Welcome to E-XFL.COM

Understanding Embedded - DSP (Digital Signal Processors)

Embedded - DSP (Digital Signal Processors) are specialized microprocessors designed to perform complex mathematical computations on digital signals in real-time. Unlike general-purpose processors, DSPs are optimized for high-speed numeric processing tasks, making them ideal for applications that require efficient and precise manipulation of digital data. These processors are fundamental in converting and processing signals in various forms, including audio, video, and communication signals, ensuring that data is accurately interpreted and utilized in embedded systems.

Applications of Embedded - DSP (Digital Signal Processors)

Details	
Product Status	Obsolete
Type	Fixed Point
Interface	Host Interface, SPI, SSP, UART
Clock Rate	140MHz
Non-Volatile Memory	ROM (48kB)
On-Chip RAM	80kB
Voltage - I/O	3.30V
Voltage - Core	2.50V
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/adsp-2195mbst-140

TABLE OF CONTENTS

General Note
This data sheet provides preliminary information for the ADSP-2195 Digital Signal Processor.

GENERAL DESCRIPTION

The ADSP-2195 DSP is a single-chip microcomputer optimized for digital signal processing (DSP) and other high speed numeric processing applications.

The ADSP-2195 combines the ADSP-219x family base architecture (three computational units, two data address generators, and a program sequencer) with three serial ports, two SPI-compatible ports, one UART port, a DMA controller, three programmable timers, general-purpose Programmable Flag pins, extensive interrupt capabilities, and on-chip program and data memory spaces.

The ADSP-2195 architecture is code-compatible with ADSP-218x family DSPs. Although the architectures are compatible, the ADSP-2195 architecture has a number of enhancements over the ADSP-218x architecture. The enhancements to computational units, data address generators, and program sequencer make the ADSP-2195 more flexible and even easier to program than the ADSP-218x DSPs.

Indirect addressing options provide addressing flexibility—premodify with no update, pre- and post-modify by an immediate 8-bit, two's-complement value and base address registers for easier implementation of circular buffering.

The ADSP-2195 integrates 48K words of on-chip memory configured as 16K words (24-bit) of program RAM, 16K words (16-bit) of data RAM, and 16K words (24-bit) of program ROM. Power-down circuitry is also provided to meet the low power needs of battery-operated portable equipment. The ADSP-2195 is available in 144-lead LQFP and mini-BGA packages.

Fabricated in a high-speed, low-power, CMOS process, the ADSP-2195 operates with a 6.25 ns instruction cycle time (160 MIPS). All instructions, except two multiword instructions, can execute in a single DSP cycle.

The ADSP-2195's flexible architecture and comprehensive instruction set support multiple operations in parallel. For example, in one processor cycle, the ADSP-2195 can:

- Generate an address for the next instruction fetch
- Fetch the next instruction
- Perform one or two data moves
- Update one or two data address pointers
- Perform a computational operation

These operations take place while the processor continues to:

- Receive and transmit data through two serial ports
- Receive and/or transmit data from a Host
- Receive or transmit data through the UART
- Receive or transmit data over two SPI ports
- Access external memory through the external memory interface
- Decrement the timers

DSP Core Architecture
The ADSP-2195 instruction set provides flexible data moves and multifunction (one or two data moves with a computation) instructions. Every single-word instruction can be executed in a single processor cycle. The ADSP-2195 assembly language uses an algebraic syntax for ease of coding and readability. A comprehensive set of development tools supports program development.

The functional block diagram on page 1 shows the architecture of the ADSP-219x core. It contains three independent computational units: the ALU, the multiplier/accumulator (MAC), and the shifter. The computational units process 16-bit data from the register file and have provisions to support multiprecision computations. The ALU performs a standard set of arithmetic and logic operations; division primitives are also supported. The MAC performs single-cycle multiply, multiply/add, and multiply/subtract operations. The MAC has two 40-bit accumulators, which help with overflow. The shifter performs logical and arithmetic shifts, normalization, denormalization, and derive exponent operations. The shifter can be used to efficiently implement numeric format control, including multiword and block floating-point representations.

Register-usage rules influence placement of input and results within the computational units. For most operations, the computational units' data registers act as a data register file, permitting any input or result register to provide input to any unit for a computation. For feedback operations, the computational units let the output (result) of any unit be input to any unit on the next cycle. For conditional or multifunction instructions, there are restrictions on which data registers may provide inputs or receive results from each computational unit. For more information, see the *ADSP-219x DSP Instruction Set Reference*.

