Symbology

Timing parameter symbols used are created in accordance with JEDEC Standard 100. To shorten the symbols, some of the pin names and other related terminology have been abbreviated as follows:

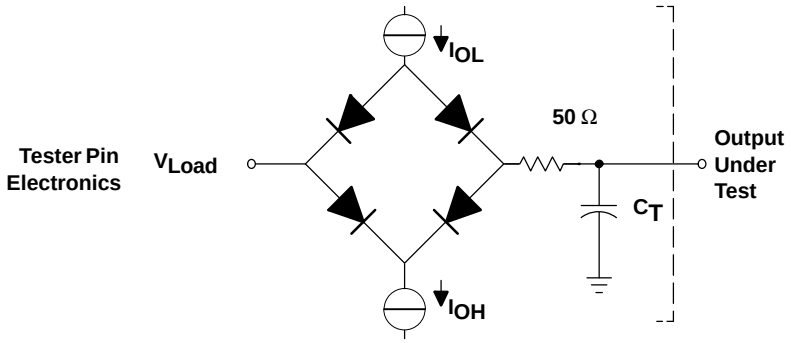
Lowercase subscripts and their meanings:

a	access time
c	cycle time (period)
d	delay time
dis	disable time
en	enable time
f	fall time
h	hold time
r	rise time
su	setup time
t	transition time
v	valid time
w	pulse duration (width)
X	Unknown, changing, or don't care level

Letters and symbols and their meanings:

H	High
L	Low
V	Valid
Z	High impedance

The following load circuit in Figure 6–1 was used on all outputs pins and I/O pins in input mode. All timing measurements in this data manual were measured from the 5471 connection to the following load circuit.



Where: I_{OL} = 8 mA (all outputs)
I_{OH} = 8 mA (all outputs)
V_{Load} = 1.65 V
C_T = 13-pF typical load circuit capacitance

Figure 6–1. 3.3-V Test Load Circuit

6.5.1 Divide-By-Two/Divide-By-Four Clock Option Timing

The frequency of the reference clock provided at the REFCLK pin can be divided by a factor of two or four to generate the internal machine cycle. The selection of the clock mode is described in Section 3.4.7.

NOTE: The frequency injected must conform to specifications listed in Table 6–2.

Table 6–2 and Table 6–3 assume testing over recommended operating conditions and H = $0.5t_{c(CO)}$ (see Figure 6–2).

Table 6–2. Divide-By-Two/Divide-By-Four Clock Option Timing Requirements

		MIN	MAX	UNIT
$t_{c(CI)}$	Cycle time, REFCLK	20	†	ns
$t_{f(CI)}$	Fall time, REFCLK		8	ns
$t_{r(CI)}$	Rise time, REFCLK		8	ns
$t_{w(CIL)}$	Pulse duration, REFCLK low	5		ns
$t_{w(CIH)}$	Pulse duration, REFCLK high	5		ns

† This device utilizes a fully static design and therefore can operate with $t_{c(CI)}$ approaching ∞. The device is characterized at frequencies approaching 0 Hz.

Unless otherwise noted, the same switching characteristics that apply to DSP_CLKOUT also apply to ARM_MCLK.

Table 6–3. Divide-By-Two/Divide-By-Four Clock Option Switching Characteristics

PARAMETER		MIN	TYP	MAX	UNIT
$t_{c(CO)}$	Cycle time, DSP_CLKOUT	40	$2t_{c(CI)}$	†	ns
$t_{d(CIH-CO)}$	Delay time, REFCLK high to DSP_CLKOUT high/low	4	10	17	ns
$t_{f(CO)}$	Fall time, DSP_CLKOUT		2		ns
$t_{r(CO)}$	Rise time, DSP_CLKOUT		2		ns
$t_{w(COL)}$	Pulse duration, DSP_CLKOUT low	H–2	H	H + 2	ns
$t_{w(COH)}$	Pulse duration, DSP_CLKOUT high	H–2	H	H + 2	ns

† This device utilizes a fully static design and therefore can operate with $t_{c(CI)}$ approaching ∞. The device is characterized at frequencies approaching 0 Hz.

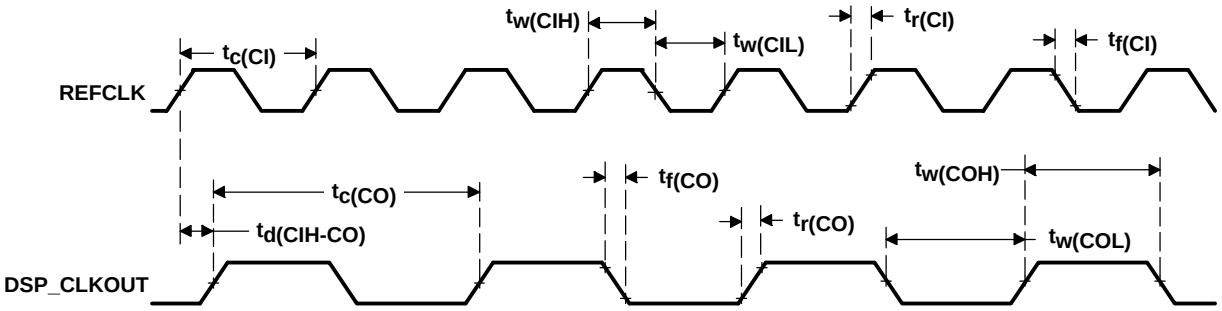


Figure 6–2. External Divide-by-Two Clock Timing

6.5.2 Multiply-By-N Clock Option (PLL Enabled) Timing

The frequency of the reference clock provided at the REFCLK pin can be multiplied by a factor of N to generate the internal machine cycle. The selection of the clock mode and the value of N is described in Section 3.4.7.

NOTE: The external frequency injected must conform to specifications listed in the Table 6–4.

Table 6–4 and Table 6–5 assume testing over recommended operating conditions and H = $0.5t_{c(CO)}$ (see Figure 6–3).

Table 6–4. Multiply-By-N Timing Requirements

			MIN	MAX	UNIT
$t_{c(CI)}$	Cycle time, REFCLK	Integer PLL multiplier N (N = 1–15)†	20‡	200	ns
		PLL multiplier N = x.5†	20‡	100	
		PLL multiplier N = x.25, x.75†	20‡	50	
$t_{f(CI)}$	Fall time, REFCLK			8	ns
$t_{r(CI)}$	Rise time, REFCLK			8	ns
$t_{w(CIL)}$	Pulse duration, REFCLK low		5		ns
$t_{w(CIH)}$	Pulse duration, REFCLK high		5		ns

† N = Multiplication factor

‡ The multiplication factor and minimum REFCLK cycle time should be chosen such that the resulting DSP_CLKOUT cycle time is within the specified range ($t_{c(CO)}$). Minimum $t_{C(CI)}$ for the ARM CPU is 21 ns.

Unless otherwise noted, the same switching characteristics that apply to DSP_CLKOUT also apply to ARM_MCLK.

Table 6–5. Multiply-By-N Switching Characteristics

PARAMETER		MIN	TYP	MAX	UNIT
$t_{c(CO)DSP}$	Cycle time, DSP_CLKOUT	10	$t_{C(CI)}/N$†		ns
$t_{c(CO)ARM}$	Cycle time, ARM_MCLK	21	$t_{C(CI)}/N$†		ns
$t_{d(CI-CO)}$	Delay time, REFCLK high/low to DSP_CLKOUT high/low	4	10	17	ns
$t_{f(CO)}$	Fall time, DSP_CLKOUT		2		ns
$t_{r(CO)}$	Rise time, DSP_CLKOUT		2		ns
$t_{w(COL)}$	Pulse duration, DSP_CLKOUT low	H – 2	H	H + 2	ns
$t_{w(COH)}$	Pulse duration, DSP_CLKOUT high	H – 2	H	H + 2	ns
t_p	Transitory phase, PLL lock up time			30	µs

† N = Multiplication factor

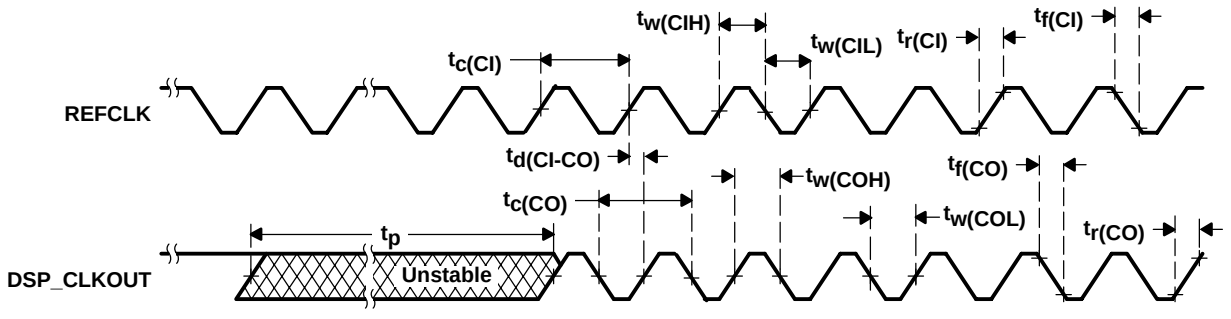


Figure 6–3. External Multiply-by-One Clock Timing

6.6 Memory and Parallel I/O Interface Timing

6.6.1 Memory Read

Table 6–6 and Table 6–7 assume testing over recommended operating conditions with $\overline{\text{DSP\_MSTRB}}$ = 0 and H = 0.5$t_{c(CO)}$ (see Figure 6–4).

Table 6–6. Memory Read Timing Requirements†

		MIN	MAX	UNIT
$t_{a(A)M}$	Access time, read data access from address valid		2H–15‡	ns
$t_{a(MSTRBL)}$	Access time, read data access from $\overline{\text{DSP\_MSTRB}}$ low		2H–15‡	ns
$t_{su(D)R}$	Setup time, read data before DSP_CLKOUT low	4		ns
$t_{h(D)R}$	Hold time, read data after DSP_CLKOUT low	4		ns
$t_{h(A\text{-}D)R}$	Hold time, read data after address invalid	0		ns
$t_{h(D)MSTRBH}$	Hold time, read data after $\overline{\text{DSP\_MSTRB}}$ high	0		ns

† Address, $\overline{\text{DSP\_PS}}$, and $\overline{\text{DSP\_DS}}$ timings are all included in timings referenced as address.
‡ This access timing reflects a zero wait-state timing.

Table 6–7. Memory Read Switching Characteristics†

PARAMETER		MIN	MAX	UNIT
$t_{d(CLKL\text{-}A)}$	Delay time, DSP_CLKOUT low to address valid§	1.5	11.5	ns
$t_{d(CLKH\text{-}A)}$	Delay time, DSP_CLKOUT high (transition) to address valid¶	1.5	11.5	ns
$t_{d(CLKL\text{-}MSL)}$	Delay time, DSP_CLKOUT low to $\overline{\text{DSP\_MSTRB}}$ low	1.5	11.5	ns
$t_{d(CLKL\text{-}MSH)}$	Delay time, DSP_CLKOUT low to $\overline{\text{DSP\_MSTRB}}$ high	1.5	11.5	ns
$t_{h(CLKL\text{-}A)R}$	Hold time, address valid after DSP_CLKOUT low§	1.5	11.5	ns
$t_{h(CLKH\text{-}A)R}$	Hold time, address valid after DSP_CLKOUT high¶	1.5	11.5	ns

† Address, $\overline{\text{DSP\_PS}}$, and $\overline{\text{DSP\_DS}}$ timings are all included in timings referenced as address.
§ In the case of a memory read preceded by a memory read
¶ In the case of a memory read preceded by a memory write

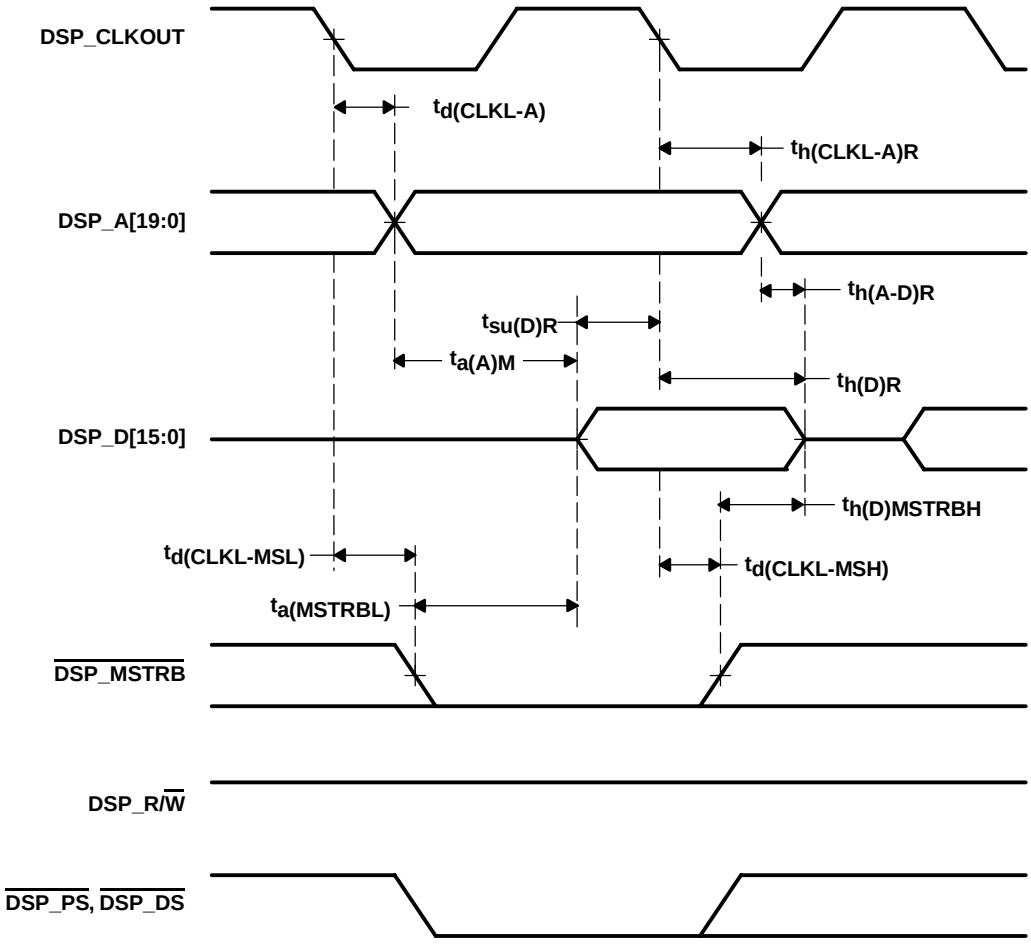


Figure 6–4. Memory Read

6.6.2 Memory Write

Table 6–8 assumes testing over recommended operating conditions with $\overline{\text{DSP\_MSTRB}} = 0$ and $H = 0.5t_{c(CO)}$ (see Figure 6–5).

Table 6–8. Memory Write Switching Characteristics†

PARAMETER		MIN	MAX	UNIT
$t_{d(CLKH-A)}$	Delay time, DSP_CLKOUT high to address valid‡	1.5	11.5	ns
$t_{d(CLKL-A)}$	Delay time, DSP_CLKOUT low to address valid§	1.5	11.5	ns
$t_{d(CLKL-MSL)}$	Delay time, DSP_CLKOUT low to $\overline{\text{DSP\_MSTRB}}$ low	1.5	11.5	ns
$t_{d(CLKL-D)W}$	Delay time, DSP_CLKOUT low to data valid	1.5	11.5	ns
$t_{d(CLKL-MSH)}$	Delay time, DSP_CLKOUT low to $\overline{\text{DSP\_MSTRB}}$ high	1.5	11.5	ns
$t_{d(CLKH-RWL)}$	Delay time, DSP_CLKOUT high to DSP_R/$\overline{\text{W}}$ low	1.5	11.5	ns
$t_{d(CLKH-RWH)}$	Delay time, DSP_CLKOUT high to DSP_R/$\overline{\text{W}}$ high	1.5	11.5	ns
$t_{d(RWL-MSTRBL)}$	Delay time, DSP_R/$\overline{\text{W}}$ low to $\overline{\text{DSP\_MSTRB}}$ low	H – 3	H + 3	ns
$t_{h(A)W}$	Hold time, address valid after DSP_CLKOUT high‡	1.5	11.5	ns
$t_{h(D)MSH}$	Hold time, write data valid after $\overline{\text{DSP\_MSTRB}}$ high	H–3	H+3§	ns
$t_{w(SL)MS}$	Pulse duration, $\overline{\text{DSP\_MSTRB}}$ low	2H–3		ns
$t_{su(A)W}$	Setup time, address valid before $\overline{\text{DSP\_MSTRB}}$ low	2H–3		ns
$t_{su(D)MSH}$	Setup time, write data valid before $\overline{\text{DSP\_MSTRB}}$ high	2H–3	2H+3§	ns
$t_{en(D–RWL)}$	Enable time, data bus driven after DSP_R/$\overline{\text{W}}$ low	H–3		ns
$t_{dis(RWH–D)}$	Disable time, DSP_R/$\overline{\text{W}}$ high to data bus high impedance		2	ns

† Address, $\overline{\text{DSP\_PS}}$, and $\overline{\text{DSP\_DS}}$ timings are all included in timings referenced as address.
‡ In the case of a memory write preceded by a memory write
§ In the case of a memory write preceded by an I/O cycle

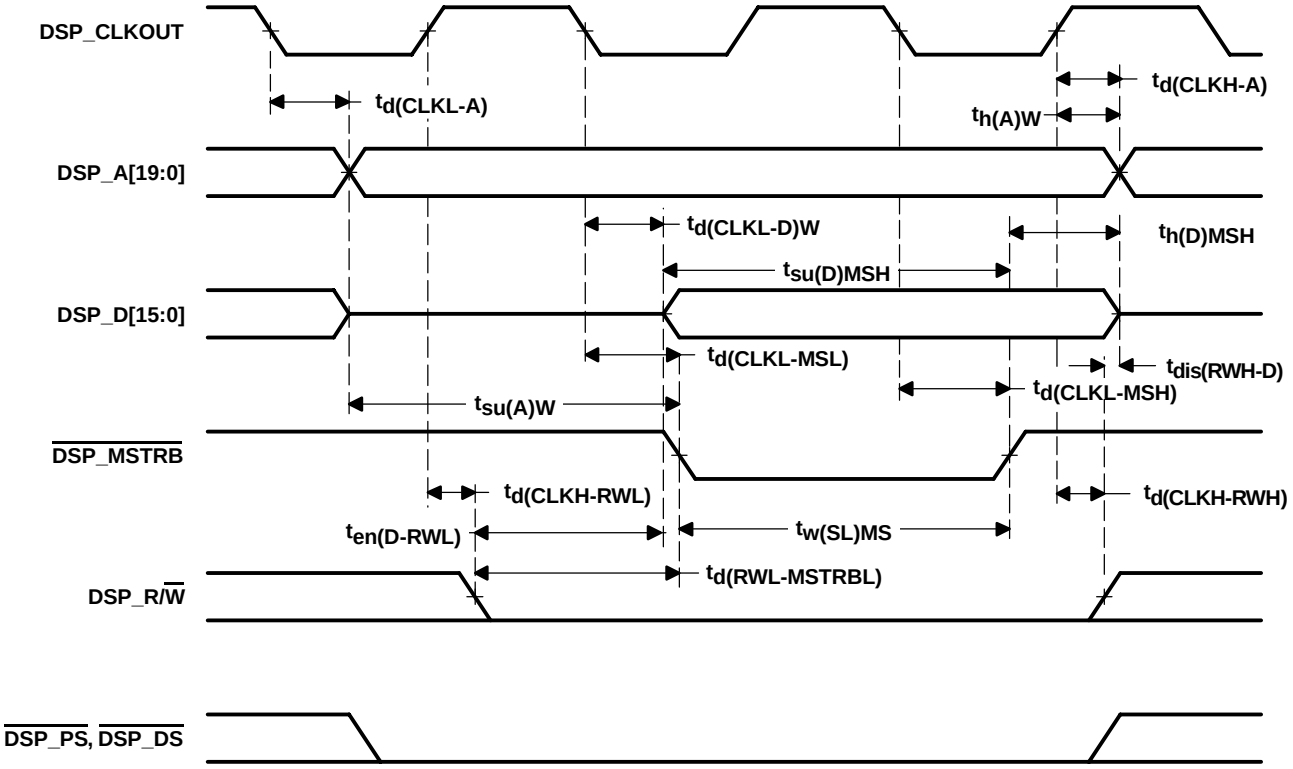


Figure 6–5. Memory Write

6.6.3 Parallel I/O Port Read Timing Requirements

Table 6–9 and Table 6–10 assume testing over recommended operating conditions with $\overline{\text{DSP\_IOSTRB}}$ = 0 and H = 0.5$t_{c(CO)}$ (see Figure 6–6).

Table 6–9. Parallel I/O Port Read Timing Requirements†

		MIN	MAX	UNIT
$t_{a(A)IO}$	Access time, read data access from address valid‡		3H–15	ns
$t_{a(ISTRBL)IO}$	Access time, read data access from $\overline{\text{DSP\_IOSTRB}}$ low‡		2H–15	ns
$t_{su(D)IOR}$	Setup time, read data before DSP_CLKOUT high	4		ns
$t_{h(D)IOR}$	Hold time, read data after DSP_CLKOUT high	0		ns
$t_{h(ISTRBH-D)R}$	Hold time, read data after $\overline{\text{DSP\_IOSTRB}}$ high	0		ns

† Address and $\overline{\text{DSP\_IS}}$ timings are included in timings referenced as address.
‡ This access timing reflects a zero wait-state timing.

Table 6–10. Parallel I/O Port Read Switching Characteristics†

PARAMETER		MIN	MAX	UNIT
$t_{d(CLKL-A)}$	Delay time, DSP_CLKOUT low to address valid	1.5	11.5	ns
$t_{d(CLKH-ISTRBL)}$	Delay time, DSP_CLKOUT high to $\overline{\text{DSP\_IOSTRB}}$ low	1.5	11.5	ns
$t_{d(CLKH-ISTRBH)}$	Delay time, DSP_CLKOUT high to $\overline{\text{DSP\_IOSTRB}}$ high	1.5	11.5	ns
$t_{h(A)IOR}$	Hold time, address after DSP_CLKOUT low	1.5	11.5	ns

† Address and $\overline{\text{DSP\_IS}}$ timings are included in timings referenced as address.

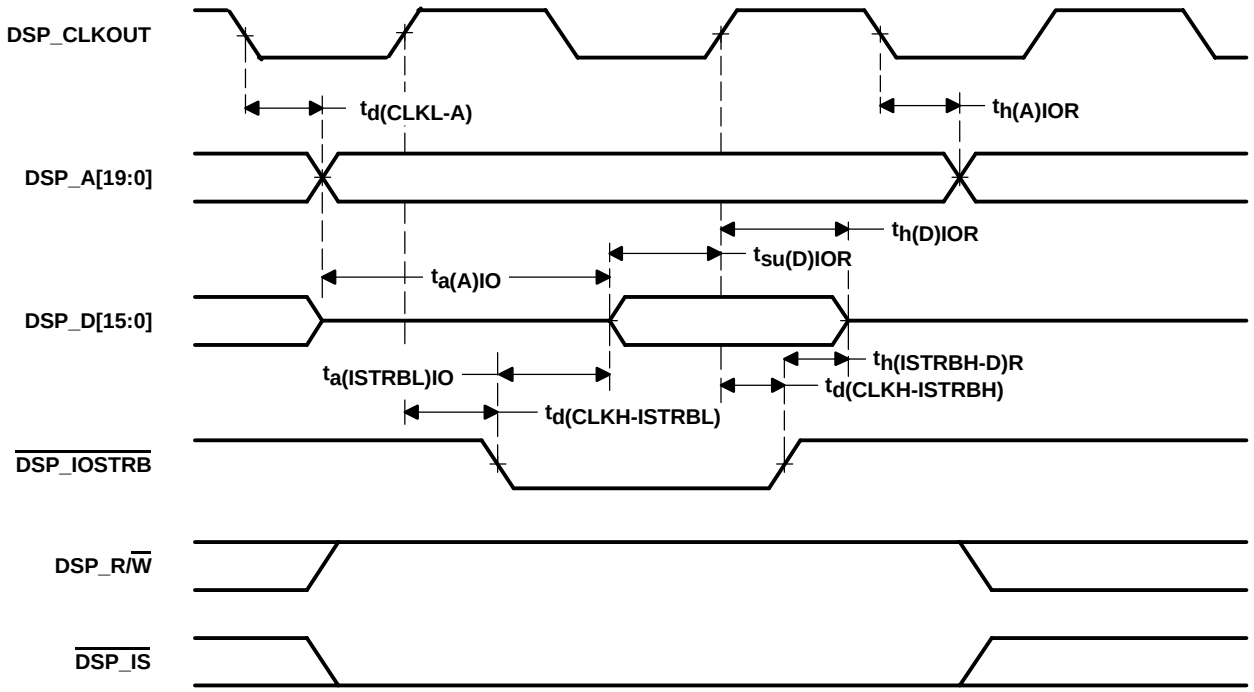


Figure 6–6. Parallel I/O Port Read

6.6.4 Parallel I/O Port Write Switching Characteristics

Table 6–11 assumes testing over recommended operating conditions with $\overline{\text{DSP\_IOSTRB}}$ = 0 and H = $0.5t_{c(CO)}$ (see Figure 6–7).

Table 6–11. Parallel I/O Port Write Switching Characteristics†

PARAMETER		MIN	MAX	UNIT
$t_{d(CLKL-A)}$	Delay time, DSP_CLKOUT low to address valid	1.5	11.5	ns
$t_{d(CLKH-ISTRBL)}$	Delay time, DSP_CLKOUT high to $\overline{\text{DSP\_IOSTRB}}$ low	1.5	11.5	ns
$t_{d(CLKH-D)IOW}$	Delay time, DSP_CLKOUT high to write data valid	1.5	11.5	ns
$t_{d(CLKH-ISTRBH)}$	Delay time, DSP_CLKOUT high to $\overline{\text{DSP\_IOSTRB}}$ high	1.5	11.5	ns
$t_{d(CLKL-RWL)}$	Delay time, DSP_CLKOUT low to DSP_R/$\overline{W}$ low	1.5	11.5	ns
$t_{d(CLKL-RWH)}$	Delay time, DSP_CLKOUT low to DSP_R/$\overline{W}$ high	1.5	11.5	ns
$t_{h(A)IOW}$	Hold time, address valid after DSP_CLKOUT low	1.5	11.5	ns
$t_{h(D)IOW}$	Hold time, write data after $\overline{\text{DSP\_IOSTRB}}$ high	H–3	H+3	ns
$t_{su(D)DSP\_IOSTRBH}$	Setup time, write data before $\overline{\text{DSP\_IOSTRB}}$ high	H–3	H+3	ns
$t_{su(A)DSP\_IOSTRBL}$	Setup time, address valid before $\overline{\text{DSP\_IOSTRB}}$ low	H–3	H+3	ns

† Address and $\overline{\text{DSP\_IS}}$ timings are included in timings referenced as address.

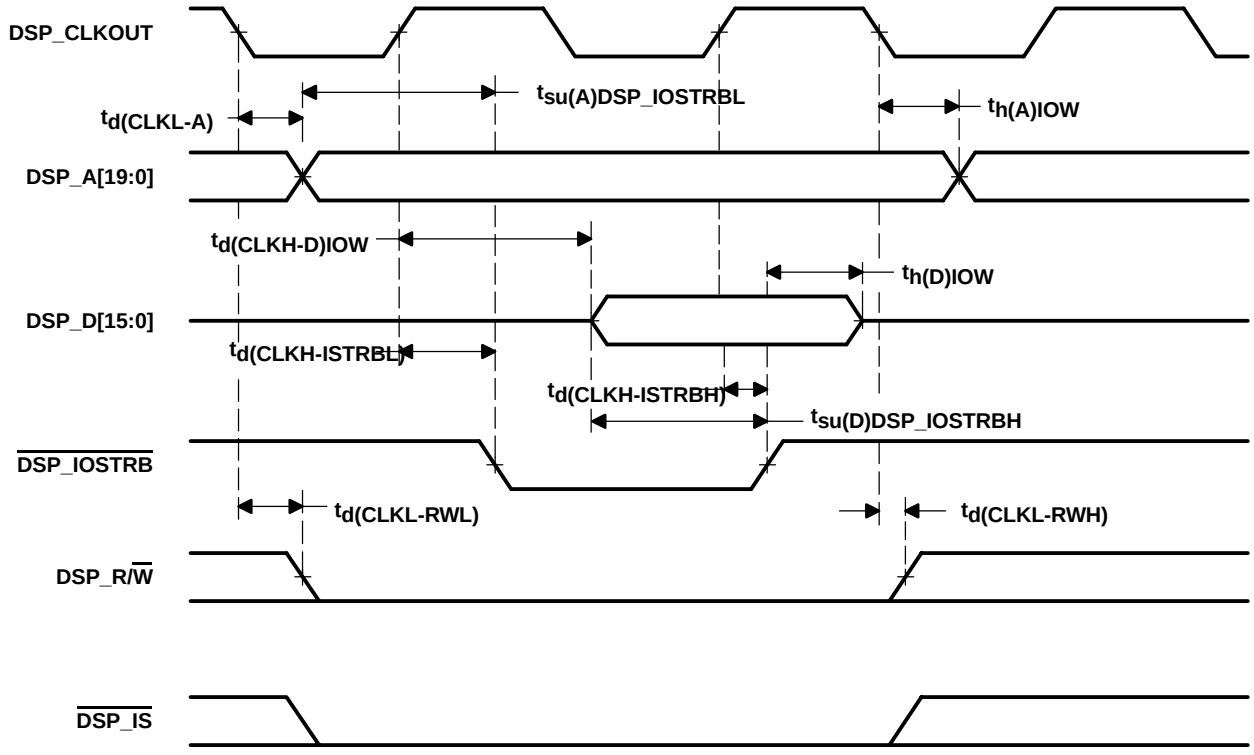


Figure 6–7. Parallel I/O Port Write

6.7 Ready Timing for Externally Generated Wait-States

Table 6–12 assumes testing over recommended operating conditions and H = $0.5t_{c(CO)}$ (see Figure 6–8, through Figure 6–11).

Table 6–12. Ready Timing For Externally Generated Wait-States Timing Requirements†

		MIN	MAX	UNIT
$t_{su(RDY)}$	Setup time, DSP_READY before DSP_CLKOUT low	5		ns
$t_{h(RDY)}$	Hold time, DSP_READY after DSP_CLKOUT low	3		ns
$t_{v(RDY)MSTRB}$	Valid time, DSP_READY after $\overline{\text{DSP\_MSTRB}}$ low‡		4H–11	ns
$t_{h(RDY)MSTRB}$	Hold time, DSP_READY after $\overline{\text{DSP\_MSTRB}}$ low‡	4H		ns
$t_{v(RDY)DSP\_IOSTRB}$	Valid time, DSP_READY after $\overline{\text{DSP\_IOSTRB}}$ low‡		5H–11	ns
$t_{h(RDY)DSP\_IOSTRB}$	Hold time, DSP_READY after $\overline{\text{DSP\_IOSTRB}}$ low‡	5H		ns
$t_{v(MSCL)}$	Valid time, $\overline{\text{DSP\_MSC}}$ low after DSP_CLKOUT low	1.5	11.5	ns
$t_{v(MSCH)}$	Valid time, $\overline{\text{DSP\_MSC}}$ high after DSP_CLKOUT low	1.5	11.5	ns

† The hardware wait-states can be used only in conjunction with the software wait-states to extend the bus cycles. To generate wait-states using DSP_READY, at least two software wait-states must be programmed.

‡ These timings are included for reference only. The critical timings for DSP_READY are those referenced to DSP_CLKOUT.