A powerful program sequencer controls the flow of instruction execution. The sequencer supports conditional jumps, subroutine calls, and low interrupt overhead. With internal loop counters and loop stacks, the ADSP-2195 executes looped code with zero overhead; no explicit jump instructions are required to maintain loops.

Two data address generators (DAGs) provide addresses for simultaneous dual operand fetches (from data memory and program memory). Each DAG maintains and updates four 16-bit address pointers. Whenever the pointer is used to access data (indirect addressing), it is pre- or post-modified by the value of one of four possible modify registers. A length value and base address may be associated with each pointer to implement automatic modulo addressing for circular buffers. Page registers in the DAGs allow circular addressing

External (Off-Chip) Memory

Each of the ADSP-2195's off-chip memory spaces has a separate control register, so applications can configure unique access parameters for each space. The access parameters include read and write wait counts, waitstate completion mode, I/O clock divide ratio, write hold time extension, strobe polarity, and data bus width. The core clock and peripheral clock ratios influence the external memory access strobe widths. For more information, see Clock Signals on page 14. The off-chip memory spaces are:

- External memory space ($\overline{MS3–0}$ pins)
- I/O memory space ($\overline{IOMS}$ pin)
- Boot memory space ($\overline{BMS}$ pin)

All of these off-chip memory spaces are accessible through the External Port, which can be configured for 8-bit or 16-bit data widths.

External Memory Space

External memory space consists of four memory banks. These banks can contain a configurable number of 64K word pages. At reset, the page boundaries for external memory have Bank0 containing pages 1–63, Bank1 containing pages 64–127, Bank2 containing pages 128–191, and Bank3 containing Pages 192–254. The $\overline{MS3–0}$ memory bank pins select Banks 3–0, respectively. The external memory interface decodes the 8 MSBs of the DSP program address to select one of the four banks. Both the ADSP-219x core and DMA-capable peripherals can access the DSP's external memory space.

I/O Memory Space

The ADSP-2195 supports an additional external memory called I/O memory space. This space is designed to support simple connections to peripherals (such as data converters and external registers) or to bus interface ASIC data registers. I/O space supports a total of 256K locations. The first 8K addresses are reserved for on-chip peripherals. The upper 248K addresses are available for external peripheral devices. The DSP's instruction set provides instructions for accessing I/O space. These instructions use an 18-bit address that is assembled from an 8-bit I/O page (IOPG) register and a 10-bit immediate value supplied in the instruction. Both the ADSP-219x core and a Host (through the Host Port Interface) can access I/O memory space.

Boot Memory Space

Boot memory space consists of one off-chip bank with 254 pages. The $\overline{BMS}$ memory bank pin selects boot memory space. Both the ADSP-219x core and DMA-capable peripherals can access the DSP's off-chip boot memory space. After reset, the DSP always starts executing instructions from the on-chip boot ROM. Depending on the boot configuration, the boot ROM code can start booting the DSP from boot memory. For more information, see Booting Modes on page 15.

Interrupts

The interrupt controller lets the DSP respond to 17 interrupts with minimum overhead. The controller implements an interrupt priority scheme as shown in Table 1. Applications can use the unassigned slots for software and peripheral interrupts.

Table 1. Interrupt Priorities/Addresses

Interrupt	IMASK/ IRPTL	Vector Address[1]
Emulator (NMI)— Highest Priority	NA	NA
Reset (NMI)	0	0x00 0000
Power-Down (NMI)	1	0x00 0020
Loop and PC Stack	2	0x00 0040
Emulation Kernel	3	0x00 0060
User Assigned Interrupt	4	0x00 0080
User Assigned Interrupt	5	0x00 00A0
User Assigned Interrupt	6	0x00 00C0
User Assigned Interrupt	7	0x00 00E0
User Assigned Interrupt	8	0x00 0100
User Assigned Interrupt	9	0x00 0120
User Assigned Interrupt	10	0x00 0140
User Assigned Interrupt	11	0x00 0160
User Assigned Interrupt	12	0x00 0180
User Assigned Interrupt	13	0x00 01A0
User Assigned Interrupt	14	0x00 01C0
User Assigned Interrupt— Lowest Priority	15	0x00 01E0

[1]These interrupt vectors start at address 0x10000 when the DSP is in "no-boot", run-form-external memory mode.