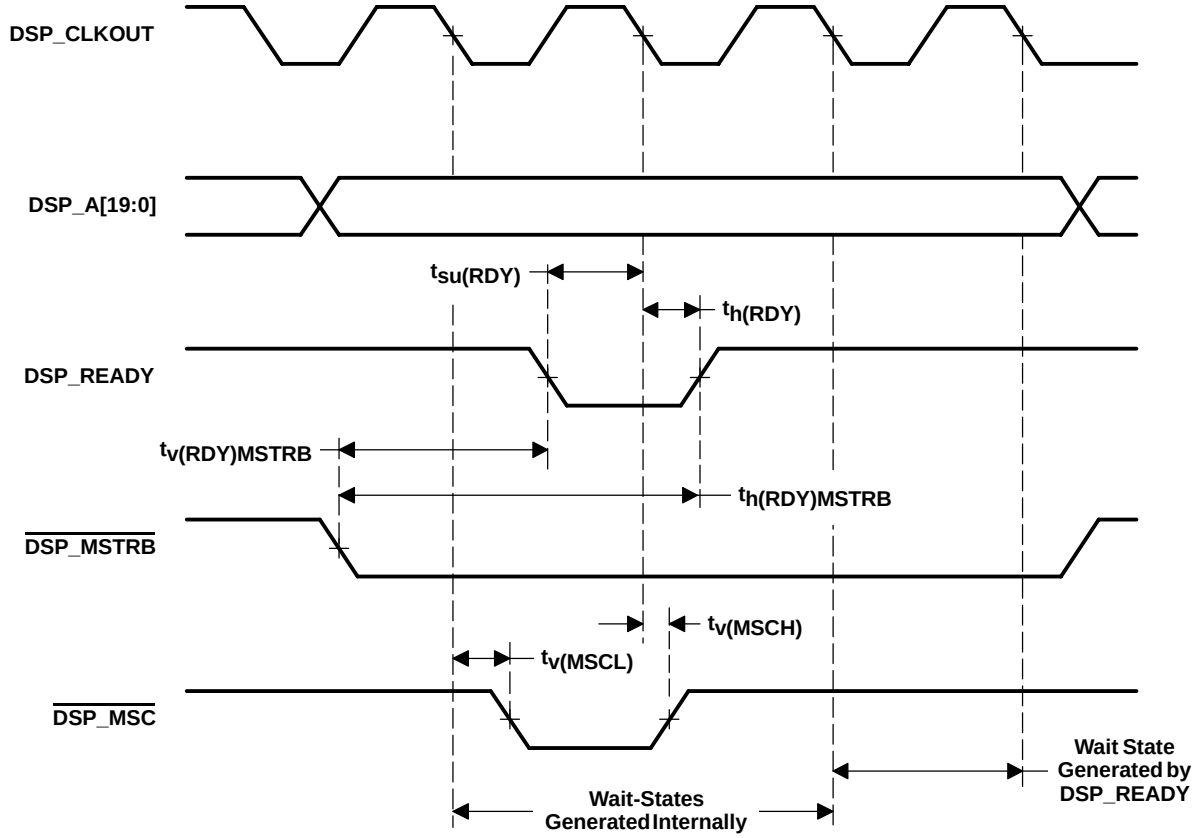


Figure 6–8. Memory Read With Externally Generated Wait-States

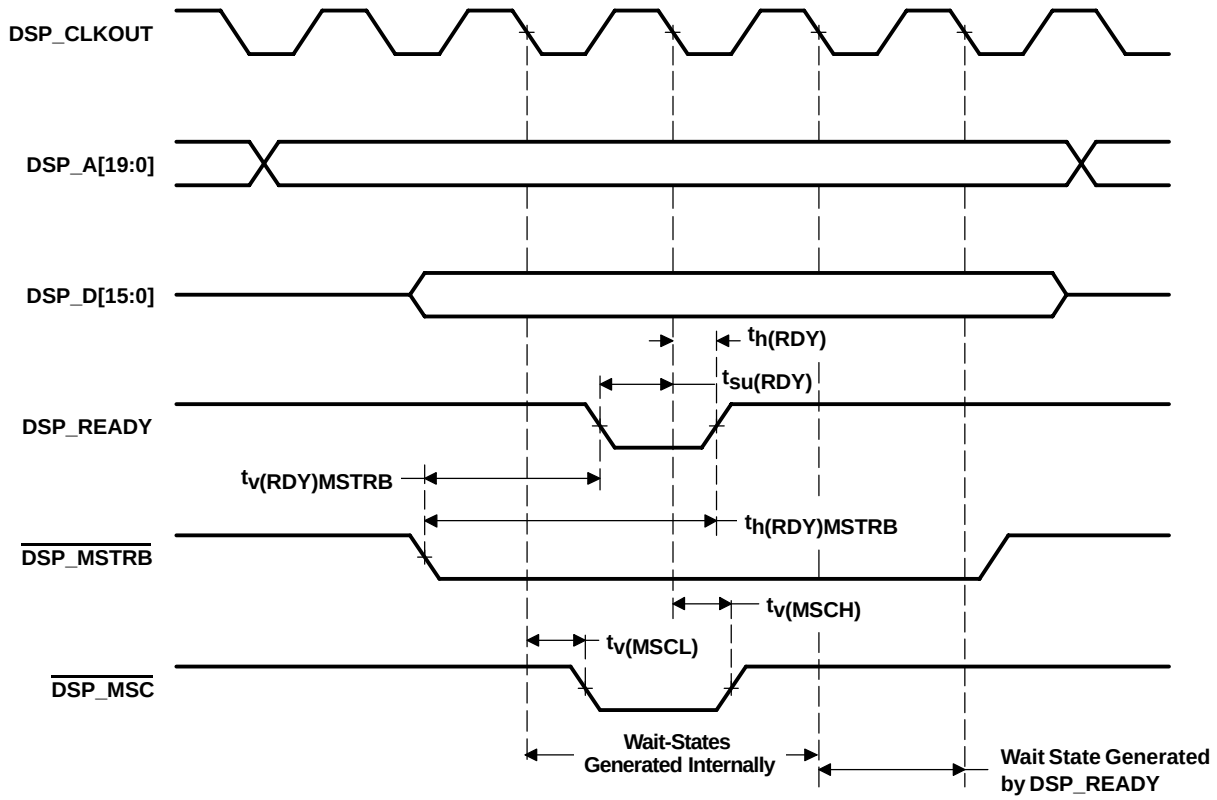


Figure 6–9. Memory Write With Externally Generated Wait-States

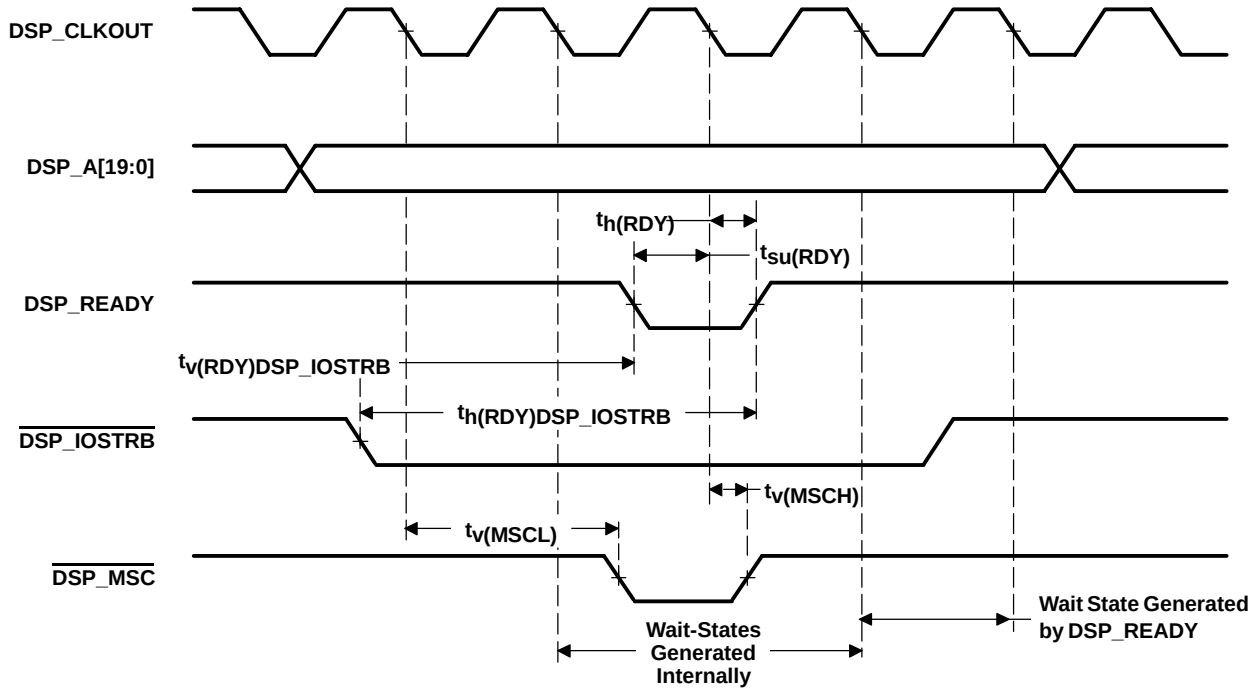


Figure 6–10. I/O Read With Externally Generated Wait-States

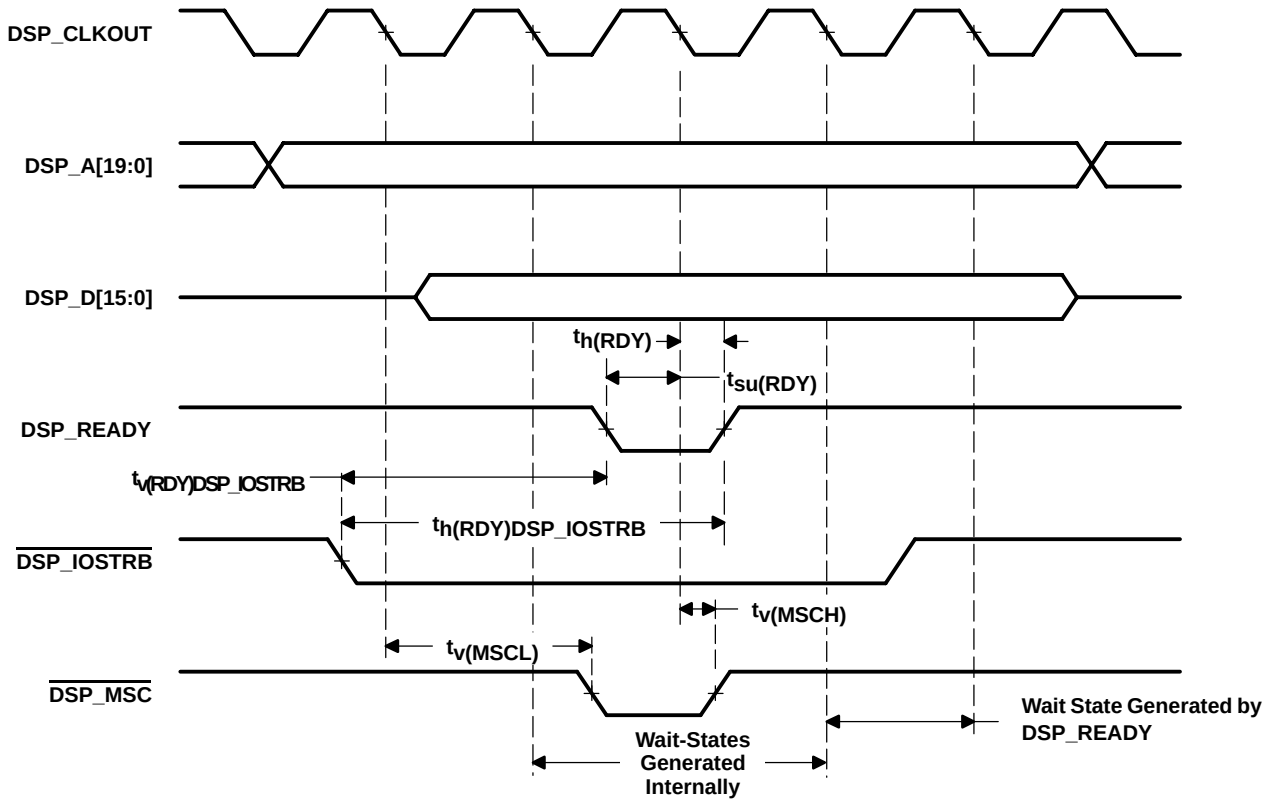


Figure 6–11. I/O Write With Externally Generated Wait-States

6.8 Interrupt Timings

Table 6–13 assumes testing over recommended operating conditions and H = $0.5t_{c(CO)}$ (see Figure 6–12).

Table 6–13. Reset and Interrupt Timing Requirements

		MIN	MAX	UNIT
$t_{w(INTH)S}$	Pulse duration, $\overline{DSP\_INT0}$ high (synchronous)	2H+1		ns
$t_{w(INTL)S}$	Pulse duration, $\overline{DSP\_INT0}$ low (synchronous)	2H+1		ns
$t_{w(INTL)A}$	Pulse duration, $\overline{DSP\_INT0}$ low (asynchronous)	4H		ns
$t_{w(INTL)WKP}$	Pulse duration, $\overline{DSP\_INT0}$ low for IDLE2/IDLE3 wakeup	10		ns
$t_{su(INT)}$	Setup time, $\overline{DSP\_INT0}$ before DSP_CLKOUT low†	8	10	ns

† The external interrupt is synchronized to the core CPU by way of a two-flip-flop synchronizer which samples these inputs with consecutive falling edges of DSP_CLKOUT. The input to the interrupt pin is required to represent a 1-0-0 sequence at the timing that is corresponding to three DSP_CLKOUT sampling sequences.

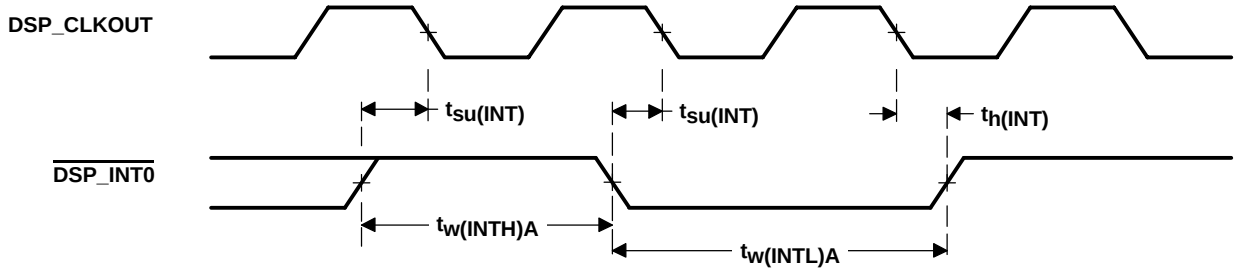


Figure 6–12. Interrupt Timing

6.9 Instruction Acquisition ($\overline{\text{DSP\_IAQ}}$) and Interrupt Acknowledge ($\overline{\text{DSP\_IACK}}$) Timings

Table 6–14 assumes testing over recommended operating conditions and H = $0.5t_{c(CO)}$ (see Figure 6–13).

Table 6–14. $\overline{\text{DSP\_IAQ}}$ and $\overline{\text{DSP\_IACK}}$ Switching Characteristics

PARAMETER		MIN	MAX	UNIT
$t_{d(CLKL-IAQL)}$	Delay time, DSP_CLKOUT low to $\overline{\text{DSP\_IAQ}}$ low	1.5	11.5	ns
$t_{d(CLKL-IAQH)}$	Delay time, DSP_CLKOUT low to $\overline{\text{DSP\_IAQ}}$ high	1.5	11.5	ns
$t_{d(A)IAQ}$	Delay time, address valid to $\overline{\text{DSP\_IAQ}}$ low		1	ns
$t_{d(CLKL-IACKL)}$	Delay time, DSP_CLKOUT low to $\overline{\text{DSP\_IACK}}$ low	1.5	11.5	ns
$t_{d(CLKL-IACKH)}$	Delay time , DSP_CLKOUT low to $\overline{\text{DSP\_IACK}}$ high	1.5	11.5	ns
$t_{d(A)IACK}$	Delay time, address valid to $\overline{\text{DSP\_IACK}}$ low		1	ns
$t_{h(A)IAQ}$	Hold time, $\overline{\text{DSP\_IAQ}}$ high after address invalid	– 2		ns
$t_{h(A)IACK}$	Hold time, $\overline{\text{DSP\_IACK}}$ high after address invalid	– 2		ns
$t_{w(IAQL)}$	Pulse duration, $\overline{\text{DSP\_IAQ}}$ low	2H–3		ns
$t_{w(IACKL)}$	Pulse duration, $\overline{\text{DSP\_IACK}}$ low	2H–3		ns

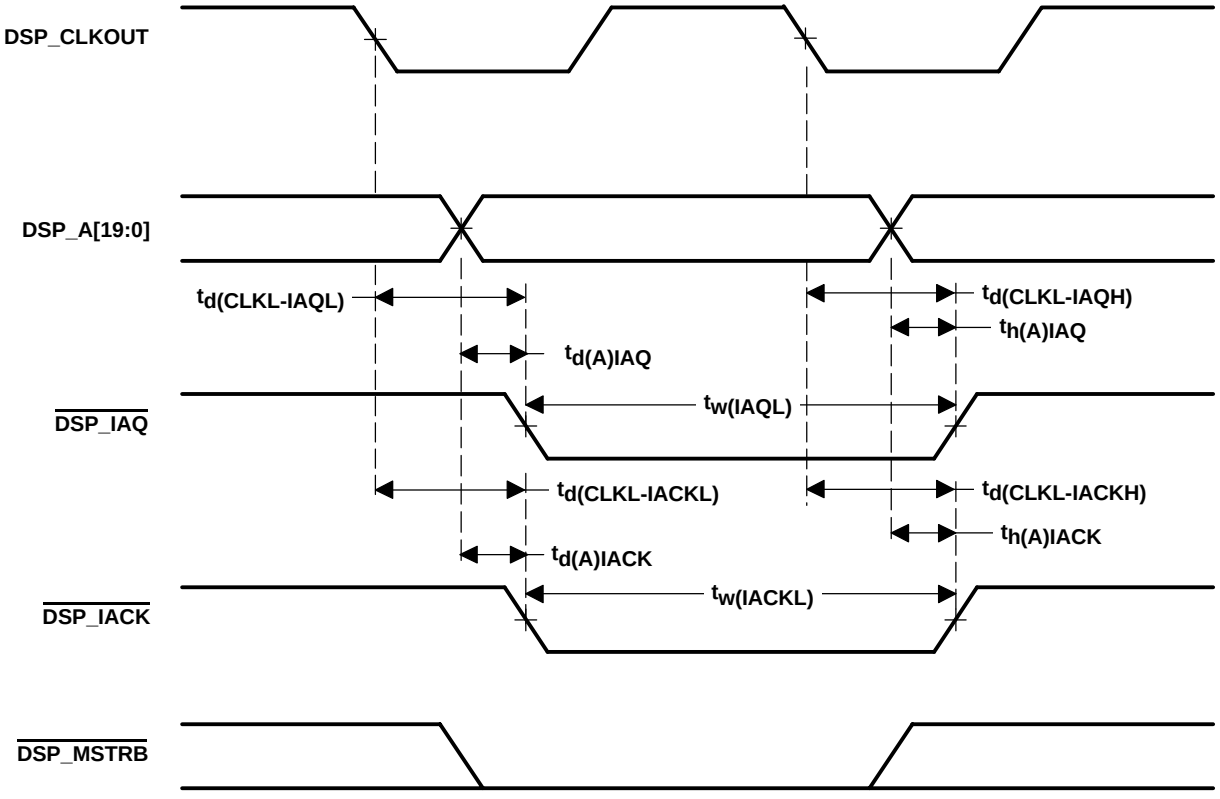


Figure 6–13. $\overline{\text{DSP\_IAQ}}$ and $\overline{\text{DSP\_IACK}}$ Timings

6.10 External Flag (DSP_XF) and DSP_TOUT Timings

Table 6–15 assumes testing over recommended operating conditions and H = 0.5$t_{c(CO)}$ (see Figure 6–14 and Figure 6–15).

Table 6–15. DSP_XF and DSP_TOUT Switching Characteristics

PARAMETER		MIN	TYP	MAX	UNIT
$t_{d(XF)}$	Delay time, DSP_CLKOUT low to DSP_XF high	1.5		11.5	ns
	Delay time, DSP_CLKOUT low to DSP_XF low	1.5		11.5	
$t_{d(TOUTH)}$	Delay time, DSP_CLKOUT low to DSP_TOUT high	1.5		11.5	ns
$t_{d(TOUTL)}$	Delay time, DSP_CLKOUT low to DSP_TOUT low	1.5		11.5	ns
$t_{w(TOUT)}$	Pulse duration, DSP_TOUT	2H–3	2H	2H+3	ns

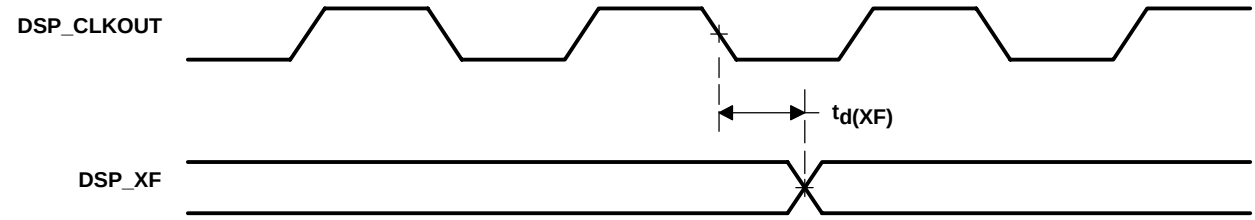


Figure 6–14. DSP_XF Timing

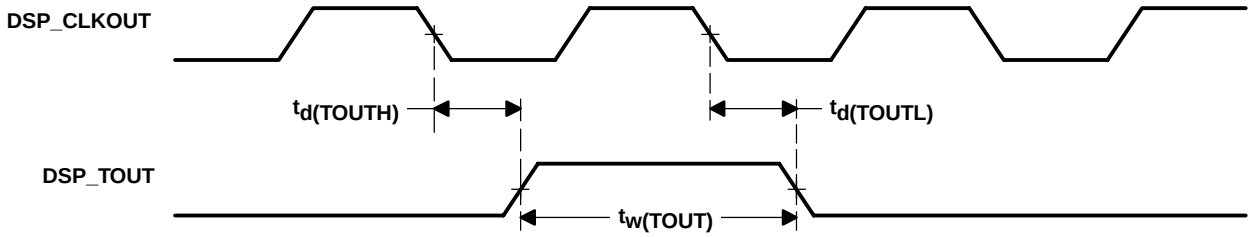


Figure 6–15. DSP_TOUT Timing

6.11 Multichannel Buffered Serial Port (McBSP) Timings

6.11.1 McBSP Receive and Transmit Timings

Table 6–16 and Table 6–17 assume testing over recommended operating conditions and H = 0.5$t_{c(CO)}$ (see Figure 6–16 and Figure 6–17).

NOTE: Polarity bits CLKRP = CLKXP = FSRP = FSXP = 0. If the polarity of any of the signals is inverted, then the timing references of that signal are also inverted.

Table 6–16. McBSP Receive and Transmit Timing Requirements

			MIN	MAX	UNIT
$t_{c(BCKRX)}$	Cycle time, BCLK_R/X	BCLK_R/X ext	4H		ns
$t_{w(BCKRX)}$	Pulse duration, BCLK_R/X or BCLK_R/X high	BCLK_R/X ext	2H–1		ns
$t_{h(BCKRL-BFRH)}$	Hold time, external BFSR high after BCLKR low	BCLKR int	0		ns
		BCLKR ext	7		
$t_{h(BCKRL-BDRV)}$	Hold time, BDR valid after BCLKR low	BCLKR int	0		ns
		BCLKR ext	7		
$t_{h(BCKXL-BFXH)}$	Hold time, external BFSX high after BCLKX low	BCLKX int	0		ns
		BCLKX ext	7		
$t_{su(BFRH-BCKRL)}$	Setup time, external BFSR high before BCLKR low	BCLKR int	9		ns
		BCLKR ext	2		
$t_{su(BDRV-BCKRL)}$	Setup time, BDR valid before BCLKR low	BCLKR int	9		ns
		BCLKR ext	2		
$t_{su(BFXH-BCKXL)}$	Setup time, external BFSX high before BCLKX low	BCLKX int	9		ns
		BCLKX ext	2		
$t_{r(BCKRX)}$	Rise time, BCLKR_R/X	BCLK_R/X ext		5	ns
$t_{f(BCKRX)}$	Fall time, BCLKR_R/X	BCLK_R/X ext		5	ns

Table 6–17. McBSP Receive and Transmit Switching Characteristics

PARAMETER			MIN	MAX	UNIT
$t_{c(BCKRX)}$	Cycle time, BCLK_R/X	BCLK_R/X int	4H		ns
$t_{w(BCKRXH)}$	Pulse duration, BCLK_R/X high	BCLK_R/X int	D–6†	D+1†	ns
$t_{w(BCKRXL)}$	Pulse duration, BCLK_R/X low	BCLK_R/X int	C–6†	C+1†	ns
$t_{d(BCKRH-BFRV)}$	Delay time, BCLKR high to internal BFSR valid	BCLKR int	–1	8	ns
$t_{d(BCKXH-BFXV)}$	Delay time, BCLKX high to internal BFSX valid	BCLKX int	1	9	ns
		BCLKX ext	4	15	
$t_{dis(BCKXH-BDXHZ)}$	Disable time, BCLKX high to BDX high impedance following last data bit	BCLKX int	–1	8	ns
		BCLKX ext	4	16	
$t_{d(BCKXH-BDXV)}$	Delay time, BCLKX high to BDX valid. This applies to all bits except the first bit transmitted.	BCLKX int	0	7	ns
		BCLKX ext	4	16	
	Delay time, BCLKX high to BDX valid.‡§ Only applies to first bit transmitted when in Data Delay 1 or 2 (XDATDLY=01b or 10b) modes	BCLKX int		7	
		BCLKX ext		16	
$t_{en(BCKXH-BDX)}$	Enable time, BCLKX high to BDX driven.‡§ Only applies to first bit transmitted when Data Delay 1 or 2 (XDATDLY=01b or 10b) modes	BCLKX int	–4		ns
		BCLKX ext	2		
$t_{d(BFXH-BDXV)}$	Delay time, BFSX high to BDX valid.‡§ Only applies to first bit transmitted when Data Delay 0 (XDATDLY=00b) mode.	BFSX int		3	ns
		BFSX ext		14	
$t_{en(BFXH-BDX)}$	Enable time, BFSX high to BDX driven.‡§ Only applies to first bit transmitted when in Data Delay 0 (XDATDLY=00b) mode	BFSX int	–1		ns
		BFSX ext	2		

† T = BCLKR/X period = (1 + CLKGDV) * 2H
C = BCLKR/X low pulse width = T/2 when CLKGDV is odd or zero and = (CLKGDV/2) * 2H when CLKGDV is even
D = BCLKR/X high pulse width = T/2 when CLKGDV is odd or zero and = (CLKGDV/2 + 1) * 2H when CLKGDV is even

‡ See the *TMS320C54x DSP Reference Set, Volume 5: Enhanced Peripherals* (literature number SPRU302) for a description of the data delay features of the McBSP.

§ The transmit delay enable (DXENA) and A-bis mode (ABIS) features of the McBSP are not implemented on the TMS320VC5471.

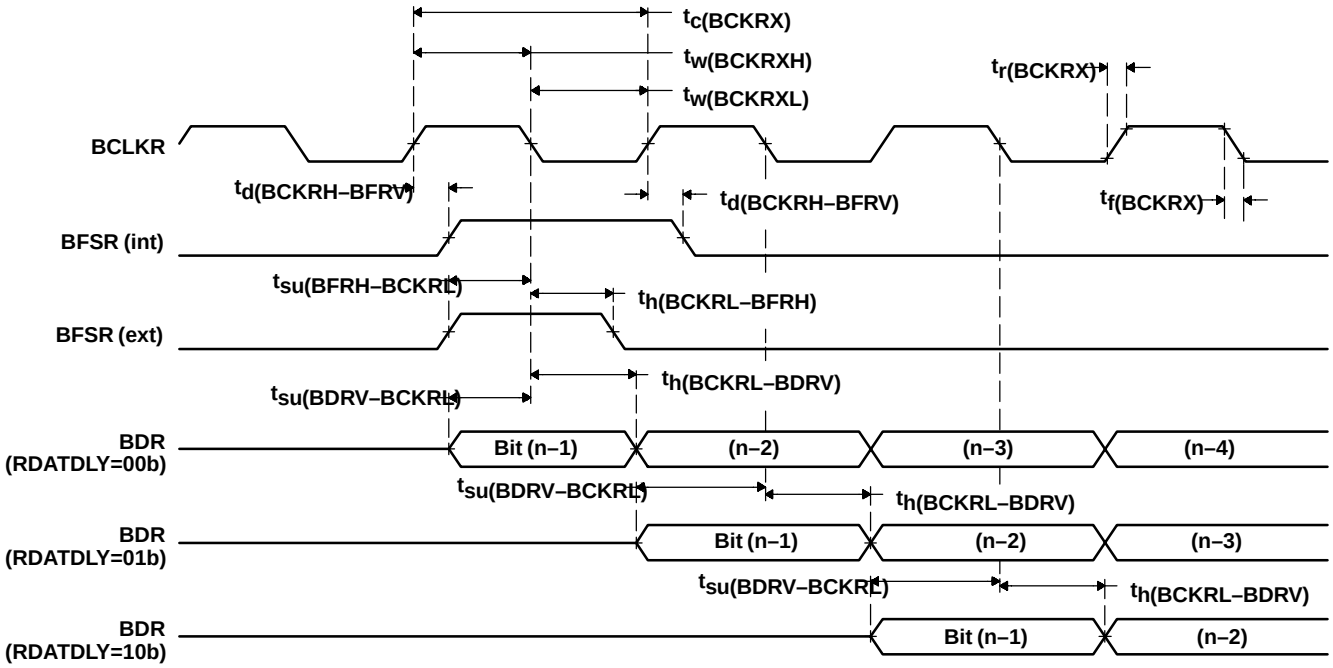


Figure 6–16. McBSP Receive Timings

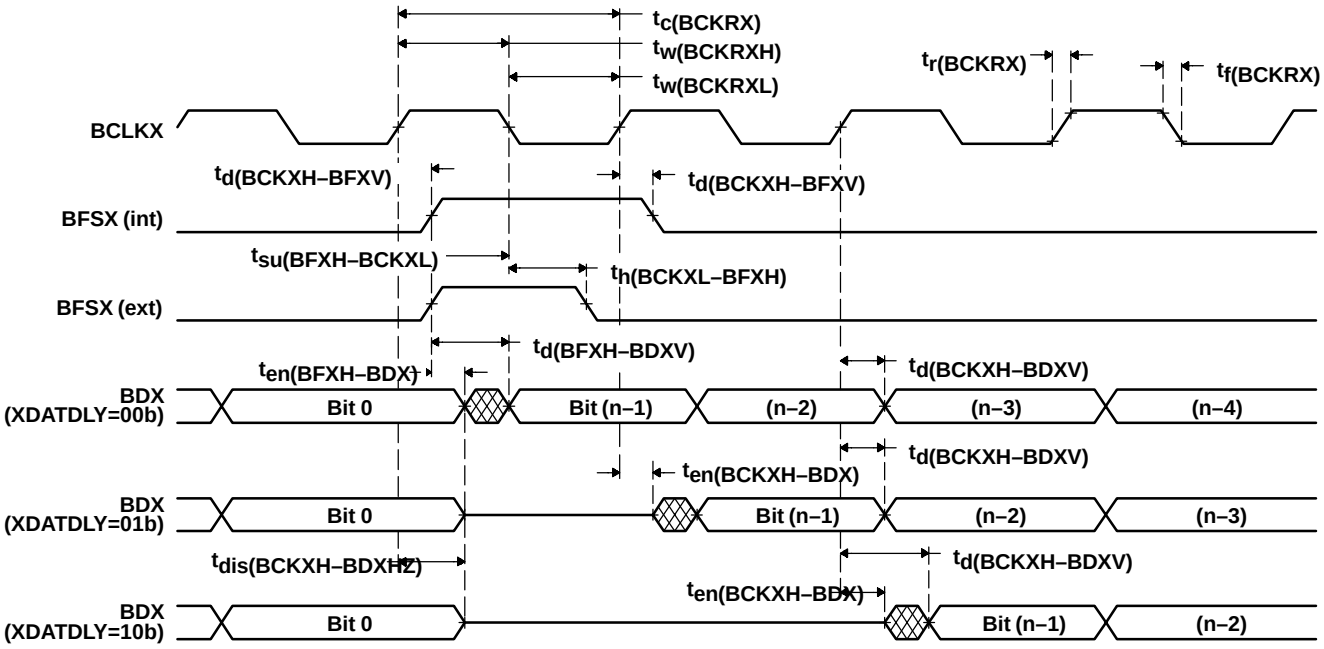


Figure 6–17. McBSP Transmit Timings

6.11.2 McBSP General-Purpose Timings

Table 6–18 and Table 6–19 assume testing over recommended operating conditions (see Figure 6–18).

Table 6–18. McBSP General-Purpose Timing Requirements

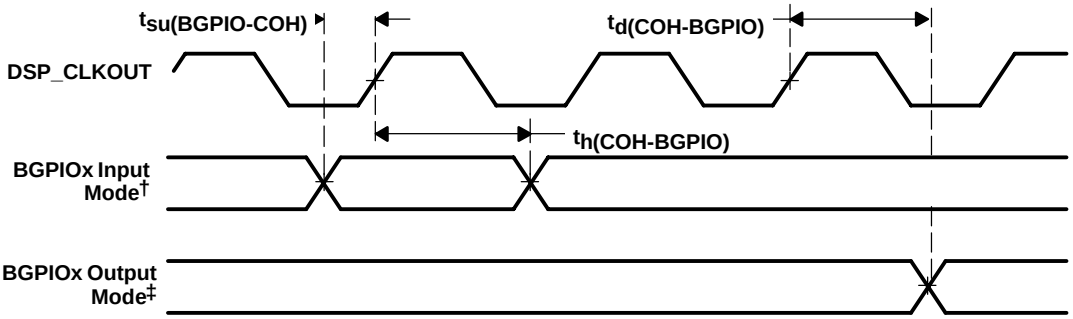
		MIN	MAX	UNIT
$t_{su(BGPIO-COH)}$	Setup time, BGPIOx input mode before DSP_CLKOUT high†	9		ns
$t_{h(COH-BGPIO)}$	Hold time, BGPIOx input mode after DSP_CLKOUT high†	0		ns

† BGPIOx refers to BCLKRx, BFSRx, BDRx, BCLKXx, or BFSXx when configured as a general-purpose input.

Table 6–19. McBSP General-Purpose Switching Characteristics

PARAMETER		MIN	MAX	UNIT
$t_{d(COH-BGPIO)}$	Delay time, DSP_CLKOUT high to BGPIOx output mode‡	–10	10	ns

‡ BGPIOx refers to BCLKRx, BFSRx, BCLKXx, BFSXx, or BDXx when configured as a general-purpose output.
NOTE: Only the DSP subsystem can access the McBSPs as GPIOs; they are not available to the ARM subsystem.



† BGPIOx refers to BCLKRx, BFSRx, BDRx, BCLKXx, or BFSXx when configured as a general-purpose input.
‡ BGPIOx refers to BCLKRx, BFSRx, BCLKXx, BFSXx, or BDXx when configured as a general-purpose output.

Figure 6–18. McBSP General-Purpose I/O Timings

6.11.3 McBSP as SPI Master or Slave Timings

Low inactive state without delay; the McBSP transmits data on the rising edge of BCLKX and receives data on the falling edge of BCLKR.

Table 6–20 and Table 6–21 assume testing over recommended operating conditions, CLKSTP = 10b, CLKXP = 0, and H = $0.5t_{c(CO)}$ (see Figure 6–19).

NOTE: For all SPI slave modes, CLKG is programmed as 1/2 of the CPU clock by setting CLKSM = CLKGDV = 1.

Table 6–20. McBSP as SPI Master or Slave Timing Requirements (CLKSTP = 10b and CLKXP = 0)

		MASTER		SLAVE		UNIT
		MIN	MAX	MIN	MAX	
$t_{su(BDRV-BCKXL)}$	Setup time, BDR valid before BCLKX low	12		– 12H		ns
$t_{h(BCKXL-BDRV)}$	Hold time, BDR valid after BCLKX low	0		12H + 5		ns
$t_{su(BFXL-BCKXH)}$	Setup time, BFSX low before BCLKX high			10		ns
$t_{c(BCKX)}$	Cycle time, BCLKX	12H		32H		ns

Table 6–21. McBSP as SPI Master or Slave Switching Characteristics (CLKSTP = 10b and CLKXP = 0)

PARAMETER		MASTER†		SLAVE		UNIT
		MIN	MAX	MIN	MAX	
$t_{h(BCKXL-BFXL)}$	Hold time, BFSX low after BCLKX low‡	T – 6	T + 6			ns
$t_{d(BFXL-BCKXH)}$	Delay time, BFSX low to BCLKX high§	C – 6	C + 6			ns
$t_{d(BCKXH-BDXV)}$	Delay time, BCLKX high to BDX valid	– 3	7	6H + 5	10H + 14	ns
$t_{dis(BCKXL-BDXHZ)}$	Disable time, BDX high impedance following last data bit from BCLKX low	C – 2	C + 3			ns
$t_{dis(BFXH-BDXHZ)}$	Disable time, BDX high impedance following last data bit from BFSX high			2H+ 3	6H + 17	ns
$t_{d(BFXL-BDXV)}$	Delay time, BFSX low to BDX valid			4H + 2	8H + 17	ns

† T = BCLKX period = (1 + CLKGDV) * 2H
C = BCLKX low pulse width = T/2 when CLKGDV is odd or zero and = (CLKGDV/2) * 2H when CLKGDV is even

‡ FSRP = FSXP = 1. As a SPI master, BFSX is inverted to provide active-low slave-enable output. As a slave, the active-low signal input on BFSX and BFSR is inverted before being used internally.
CLKXM = FSXM = 1, CLKRM = FSRM = 0 for master McBSP
CLKXM = CLKRM = FSXM = FSRM = 0 for slave McBSP

§ BFSX should be low before the rising edge of clock to enable slave devices and then begin a SPI transfer at the rising edge of the master clock (BCLKX).

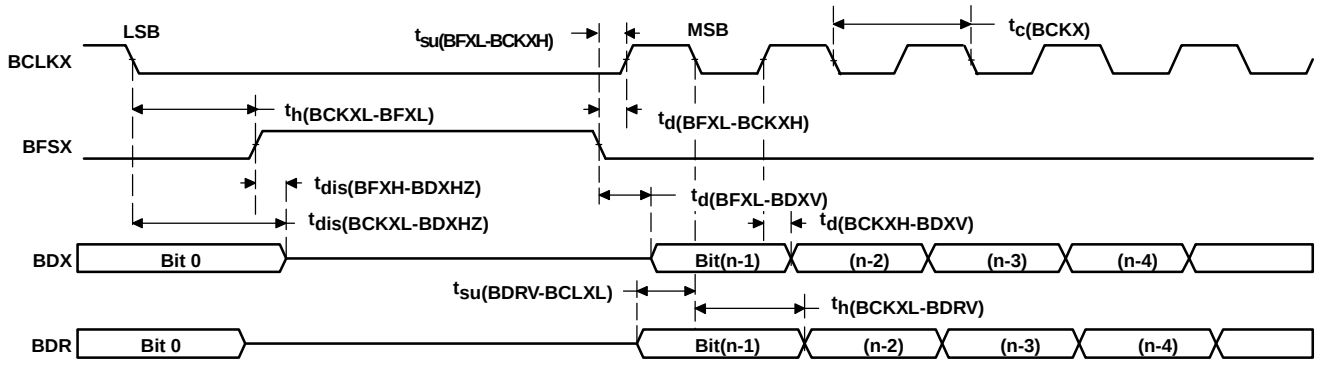


Figure 6–19. McBSP Timing as SPI Master or Slave: CLKSTP = 10b, CLKXP = 0

Low inactive state with delay; the McBSP transmits data one-half cycle ahead of the rising edge of BCLKX and receives data on the rising edge of BCLKR.