Table 2 shows the ID and priority at reset of each of the peripheral interrupts. To assign the peripheral interrupts a different priority, applications write the new priority to their corresponding control bits (determined by their ID) in the Interrupt Priority Control register. The peripheral interrupt's position in the IMASK and IRPTL register and its vector address depend on its priority level, as shown in Table 1. Because the IMASK and IRPTL registers are limited to 16 bits, any peripheral interrupts assigned a

priority level of 11 are aliased to the lowest priority bit position (15) in these registers and share vector address 0x00 01E0.

Table 2. Peripheral Interrupts and Priority at Reset

Interrupt	ID	Reset Priority
Slave DMA/Host Port Interface	0	0
SPORT0 Receive	1	1
SPORT0 Transmit	2	2
SPORT1 Receive	3	3
SPORT1 Transmit	4	4
SPORT2 Receive/SPI0	5	5
SPORT2 Transmit/SPI1	6	6
UART Receive	7	7
UART Transmit	8	8
Timer A	9	9
Timer B	10	10
Timer C	11	11
Programmable Flag 0 (any PFx)	12	11
Programmable Flag 1 (any PFx)	13	11
Memory DMA port	14	11

Interrupt routines can either be nested with higher priority interrupts taking precedence or processed sequentially. Interrupts can be masked or unmasked with the IMASK register. Individual interrupt requests are logically ANDed with the bits in IMASK; the highest priority unmasked interrupt is then selected. The emulation, power-down, and reset interrupts are nonmaskable with the IMASK register, but software can use the DIS INT instruction to mask the power-down interrupt.

The Interrupt Control (ICNTL) register controls interrupt nesting and enables or disables interrupts globally. The general-purpose Programmable Flag (PFx) pins can be configured as outputs, can implement software interrupts, and (as inputs) can implement hardware interrupts. Programmable Flag pin interrupts can be configured for level-sensitive, single edge-sensitive, or dual edge-sensitive operation.

Table 3. Interrupt Control (ICNTL) Register Bits

Bit	Description
0–3	Reserved
4	Interrupt Nesting Enable
5	Global Interrupt Enable
6	Reserved
7	MAC-Biased Rounding Enable
8–9	Reserved
10	PC Stack Interrupt Enable
11	Loop Stack Interrupt Enable
12–15	Reserved

The IRPTL register is used to force and clear interrupts. On-chip stacks preserve the processor status and are automatically maintained during interrupt handling. To support interrupt, loop, and subroutine nesting, the PC stack is 33 levels deep, the loop stack is eight levels deep, and the status stack is 16 levels deep. To prevent stack overflow, the PC stack can generate a stack-level interrupt if the PC stack falls below three locations full or rises above 28 locations full.

The following instructions globally enable or disable interrupt servicing, regardless of the state of IMASK.

```
ENA INT;
DIS INT;
```

At reset, interrupt servicing is disabled.

For quick servicing of interrupts, a secondary set of DAG and computational registers exist. Switching between the primary and secondary registers lets programs quickly service interrupts, while preserving the DSP's state.

DMA Controller

The ADSP-2195 has a DMA controller that supports automated data transfers with minimal overhead for the DSP core. Cycle stealing DMA transfers can occur between the ADSP-2195's internal memory and any of its DMA-capable peripherals. Additionally, DMA transfers can be accomplished between any of the DMA-capable peripherals and external devices connected to the external memory interface. DMA-capable peripherals include the Host port, SPORTs, SPI ports, and UART. Each individual DMA-capable peripheral has a dedicated DMA channel. To describe each DMA sequence, the DMA controller uses a

Bus Request and Bus Grant

The ADSP-2195 can relinquish control of the data and address buses to an external device. When the external device requires access to the bus, it asserts the bus request ($\overline{BR}$) signal. The ($\overline{BR}$) signal is arbitrated with core and peripheral requests. External Bus requests have the lowest priority. If no other internal request is pending, the external bus request will be granted. Due to synchronizer and arbitration delays, bus grants will be provided with a minimum of three peripheral clock delays. The ADSP-2195 will respond to the bus grant by:

- Three-stating the data and address buses and the $\overline{MS3–0}$, $\overline{BMS}$, $\overline{IOMS}$, $\overline{RD}$, and $\overline{WR}$ output drivers.
- Asserting the bus grant ($\overline{BG}$) signal.

The ADSP-2195 will halt program execution if the bus is granted to an external device and an instruction fetch or data read/write request is made to external general-purpose or peripheral memory spaces. If an instruction requires two external memory read accesses, the bus will not be granted between the two accesses. If an instruction requires an external memory read and an external memory write access, the bus may be granted between the two accesses. The external memory interface can be configured so that the core will have exclusive use of the interface. DMA and Bus Requests will be granted. When the external device releases $\overline{BR}$, the DSP releases $\overline{BG}$ and continues program execution from the point at which it stopped.