Table 6–22 and Table 6–23 assume testing over recommended operating conditions, CLKSTP = 11b, CLKXP = 0, and H = 0.5$t_{c(CO)}$ (see Figure 6–20).

NOTE: For all SPI slave modes, CLKG is programmed as 1/2 of the CPU clock by setting CLKSM = CLKGDV = 1.

Table 6–22. McBSP as SPI Master or Slave Timing Requirements (CLKSTP = 11b and CLKXP = 0)

		MASTER		SLAVE		UNIT
		MIN	MAX	MIN	MAX	
$t_{su(BDRV-BCKXL)}$	Setup time, BDR valid before BCLKX low	12		– 12H		ns
$t_{h(BCKXH-BDRV)}$	Hold time, BDR valid after BCLKX high	0		12H + 5		ns
$t_{su(BFXL-BCKXH)}$	Setup time, BFSX low before BCLKX high			10		ns
$t_{c(BCKX)}$	Cycle time, BCLKX	12H		32H		ns

Table 6–23. McBSP as SPI Master or Slave Switching Characteristics (CLKSTP = 11b and CLKXP = 0)

PARAMETER		MASTER†		SLAVE		UNIT
		MIN	MAX	MIN	MAX	
$t_{h(BCKXL-BFXL)}$	Hold time, BFSX low after BCLKX low‡	T – 6	T + 6			ns
$t_{d(BFXL-BCKXH)}$	Delay time, BFSX low to BCLKX high§	C – 6	C + 6			ns
$t_{d(BCKXL-BDXV)}$	Delay time, BCLKX low to BDX valid	– 3	7	6H + 5	10H + 14	ns
$t_{dis(BCKXL-BDXHZ)}$	Disable time, BDX high impedance following last data bit from BCLKX low	– 2	4	6H + 3	10H + 17	ns
$t_{d(BFXL-BDXV)}$	Delay time, BFSX low to BDX valid	C – 6	C + 6	4H – 2	8H + 17	ns

† T = BCLKX period = (1 + CLKGDV) * 2H
C = BCLKX low pulse width = T/2 when CLKGDV is odd or zero and = (CLKGDV/2) * 2H when CLKGDV is even
D = BCLKX high pulse width = T/2 when CLKGDV is odd or zero and = (CLKGDV/2 + 1) * 2H when CLKGDV is even

‡ FSRP = FSXP = 1. As a SPI master, BFSX is inverted to provide active-low slave-enable output. As a slave, the active-low signal input on BFSX and BFSR is inverted before being used internally.
CLKXM = FSXM = 1, CLKRM = FSRM = 0 for master McBSP
CLKXM = CLKRM = FSXM = FSRM = 0 for slave McBSP

§ BFSX should be low before the rising edge of clock to enable slave devices and then begin a SPI transfer at the rising edge of the master clock (BCLKX).

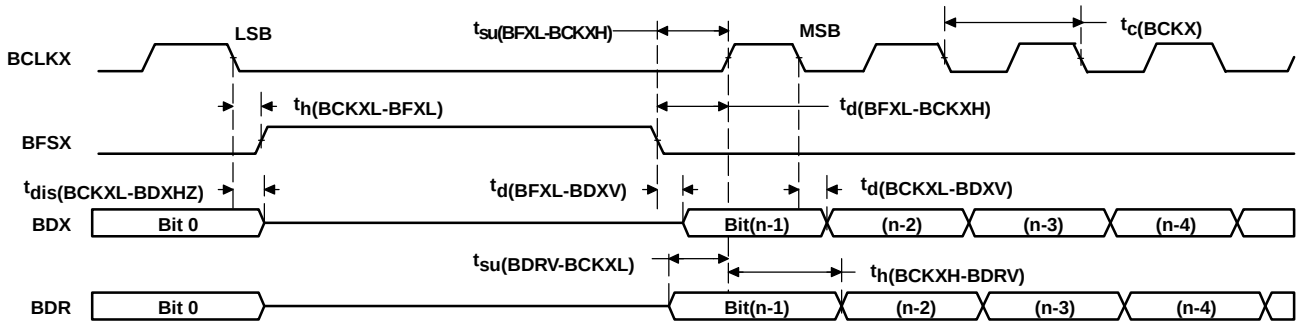


Figure 6–20. McBSP Timing as SPI Master or Slave: CLKSTP = 11b, CLKXP = 0

High inactive state without delay; the McBSP transmits data on the falling edge of BCLKX and receives data on the rising edge of BCLKR.

Table 6–24 and Table 6–25 assume testing over recommended operating conditions, CLKSTP = 10b, CLKXP = 1, and H = $0.5t_{c(CO)}$ (see Figure 6–21).

NOTE: For all SPI slave modes, CLKG is programmed as 1/2 of the CPU clock by setting CLKSM = CLKGDV = 1.

Table 6–24. McBSP as SPI Master or Slave Timing Requirements (CLKSTP = 10b and CLKXP = 1)

		MASTER		SLAVE		UNIT
		MIN	MAX	MIN	MAX	
$t_{su(BDRV-BCKXH)}$	Setup time, BDR valid before BCLKX high	12		– 12H		ns
$t_{h(BCKXH-BDRV)}$	Hold time, BDR valid after BCLKX high	0		12H + 5		ns
$t_{su(BFXL-BCKXL)}$	Setup time, BFSX low before BCLKX low			10		ns
$t_{c(BCKX)}$	Cycle time, BCLKX	12H		32H		ns

Table 6–25. McBSP as SPI Master or Slave Switching Characteristics (CLKSTP = 10b and CLKXP = 1)

PARAMETER		MASTER†		SLAVE		UNIT
		MIN	MAX	MIN	MAX	
$t_{h(BCKXH-BFXL)}$	Hold time, BFSX low after BCLKX high‡	T – 6	T + 6			ns
$t_{d(BFXL-BCKXL)}$	Delay time, BFSX low to BCLKX low§	C – 6	C + 6			ns
$t_{d(BCKXL-BDXV)}$	Delay time, BCLKX low to BDX valid	– 3	7	6H + 5	10H + 14	ns
$t_{dis(BCKXH-BDXHZ)}$	Disable time, BDX high impedance following last data bit from BCLKX high	D – 2	D + 3			ns
$t_{dis(BFXH-BDXHZ)}$	Disable time, BDX high impedance following last data bit from BFSX high			2H + 3	6H + 17	ns
$t_{d(BFXL-BDXV)}$	Delay time, BFSX low to BDX valid			4H – 2	8H + 17	ns

† T = BCLKX period = (1 + CLKGDV) * 2H
C = BCLKX low pulse width = T/2 when CLKGDV is odd or zero and = (CLKGDV/2) * 2H when CLKGDV is even

‡ FSRP = FSXP = 1. As a SPI master, BFSX is inverted to provide active-low slave-enable output. As a slave, the active-low signal input on BFSX and BFSR is inverted before being used internally.
CLKXM = FSXM = 1, CLKRM = FSRM = 0 for master McBSP
CLKXM = CLKRM = FSXM = FSRM = 0 for slave McBSP

§ BFSX should be low before the rising edge of clock to enable slave devices and then begin a SPI transfer at the rising edge of the master clock (BCLKX).

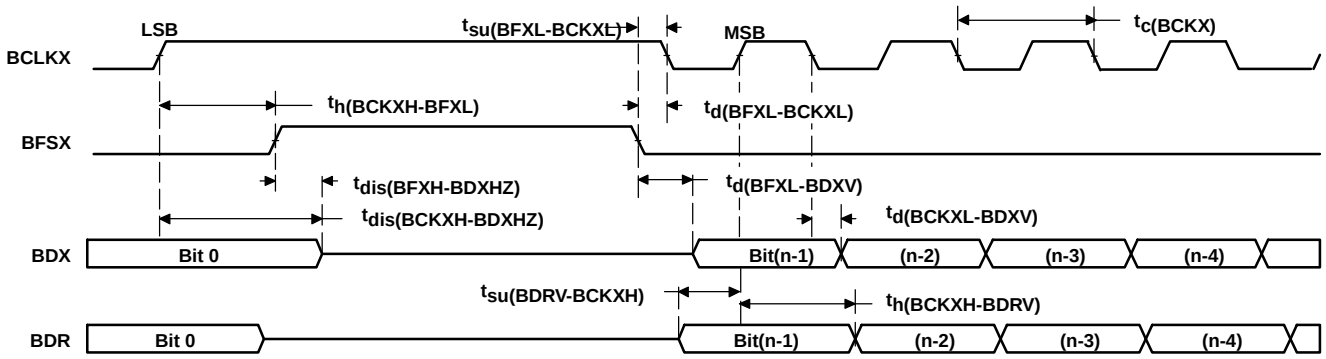


Figure 6–21. McBSP Timing as SPI Master or Slave: CLKSTP = 10b, CLKXP = 1

High inactive state with delay; the McBSP transmits data one-half cycle ahead of the falling edge of BCLKX and receives data on the falling edge of BCLKR.

Table 6–26 and Table 6–27 assume testing over recommended operating conditions, CLKSTP = 11b, CLKXP = 1, and H = 0.5$t_{c(CO)}$ (see Figure 6–22).

NOTE: For all SPI slave modes, CLKG is programmed as 1/2 of the CPU clock by setting CLKSM = CLKGDV = 1.

Table 6–26. McBSP as SPI Master or Slave Timing Requirements (CLKSTP = 11b and CLKXP = 1)

		MASTER		SLAVE		UNIT
		MIN	MAX	MIN	MAX	
$t_{su(BDRV-BCKXL)}$	Setup time, BDR valid before BCLKX low	12		– 12H		ns
$t_{h(BCKXL-BDRV)}$	Hold time, BDR valid after BCLKX low	0		12H + 5		ns
$t_{su(BFXL-BCKXL)}$	Setup time, BFSX low before BCLKX low			10		ns
$t_{c(BCKX)}$	Cycle time, BCLKX	12H		32H		ns

Table 6–27. McBSP as SPI Master or Slave Switching Characteristics (CLKSTP = 11b and CLKXP = 1)

PARAMETER		MASTER†		SLAVE		UNIT
		MIN	MAX	MIN	MAX	
$t_{h(BCKXH-BFXL)}$	Hold time, BFSX low after BCLKX high‡	T – 6	T + 6			ns
$t_{d(BFXL-BCKXL)}$	Delay time, BFSX low to BCLKX low§	C – 6	C + 6			ns
$t_{d(BCKXH-BDXV)}$	Delay time, BCLKX high to BDX valid	– 3	7	6H + 5	10H + 14	ns
$t_{dis(BCKXH-BDXHZ)}$	Disable time, BDX high impedance following last data bit from BCLKX high	– 2	4	6H + 3	10H + 17	ns
$t_{d(BFXL-BDXV)}$	Delay time, BFSX low to BDX valid	C – 6	C + 6	4H – 2	8H + 17	ns

† T = BCLKX period = (1 + CLKGDV) * 2H
C = BCLKX low pulse width = T/2 when CLKGDV is odd or zero and = (CLKGDV/2) * 2H when CLKGDV is even
D = BCLKX high pulse width = T/2 when CLKGDV is odd or zero and = (CLKGDV/2 + 1) * 2H when CLKGDV is even

‡ FSRP = FSXP = 1. As a SPI master, BFSX is inverted to provide active-low slave-enable output. As a slave, the active-low signal input on BFSX and BFSR is inverted before being used internally.
CLKXM = FSXM = 1, CLKRM = FSRM = 0 for master McBSP
CLKXM = CLKRM = FSXM = FSRM = 0 for slave McBSP

§ BFSX should be low before the rising edge of clock to enable slave devices and then begin a SPI transfer at the rising edge of the master clock (BCLKX).

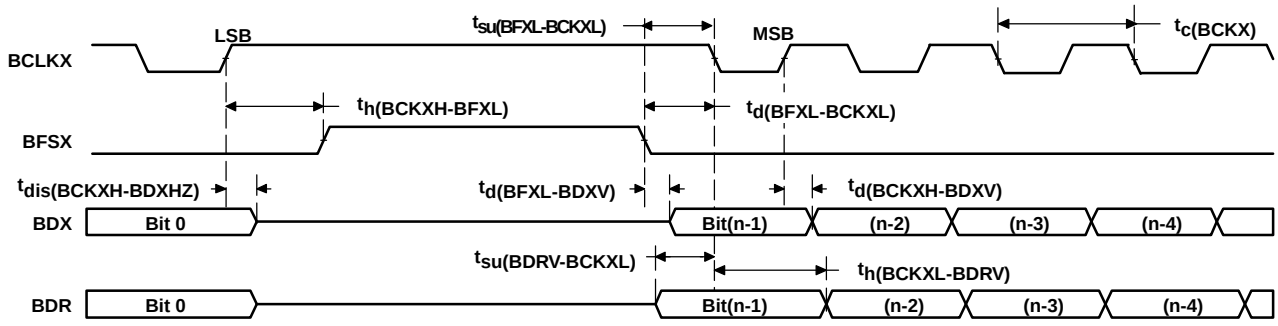


Figure 6–22. McBSP Timing as SPI Master or Slave: CLKSTP = 11b, CLKXP = 1

6.12 Synchronous DRAM Timings

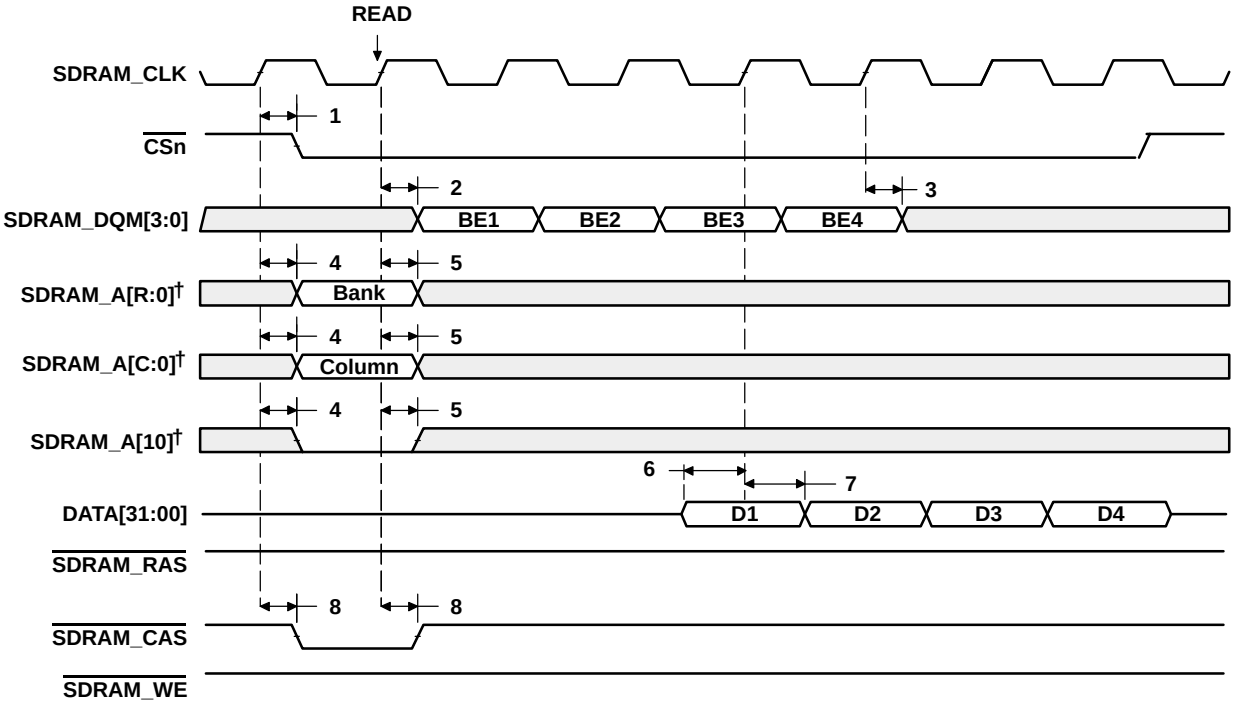
Table 6–28 and Table 6–29 assume testing over recommended operating conditions (see Figure 6–23 through Figure 6–29).

Table 6–28. Synchronous DRAM Timing Requirements

NO.			MIN	MAX	UNIT
6	$t_{su(DV-SCLKH)}$	Setup time, read data valid before SDRAM_CLK high	4.5		ns
7	$t_{h(SCLKH-DV)}$	Hold time, read data valid after SDRAM_CLK high	2		ns

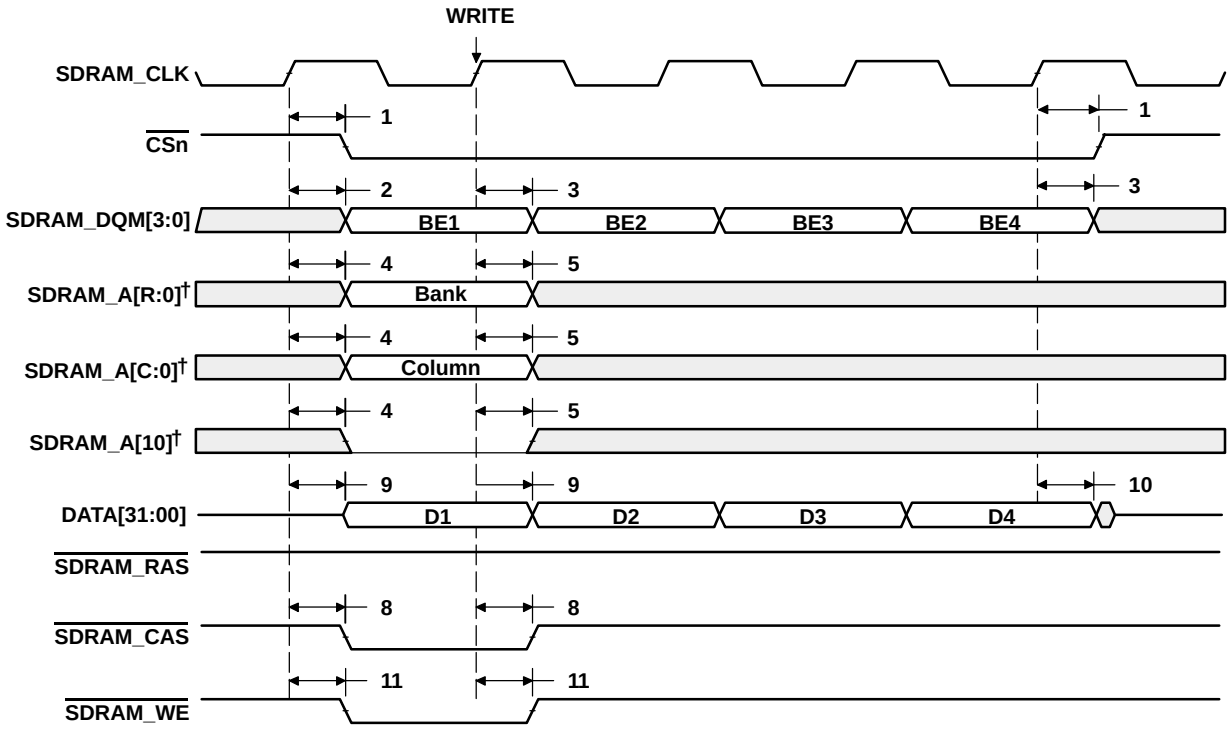
Table 6–29. Synchronous DRAM Switching Characteristics

NO.	PARAMETER		MIN	MAX	UNIT
1	$t_{d(SCLKH-CSV)}$	Delay time, SDRAM_CLK high to chip select valid	0.5	11	ns
2	$t_{d(SCLKH-DQMV)}$	Delay time, SDRAM_CLK high to byte enable valid	0.5	11	ns
3	$t_{d(SCLKH-DQMIV)}$	Delay time, SDRAM_CLK high to byte enable invalid	0.5	11	ns
4	$t_{d(SCLKH-AV)}$	Delay time, SDRAM_CLK high to address valid	0.5	11	ns
5	$t_{d(SCLKH-AIV)}$	Delay time, SDRAM_CLK high to address invalid	0.5	11	ns
8	$t_{d(SCLKH-CASV)}$	Delay time, SDRAM_CLK high to $\overline{\text{SDRAM\_CAS}}$ valid	0.5	11	ns
9	$t_{d(SCLKH-DV)}$	Delay time, SDRAM_CLK high to data valid	0.5	11	ns
10	$t_{d(SCLKH-DIV)}$	Delay time, SDRAM_CLK high to data invalid	0.5	11	ns
11	$t_{d(SCLKH-WEV)}$	Delay time, SDRAM_CLK high to $\overline{\text{SDRAM\_WE}}$ valid	0.5	11	ns
12	$t_{d(SCLKH-RASV)}$	Delay time, SDRAM_CLK high to, $\overline{\text{SDRAM\_RAS}}$ valid	0.5	11	ns



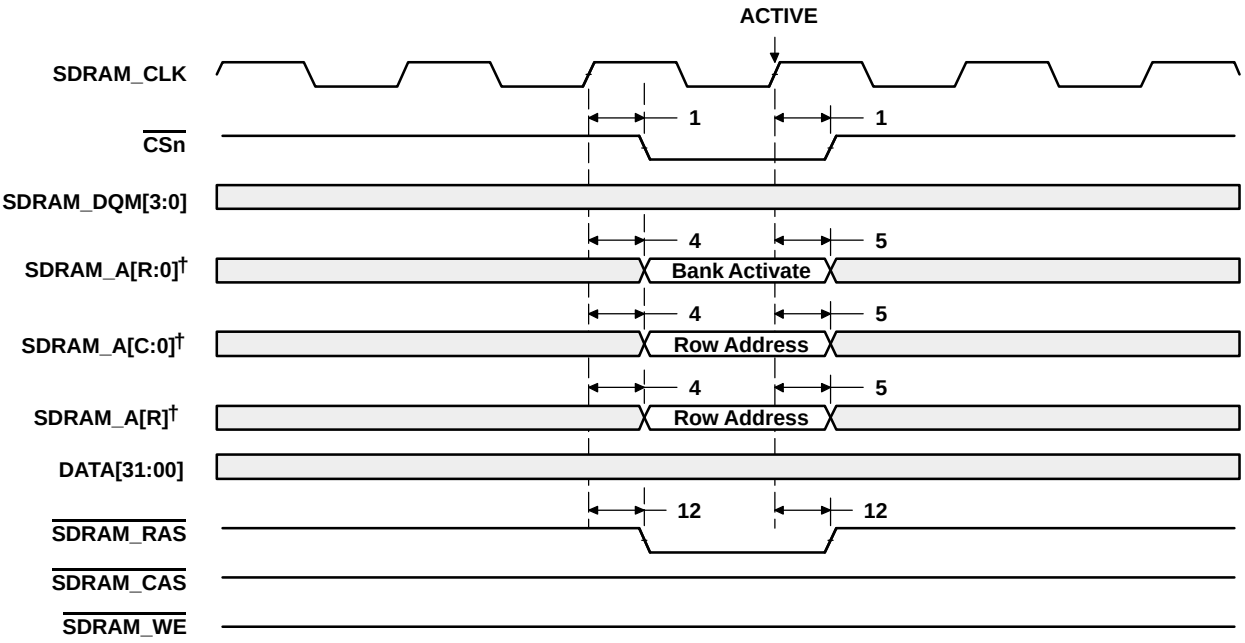
† The values for R and C depend on the particular size of the SRAMs used.

Figure 6–23. SDRAM Read Command (CAS Latency 3)



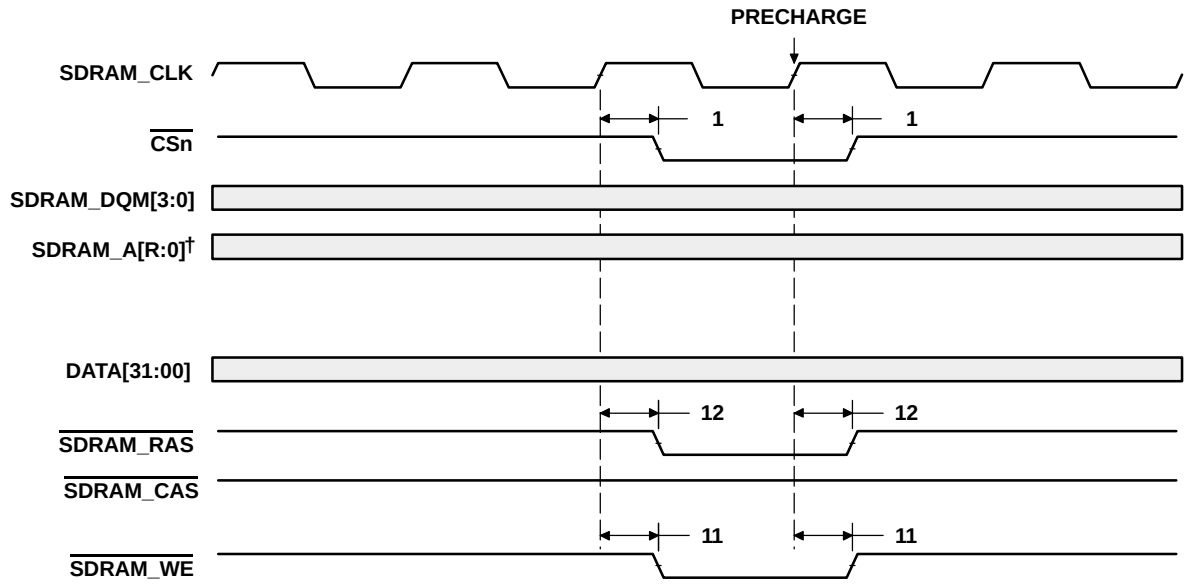
† The values for R and C depend on the particular size of the SRAMs used.

Figure 6–24. SDRAM Write Command



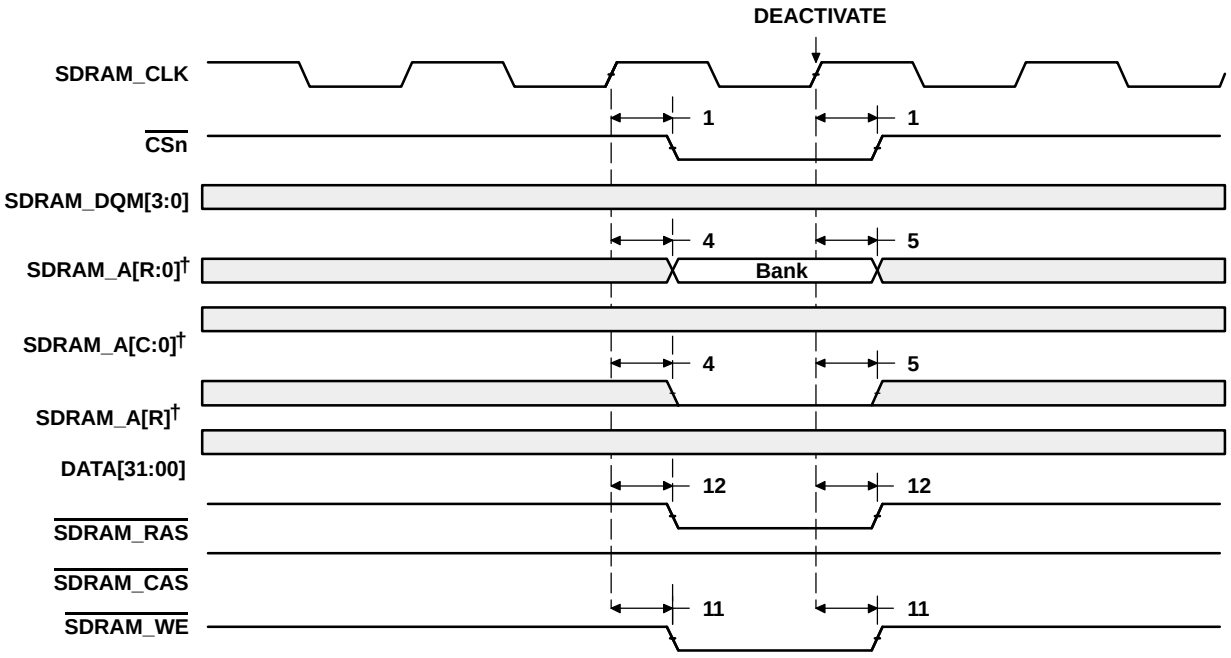
† The values for R and C depend on the particular size of the SRAMs used.

Figure 6–25. SDRAM Active Command



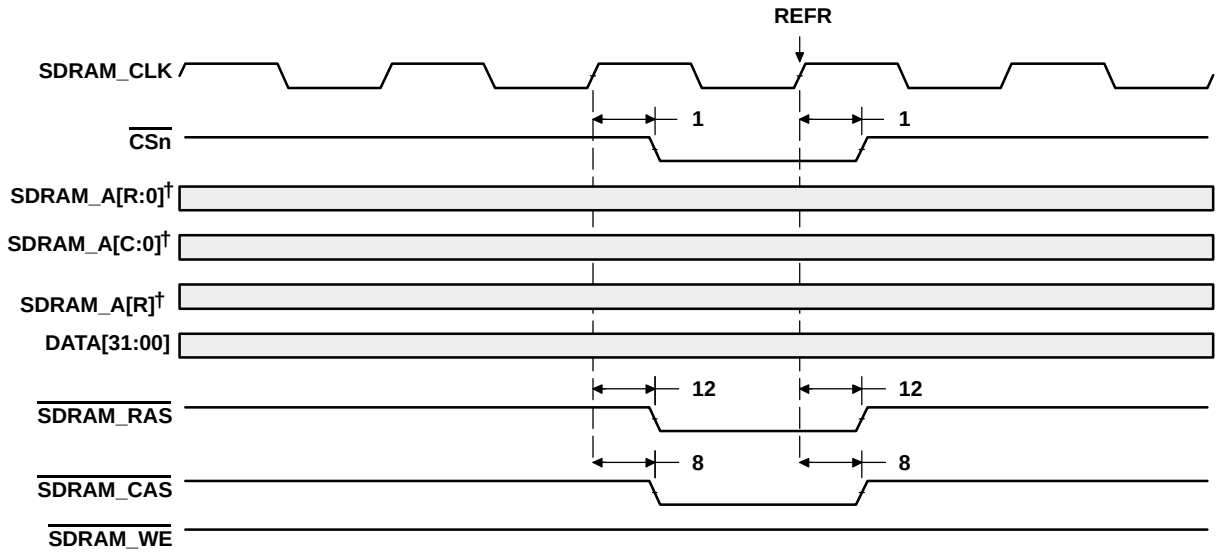
† The values for R and C depend on the particular size of the SRAMs used.

Figure 6–26. SDRAM Precharge Command



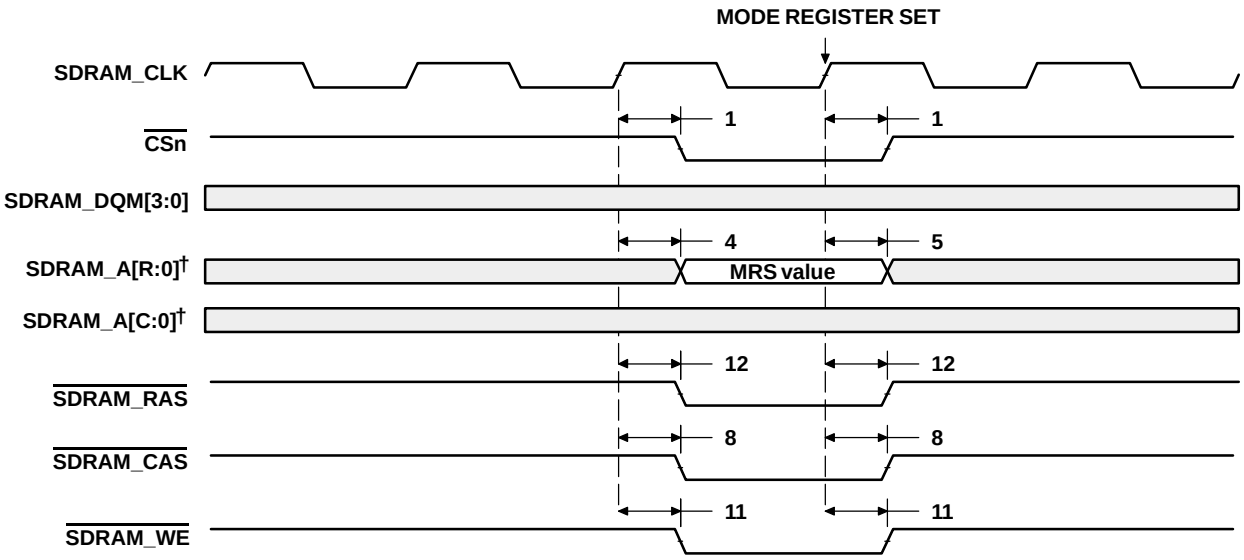
† The values for R and C depend on the particular size of the SRAMs used.

Figure 6–27. SDRAM Deactivate Command



† The values for R and C depend on the particular size of the SRAMs used.

Figure 6–28. SDRAM Refresh Command



† The values for R and C depend on the particular size of the SRAMs used.

Figure 6–29. SDRAM Mode Register Set Command

6.13 I²C Bus Timings

Table 6–30 through Table 6–32 assume testing over recommended operating conditions (see Figure 6–30 through Figure 6–32).