The bus request feature operates at all times, even while the DSP is booting and $\overline{RESET}$ is active.

The ADSP-2195 asserts the $\overline{BGH}$ pin when it is ready to start another external port access, but is held off because the bus was previously granted. This mechanism can be extended to define more complex arbitration protocols for implementing more elaborate multimaster systems.

Instruction Set Description

The ADSP-2195 assembly language instruction set has an algebraic syntax that was designed for ease of coding and readability. The assembly language, which takes full advantage of the processor's unique architecture, offers the following benefits:

- ADSP-219x assembly language syntax is a superset of and source-code-compatible (except for two data registers and DAG base address registers) with ADSP-218x family syntax. It may be necessary to restructure ADSP-218x programs to accommodate the ADSP-2195's unified memory space and to conform to its interrupt vector map.
- The algebraic syntax eliminates the need to remember cryptic assembler mnemonics. For example, a typical arithmetic add instruction, such as AR = AX0 + AY0, resembles a simple equation.
- Every instruction, but two, assembles into a single, 24-bit word that can execute in a single instruction cycle. The exceptions are two dual word instructions. One writes 16- or 24-bit immediate data to memory, and the other is an absolute jump/call with the 24-bit address specified in the instruction.
- Multifunction instructions allow parallel execution of an arithmetic, MAC, or shift instruction with up to two fetches or one write to processor memory space during a single instruction cycle.
- Program flow instructions support a wider variety of conditional and unconditional jumps/calls and a larger set of conditions on which to base execution of conditional instructions.

Development Tools

The ADSP-2195 is supported with a complete set of software and hardware development tools, including Analog Devices' emulators and VisualDSP++® development environment. The same emulator hardware that supports other ADSP-219x DSPs, also fully emulates the ADSP-2195.

The VisualDSP++ project management environment lets programmers develop and debug an application. This environment includes an easy-to-use assembler that is based on an algebraic syntax; an archiver (librarian/library builder), a linker, a loader, a cycle-accurate instruction-level simulator, a C/C++ compiler, and a C/C++ run-time library that includes DSP and mathematical functions. Two key points for these tools are:

- Compiled ADSP-219x C/C++ code efficiency—the compiler has been developed for efficient translation of C/C++ code to ADSP-219x assembly. The DSP has architectural features that improve the efficiency of compiled C/C++ code.
- ADSP-218x family code compatibility—The assembler has legacy features to ease the conversion of existing ADSP-218x applications to the ADSP-219x.

Debugging both C/C++ and assembly programs with the VisualDSP++ debugger, programmers can:

- View mixed C/C++ and assembly code (interleaved source and object information)
- Insert break points
- Set conditional breakpoints on registers, memory, and stacks
- Trace instruction execution
- Perform linear or statistical profiling of program execution
- Fill, dump, and graphically plot the contents of memory
- Source level debugging
- Create custom debugger windows

The VisualDSP++ IDE lets programmers define and manage DSP software development. Its dialog boxes and property pages let programmers configure and manage all

of the ADSP-219x development tools, including the syntax highlighting in the VisualDSP++ editor. This capability permits:

- Control how the development tools process inputs and generate outputs.
- Maintain a one-to-one correspondence with the tool's command line switches.

Analog Devices' DSP emulators use the IEEE 1149.1 JTAG test access port of the ADSP-2195 processor to monitor and control the target board processor during emulation. The emulator provides full-speed emulation, allowing inspection and modification of memory, registers, and processor stacks. Nonintrusive in-circuit emulation is assured by the use of the processor's JTAG interface—the emulator does not affect target system loading or timing.

In addition to the software and hardware development tools available from Analog Devices, third parties provide a wide range of tools supporting the ADSP-219x processor family. Hardware tools include ADSP-219x PC plug-in cards. Third Party software tools include DSP libraries, real-time operating systems, and block diagram design tools.

Designing an Emulator-Compatible DSP Board (Target)

The White Mountain DSP (Product Line of Analog Devices, Inc.) family of emulators are tools that every DSP developer needs to test and debug hardware and software systems. Analog Devices has supplied an IEEE 1149.1 JTAG Test Access Port (TAP) on each JTAG DSP. The emulator uses the TAP to access the internal features of the DSP, allowing the developer to load code, set breakpoints, observe variables, observe memory, and examine registers. The DSP must be halted to send data and commands, but once an operation has been completed by the emulator, the DSP system is set running at full speed with no impact on system timing.