Table 6–30. I²C Bus Device Switching Characteristics of the SDA and SCL Bus Lines

PARAMETER		STANDARD MODE		FAST MODE		UNIT
		MIN	MAX	MIN	MAX	
$t_{c(SCL)}$	Cycle time, SCL	10		2.5		µs
$t_{w(SCLH)}$	Pulse duration, SCL high	4		0.6		µs
$t_{w(SCLL)}$	Pulse duration, SCL low	4.7		1.3		µs
$t_{r(SCL)}$	Rise time, SCL		1000		300	ns
$t_{f(SCL)}$	Fall time, SCL		300		300	ns
$t_{r(SDA)}$	Rise time, SDA		1000		300	ns
$t_{f(SDA)}$	Fall time, SDA		300		300	ns
$t_{w(SP)}$	Pulse duration, spike suppression			0	50	ns
$t_{su(SDA–SCLH)}$	Setup time, SCL high after SDA valid	250		100		ns
$t_{h(SDA–SCLL)}$	Hold time, SCL low after SDA invalid	0	3.45	0	0.9	µs

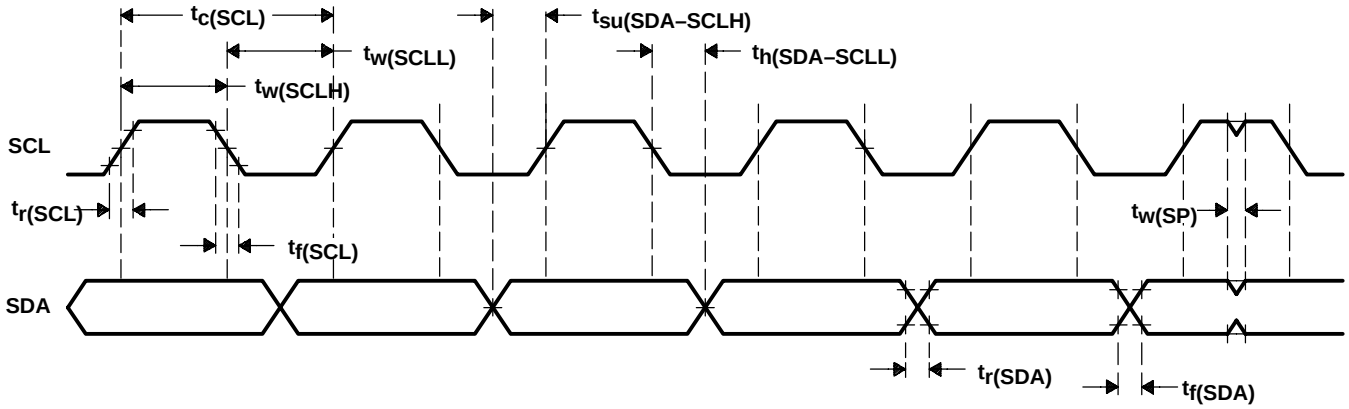


Figure 6–30. Definition of Timing on the I²C Bus

Table 6–31. I²C Bus Device Timing Requirements (STOP and START Conditions)

		STANDARD MODE		FAST MODE		UNIT
		MIN	MAX	MIN	MAX	
$t_{h(SDAL–SCLL1)}$	Hold time, SCL low from SDA low (START condition)	4		0.6		µs
$t_{su(SCLH-SDAH)}$	Setup time, SDA high from SCL high (STOP condition)	4		0.6		µs
$t_{w(SDAH)}$	Pulse duration, SDA high. Bus free time between a STOP and START condition.	4.7		1.3		µs

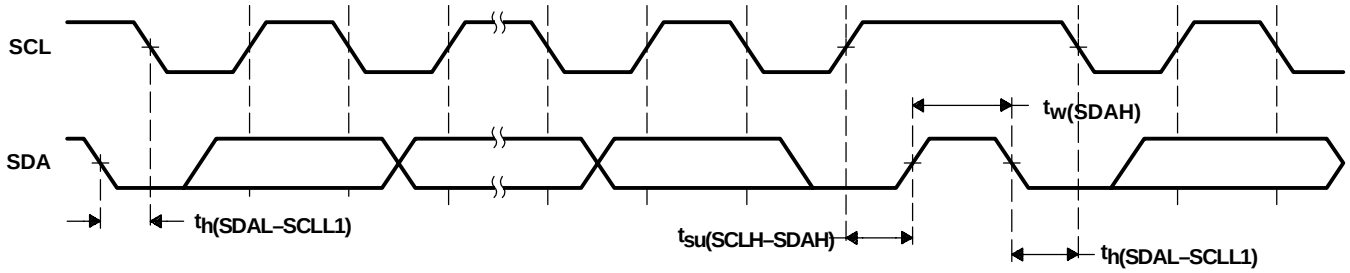


Figure 6–31. I²C Bus Timings (STOP and START Conditions)

Table 6–32. I²C Bus Device Timing Requirements (Repeated START Condition)

		STANDARD MODE		FAST MODE		UNIT
		MIN	MAX	MIN	MAX	
$t_{su(SCLH–SDAL)}$	Setup time, SDA low after SCL high	4.7		0.6		µs
$t_{d(SDAL–SCLL2)}$	Hold time, SCL low after SDA low	4		0.6		µs

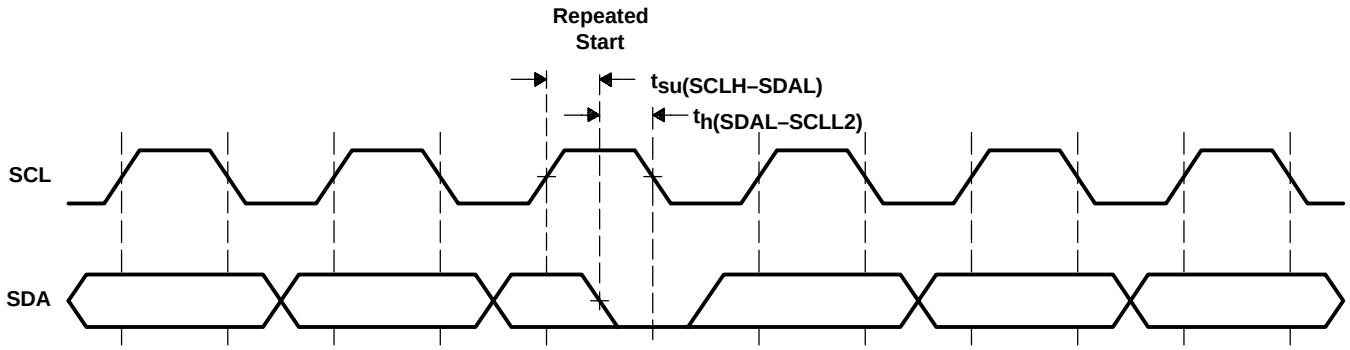


Figure 6–32. I²C Bus Timings (Repeated START Condition)

6.14 MII Timings

Table 6–33 and Table 6–34 assume testing over recommended operating conditions (see Figure 6–33 and Figure 6–34).

Table 6–33. MII Timing Requirements (Receive)

			MIN†	MAX	UNIT
1	$t_{su(TCLKH-RXD)}$	Setup time, read RXD[3:0] before TCLK high	10/10		ns
	$t_{su(TCLKH-RXER)}$	Setup time, read RX error valid before TCLK high	10/10		
	$t_{su(TCLKH-RXDV)}$	Setup time, read RX data valid before TCLK high	10/10		
2	$t_{hd(RXD-TCLKH)}$	Hold time, read RXD[3:0] after TCLK high	10/10		ns
	$t_{hd(RXER-TCLKH)}$	Hold time, read RX error valid after TCLK high	10/10		
	$t_{hd(RXDV-TCLKH)}$	Hold time, read RX data valid after TCLK high	10/10		
3	$t_{w(TCLKH)}$	Pulse duration, TCLK high	140/14		ns
4	$t_{w(TCLKL)}$	Pulse duration, TCLK low	140/14		ns
5	$t_{w(TCLK)}$	Cycle time, TCLK	400/40		ns

† The timings format is based on 10 Mb/s / 100 Mb/s.

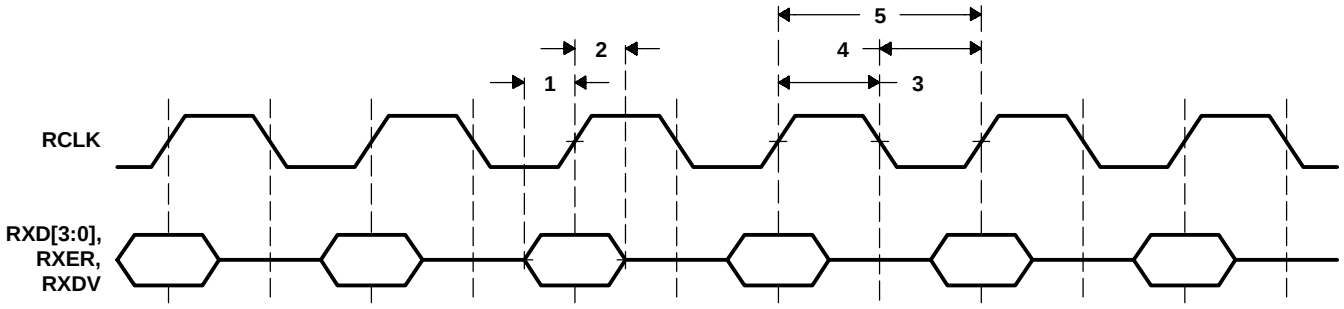


Figure 6–33. MII Receive Timing

Table 6–34. MII Timing Requirements (Transmit)

			MIN	MAX	UNIT
6	$t_{d(RCLKH-TXD)}$	Delay time, RCLK high to TXD valid	1/1	30/30	ns
	$t_{d(RCLKH-TXER)}$	Delay time, RCLK high to TXER valid	1/1	30/30	
	$t_{d(RCLKH-TXEN)}$	Delay time, RCLK high to TXEN valid	1/1	30/30	
7	$t_{h(RCLKH-TXD)}$	Hold time, TXD valid before RCLK high	0/0	25/25	ns
	$t_{h(RCLKH-TXER)}$	Hold time, TXER valid before RCLK high	0/0	25/25	
	$t_{h(RCLKH-TXEN)}$	Hold time, TXEN valid before RCLK high	0/0	25/25	

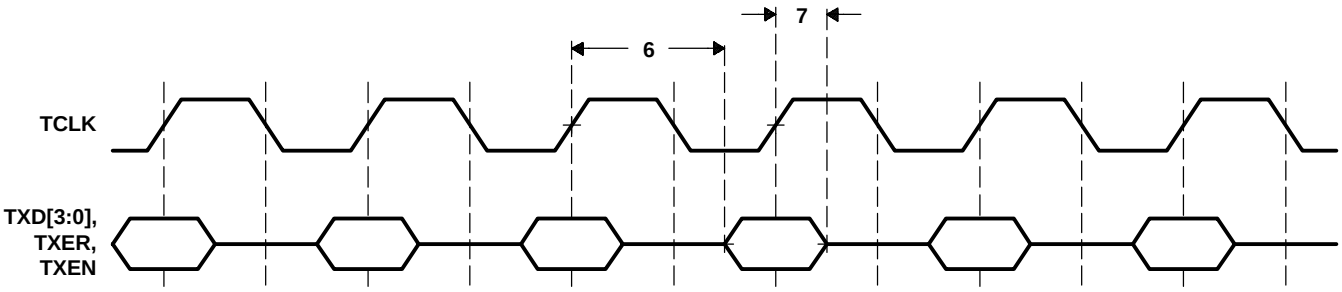


Figure 6–34. MII Transmit Timing

6.15 ARM Clock Timings

The ARM SRAM timings are derived from the ARM clock. The clock switching characteristics are defined in Table 6–35 for use in the ARM SRAM timing definitions.

Table 6–35. ARM SRAM Switching Characteristics

	PARAMETER		MIN	TYP	MAX	UNIT
A	$t_{c(ARM)}$	Cycle time, ARM clock	21.05			ns
B	$t_{w(ARM–H)}$	Pulse duration, ARM clock high	A/2 – 3†	A/2†	A/2 + 3†	ns
C	$t_{w(ARM–L)}$	Pulse duration, ARM clock low	A/2 – 3†	A/2†	A/2 + 3†	ns
D	$t_{w(ARM–W)}$	Pulse duration, ARM write cycle		A/2 + nA†‡		ns
E	$t_{w(ARM–R)}$	Pulse duration, ARM read cycle		A + nA†‡		ns

† A = $t_{c(ARM)}$
‡ n = the number of software wait states

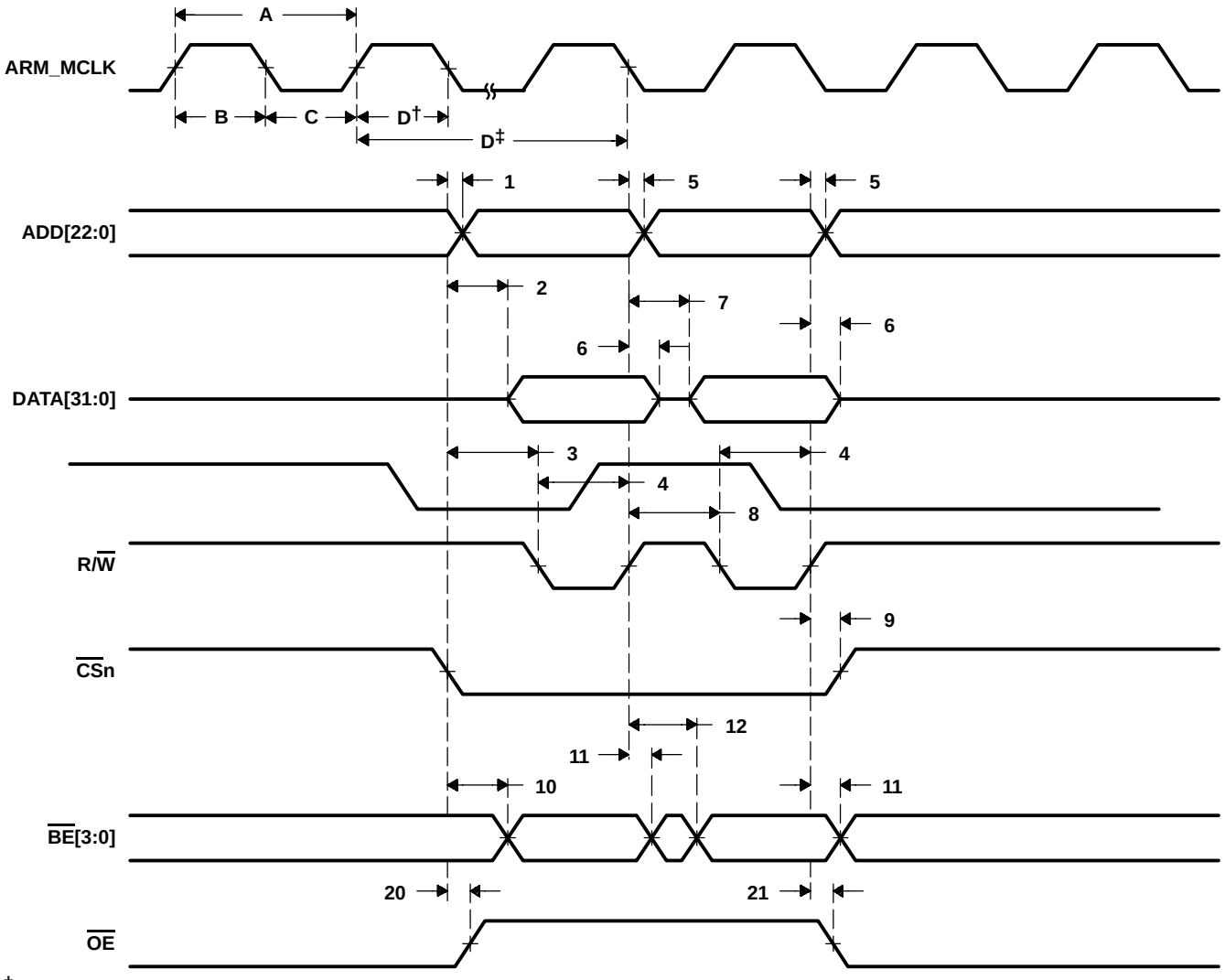
The ARM SRAM write timing requirements listed in Table 6–36 represent back-to-back write operations to a zero-wait-state SRAM device.

Table 6–36 through Table 6–37 assume testing over recommended operating conditions (see Figure 6–35 and Figure 6–36).

Table 6–36. ARM SRAM Timing Requirements (Write)

			MIN	NOM	MAX	UNIT
1	$t_{d(CSL-AV)}$	Delay time, address valid from $\overline{CS}$ low	– 3		4	ns
2	$t_{d(CSL-DV)}$	Delay time, data valid from $\overline{CS}$ low	–2		5	ns
3	$t_{d(CSL-WEL)}$	Delay time, write enable low from $\overline{CS}$ low	4		13	ns
4	$t_{w(WEL)}$	Pulse width, write enable low	D – 3.5§	D§	D + 3.5§	ns
5	$t_{d(WEH–AIV)}$	Delay time, address invalid from write enable high	0		7	ns
6	$t_{d(WEH–DIV)}$	Delay time, data invalid from write enable high	0		7	ns
7	$t_{d(WEH–DV)}$	Delay time, data valid from write enable high	2		9	ns
8	$t_{w(WEH)}$	Pulse width, write enable high	A/2 – 3.5†	A/2†	A/2 + 3.5†	ns
9	$t_{d(WEH–CSH)}$	Delay time, $\overline{CS}$ high from write enable high	0		7	ns
10	$t_{d(CSL–BEV)}$	Delay time, byte enable valid from $\overline{CS}$ low	– 3		4	ns
11	$t_{d(WEH–BEIV)}$	Delay time, byte enable invalid from write enable high	0		7	ns
12	$t_{d(WEH–BEV)}$	Delay time, byte enable valid from write enable high	2		9	ns
20	$t_{d(CSL-OEH)}$	Delay time, output enable high from $\overline{CS}$ low	– 3		4	ns
21	$t_{d(WEH–OEL)}$	Delay time, output enable low from write enable high	0		7	ns

† A = $t_{c(ARM)}$
§ D = $t_{w(ARM-W)}$



† n = 0 wait states
‡ n = 1 or more wait states

NOTE: When performing a 32-bit access on a 16-bit-wide SRAM, two of the byte-enables will be active at the same time. The particular byte-enables active depend on the endiannism selected.

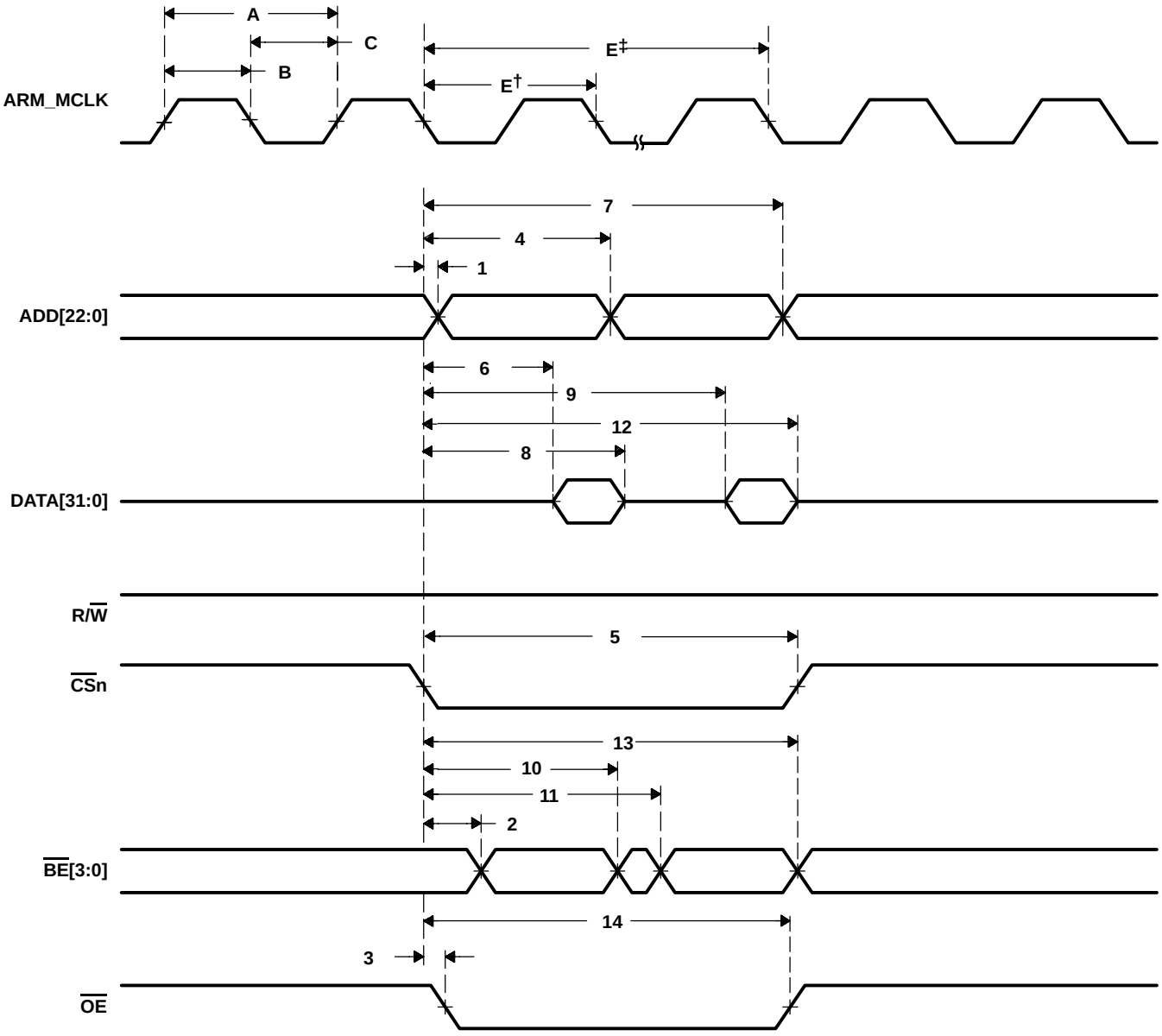
Figure 6–35. ARM SRAM Write Timing

The ARM SRAM read timing requirements listed in Table 6–37 represent back-to-back read operations to a zero-wait-state SRAM device.

Table 6–37. ARM SRAM Timing Requirements (Read)

			MIN	MAX	UNIT
1	$t_{d(CSL-AV)}$	Delay time, address valid from $\overline{CS}$ low	– 3	4	ns
2	$t_{d(CSL–BHEV1)}$	Delay time, byte enable high valid 1 from $\overline{CS}$ low	– 3	4	ns
3	$t_{d(CSL-OEL)}$	Delay time, output enable low from $\overline{CS}$ low	– 3	4	ns
4	$t_{d(CSL-AIV1)}$	Delay time, address invalid 1 from $\overline{CS}$ low	E – 3†	E + 4†	ns
5	$t_{w(CSL)}$	Pulse duration, $\overline{CS}$ low	2E – 3.5†	2E + 3.5†	ns
6	$t_{su(CSL-DV1)}$	Setup time, data valid 1 from $\overline{CS}$ low	E – 12†		ns
7	$t_{d(CSL–AIV2)}$	Delay time, address invalid 2 from $\overline{CS}$ low	2E – 3†	2E + 4†	ns
8	$t_{h(CSL–DIV1)}$	Hold time, data invalid 1 from $\overline{CS}$ low	E – 1†		ns
9	$t_{su(CSL–DV2)}$	Setup time, data valid 2 from $\overline{CS}$ low	2E – 12†		ns
10	$t_{d(CSL–BEIV1)}$	Delay time, byte enable invalid 1 from $\overline{CS}$ low	E – 4†	E + 4†	ns
11	$t_{d(CSL–BEV2)}$	Delay time, byte enable valid 2 from $\overline{CS}$ low	E – 2†	E + 6†	ns
12	$t_{h(CSL–DIV2)}$	Hold time, data invalid 2 from $\overline{CS}$ low	2E – 1†		ns
13	$t_{d(CSL–BEIV2)}$	Delay time, byte enable invalid 2 from $\overline{CS}$ low	2E – 4†	2E + 4†	ns
14	$t_{d(CSL–OEH)}$	Delay time, output enable high from $\overline{CS}$ low	2E – 3†	2E + 4†	ns

† $E = t_{w(ARM–R)}$



† n = 0 wait states
‡ n = 1 or more wait states

NOTE: When performing a 32-bit access on a 16-bit-wide SRAM, two of the byte-enables will be active at the same time. The particular byte-enables active depend on the endiannism selected.

Figure 6–36. ARM SRAM Read Timing

6.16 SPI Clock Timings

SPI timings are derived from the internal SPI clock. This clock is only available externally during a serial transfer.

Table 6–38 through Table 6–40 assume testing over recommended operating conditions (see Figure 6–37 and Figure 6–38).

Table 6–38. SPI Clock Switching Characteristics

PARAMETER			MIN	TYP	MAX	UNIT
$t_{c(SPI)}$	Cycle time, CLKX_SPI		168.4	A†		ns
$t_{w(SPI–H)}$	Pulse duration, CLKX_SPI high		A/2 – 3†	A/2†	A/2 + 3†	ns
$t_{w(SPI–L)}$	Pulse duration, CLKX_SPI low		A/2 – 3†	A/2†	A/2 + 3†	ns
$t_{r(SPI)}$	Rise time, SPI output signals.	CLKX_SPI, MCUDO, MCUEN0, MCUEN1, and MCUEN2	1		4	ns
$t_{f(SPI)}$	Fall time, SPI output signals.	CLKX_SPI, MCUDO, MCUEN0, MCUEN1, and MCUEN2	1		4	ns

† A = 4 x PTV x $t_{c(ARM)}$

Table 6–39. SPI Falling Edge Timing Requirements

		MIN	NOM	MAX	UNIT
$t_{d(ENFS–CLKH)}$	Delay time, Falling enable start active to first clock high	A/2 – 3†	A/2†	A/2 + 3†	ns
$t_{d(CLKL–ENFE)}$	Delay time, Last clock low to falling enable end	A – 3†	A†	A + 3†	ns
$t_{w(ENFT)}$	Pulse duration, Falling enable toggle	A – 3†	A†	A + 3†	ns
$t_{d(CLKH–DOV)}$	Delay time, Clock high to data out valid	2		8	ns
$t_{su(DIV–CLKL)}$	Setup time, Data in valid to clock low	6			ns
$t_{h(CLKL–DIIV)}$	Hold time, Clock low to data in invalid	4			ns

† A = 4 x PTV x $t_{c(ARM)}$

In this mode, all timing related activity for Data In and Enable is based on the falling edge of the SPI Clock. The timing activity for Data Out will be based on the rising edge of the SPI clock.

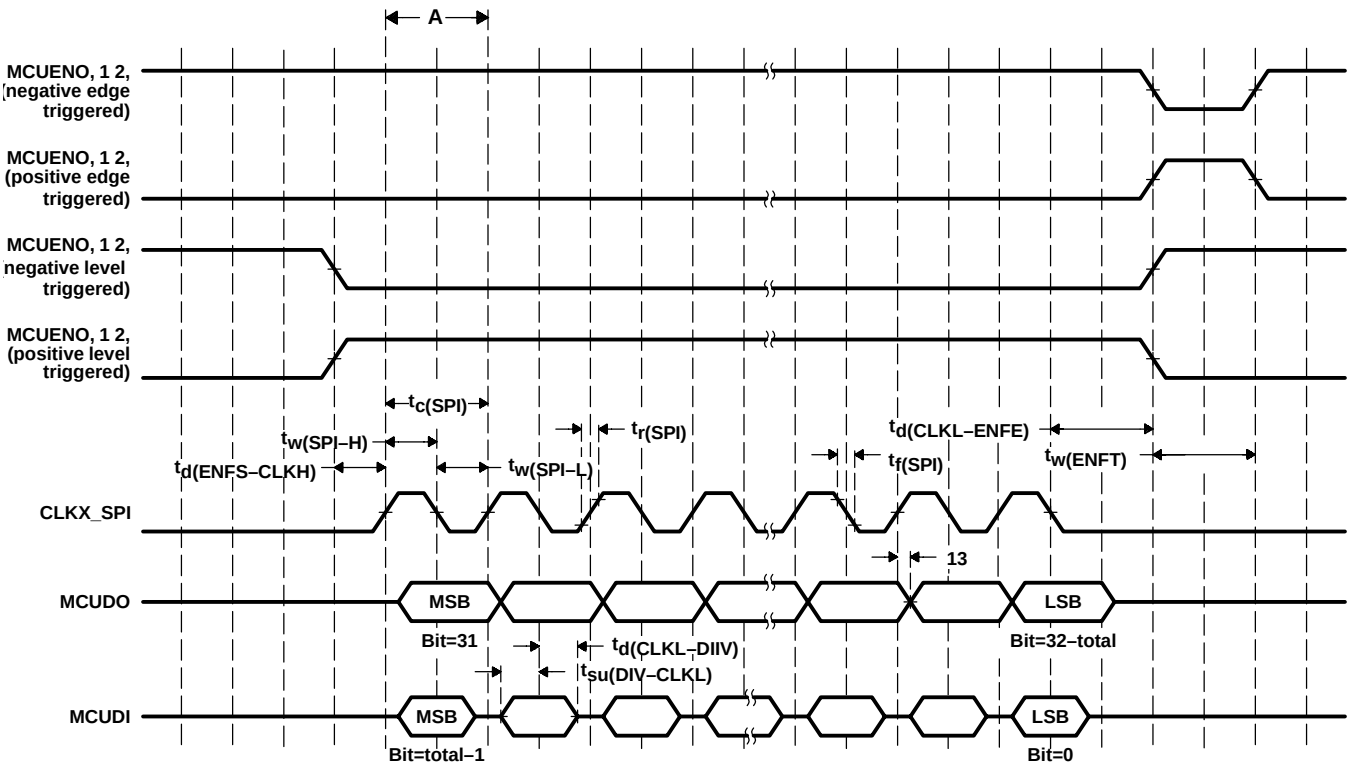


Figure 6–37. SPI Falling Edge Timings

Table 6–40. SPI Rising Edge Timing Requirements

		MIN	NOM	MAX	UNIT
$t_{d(ENRS–CLKH)}$	Delay time, Rising enable start active to first clock high	A – 3†	A†	A + 3†	ns
$t_{d(CLKL–ENRE)}$	Delay time, Last clock low to rising enable end	A/2 – 3†	A/2†	A/2 + 3†	ns
$t_{w(ENRT)}$	Pulse duration, Rising enable toggle	A – 3†	A†	A + 3†	ns
$t_{d(CLKL–DOV)}$	Delay time, Clock low to data out valid	2		8	ns
$t_{su(DIV–CLKH)}$	Setup time, Data in valid to clock high	6			ns
$t_{h(CLKH–DIIV)}$	Hold time, Clock high to data in invalid	4			ns

† A = 4 x PTV x $t_{c(ARM)}$

In this mode, all timing related activity for Data In and Enable is based on the falling edge of the SPI Clock. The timing activity for Data Out will be based on the rising edge of the SPI clock.

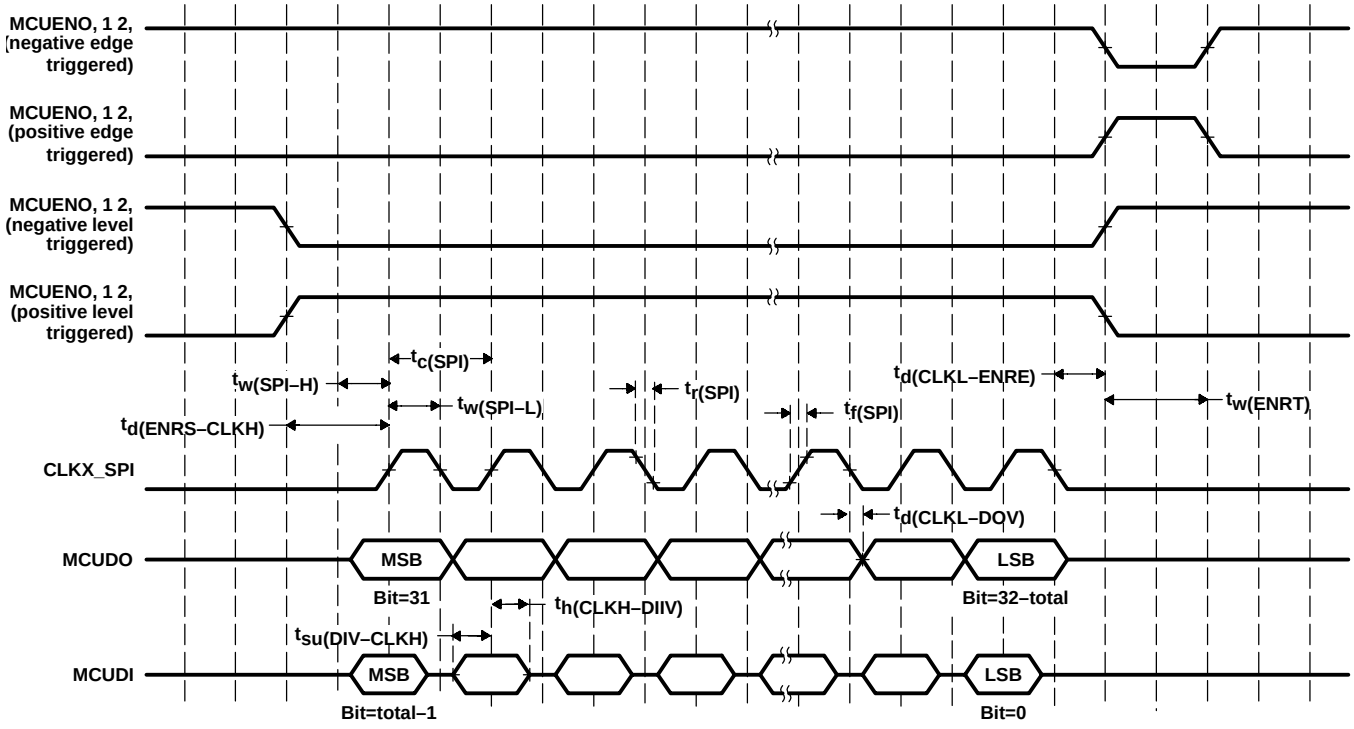
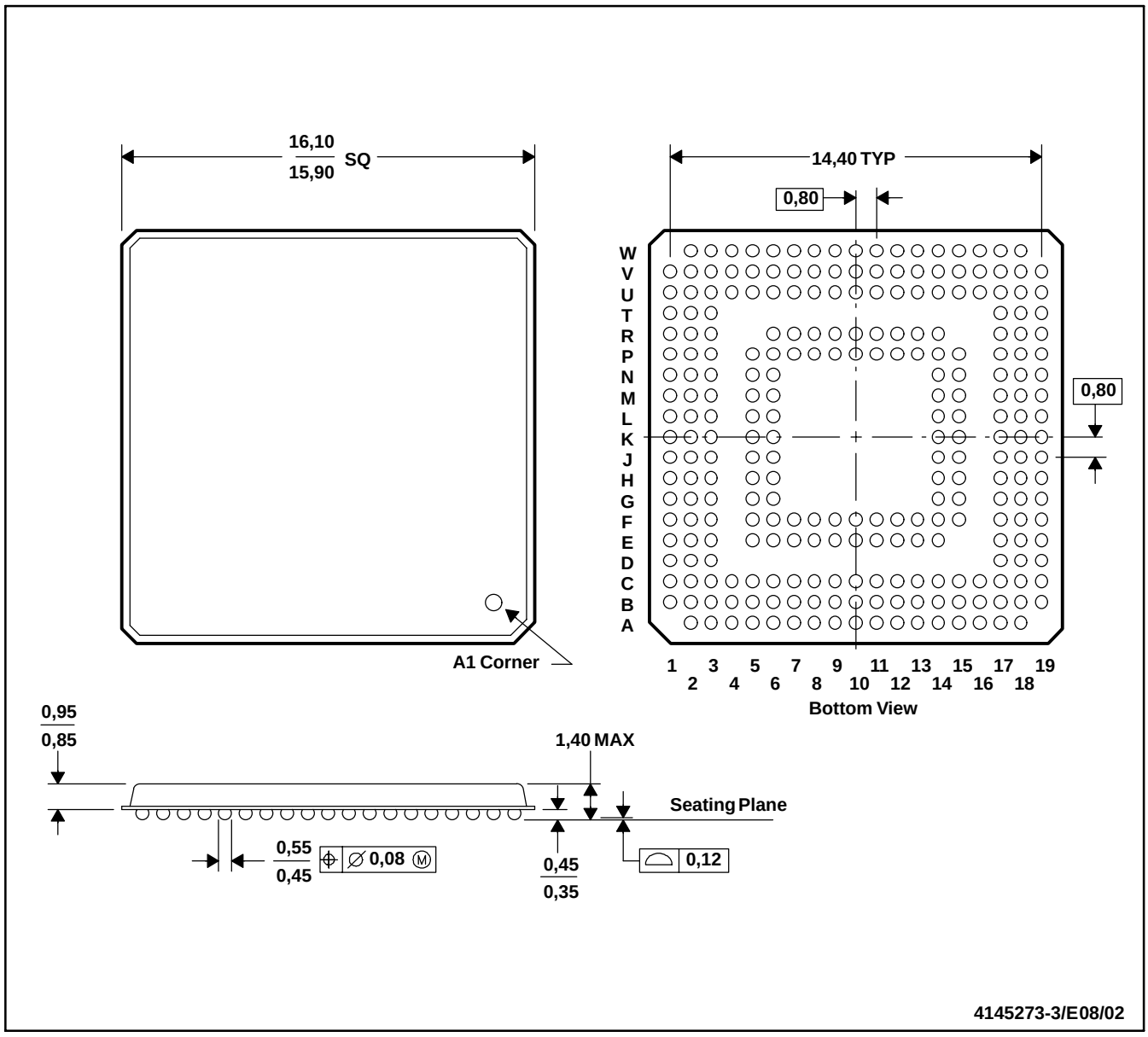


Figure 6–38. SPI Rising Edge Timings

7 Mechanical Data

7.1 Ball Grid Array Mechanical Data

GHK (S–PBGA–N257) **PLASTIC BALL GRID ARRAY**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice
C. MicroStar BGA™ configuration

Figure 7–1. TMS320VC5471 257-Ball MicroStar BGA™ Plastic Ball Grid Array Package

MicroStar BGA is a trademark of Texas Instruments.