To use these emulators, the target's design must include the interface between an Analog Devices' JTAG DSP and the emulation header on a custom DSP target board.

Target Board Header

The emulator interface to an Analog Devices' JTAG DSP is a 14-pin header, as shown in Figure 7. The customer must supply this header on the target board in order to communicate with the emulator. The interface consists of a standard dual row 0.025" square post header, set on 0.1" × 0.1" spacing, with a minimum post length of 0.235". Pin 3 is the key position used to prevent the pod from being inserted backwards. This pin must be clipped on the target board.

Also, the clearance (length, width, and height) around the header must be considered. Leave a clearance of at least 0.15" and 0.10" around the length and width of the header, and reserve a height clearance to attach and detach the pod connector.

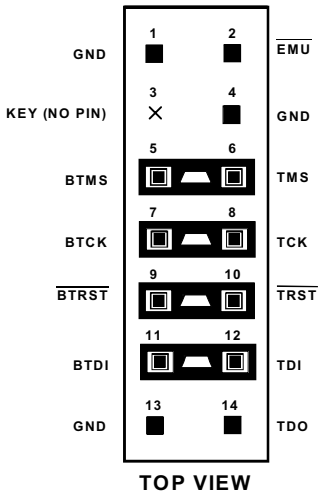

Figure 7. JTAG Target Board Connector for JTAG Equipped Analog Devices DSP (Jumpers in Place)

As can be seen in Figure 7, there are two sets of signals on the header. There are the standard JTAG signals TMS, TCK, TDI, TDO, $\overline{\text{TRST}}$, and $\overline{\text{EMU}}$ used for emulation purposes (via an emulator). There are also secondary JTAG signals BTMS, BTCK, BTDI, and $\overline{\text{BTRST}}$ that are optionally used for board-level (boundary scan) testing.

When the emulator is not connected to this header, place jumpers across BTMS, BTCK, $\overline{\text{BTRST}}$, and BTDI as shown in Figure 8. This holds the JTAG signals in the correct state to allow the DSP to run free. Remove all the jumpers when connecting the emulator to the JTAG header.

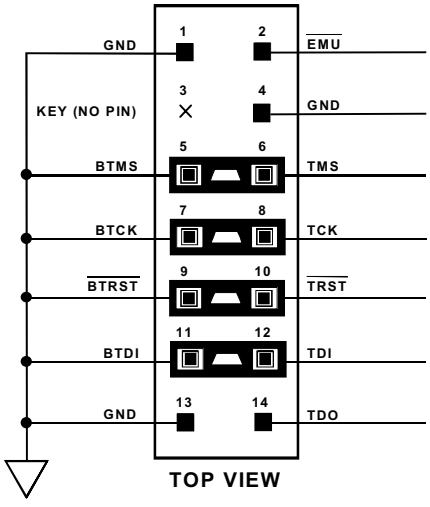

Figure 8. JTAG Target Board Connector with No Local Boundary Scan

Table 7. Pin Descriptions (Continued)

Pin	Type	Function
D11 /PF11 /SPI1SEL5	I/O/T I/O I	Data 11 (if 16-bit external bus)/Programmable Flags 11 (if 8-bit external bus)/SPI1 Slave Select output 5 (if 8-bit external bus, when SPI1 enabled)
D10 /PF10 /SPI0SEL5	I/O/T I/O I	Data 10 (if 16-bit external bus)/Programmable Flags 10 (if 8-bit external bus)/SPI0 Slave Select output 5 (if 8-bit external bus, when SPI0 enabled)
D9 /PF9 /SPI1SEL4	I/O/T I/O I	Data 9 (if 16-bit external bus)/Programmable Flags 9 (if 8-bit external bus)/SPI1 Slave Select output 4 (if 8-bit external bus, when SPI1 enabled)
D8 /PF8 /SPI0SEL4	I/O/T I/O I	Data 8 (if 16-bit external bus)/Programmable Flags 8 (if 8-bit external bus)/SPI0 Slave Select output 4 (if 8-bit external bus, when SPI0 enabled)
PF7 /SPI1SEL3 /DF	I/O/T I I	Programmable Flags 7/SPI1 Slave Select output 3 (when SPI0 enabled)/Divisor Frequency (divisor select for PLL input during boot)
PF6 /SPI0SEL3 /MSEL6	I/O/T I I	Programmable Flags 6/SPI0 Slave Select output 3 (when SPI0 enabled)/Multiplier Select 6 (during boot)
PF5 /SPI1SEL2 /MSEL5	I/O/T I I	Programmable Flags 5/SPI1 Slave Select output 2 (when SPI0 enabled)/Multiplier Select 5 (during boot)
PF4 /SPI0SEL2 /MSEL4	I/O/T I I	Programmable Flags 4/SPI0 Slave Select output 2 (when SPI0 enabled)/Multiplier Select 4 (during boot)
PF3 /SPI1SEL1 /MSEL3	I/O/T I I	Programmable Flags 3/SPI1 Slave Select output 1 (when SPI0 enabled)/Multiplier Select 3 (during boot)
PF2 /SPI0SEL1 /MSEL2	I/O/T I I	Programmable Flags 2/SPI0 Slave Select output 1 (when SPI0 enabled)/Multiplier Select 2 (during boot)
PF1 /SPISS1 /MSEL1	I/O/T I I	Programmable Flags 1/SPI1 Slave Select input (when SPI1 enabled)/Multiplier Select 1 (during boot)
PF0 /SPISS0 /MSEL0	I/O/T I I	Programmable Flags 0/SPI0 Slave Select input (when SPI0 enabled)/Multiplier Select 0 (during boot)
$\overline{RD}$	O/T	External Port Read Strobe
$\overline{WR}$	O/T	External Port Write Strobe
ACK	I	External Port Access Ready Acknowledge
$\overline{BMS}$	O/T	External Port Boot Space Select
$\overline{IOMS}$	O/T	External Port IO Space Select

ELECTRICAL CHARACTERISTICS (CONTINUED)

Parameter[1]	Description	Test Conditions	Min	Typical	Max	Unit
$I_{DD-IDLE2}$	Supply Current (Core) Idle2	PLL Enabled, HCLK = 80 MHz, CCLK Disabled[7]		1		mA
$I_{DD-TYPICAL}$	Supply Current (Core) Typical	HCLK = 80 MHz, CCLK = 160 MHz[7,8]		184		mA
$I_{DD-PEAK}$	Supply Current (Core) Peak	HCLK = 80 MHz, CCLK = 160 MHz[7,8]		215		mA
$I_{DD-PERIPHERAL1}$	Supply Current (Peripheral)	PLL Enabled, Core, HCLK, CLKIN Disabled[7]		5		mA
$I_{DD-PERIPHERAL2}$	Supply Current (Peripheral)	HCLK = 80 MHz[7]		60		mA
$I_{DD-POWERDOWN}$	Supply Current	PLL, Core, HCLK, CLKIN Disabled[7]		100		µA
C_{IN}	Input Capacitance[9, 10]	f_{IN} = 1 MHz, T_{CASE} = 25°C, V_{IN} = 2.5 V			TBD	pF

[1]Specifications subject to change without notice.

[2]Applies to output and bidirectional pins: DATA15–0, ADDR21–0, HAD15–0, $\overline{MS3–0}$, $\overline{IOMS}$, $\overline{RD}$, $\overline{WR}$, CLKOUT, HACK, PF7–0, TMR2–0, $\overline{BGH}$, $\overline{BG}$, DT0, DT1, DT2/MISO0, TCLK0, TCLK1, TCLK2/SCK0, RCLK0, RCLK1, RCLK2/SCK1, TFS0, TFS1, TFS2/MOSI0, RFS0, RFS1, RFS2/MOSI1, $\overline{BMS}$, TDO, TXD, $\overline{EMU}$.

[3]Applies to input pins: ACK, $\overline{BR}$, $\overline{HCMS}$, $\overline{HCIOMS}$, OPMODE, BMODE1–0, HA16, HALE, $\overline{HRD}$, $\overline{HWR}$, CLKIN, $\overline{RESET}$, TCK, TDI, TMS, $\overline{TRST}$, DR0, DR1, BYPASS, RXD.

[4]Applies to input pins with internal pull-ups: BMODE0, BMODE1, OPMODE, BYPASS, TCK, TMS, TDI, $\overline{RESET}$.

[5]Applies to input pin with internal pull-down: $\overline{TRST}$

[6]Applies to three-statable pins: DATA15–0, ADDR21–0, $\overline{MS3–0}$, $\overline{RD}$, $\overline{WR}$, PF7–0, $\overline{BMS}$, $\overline{IOMS}$, TFSx, RFSx, TDO, $\overline{EMU}$.

[7]Test Conditions: @ V_{DDINT} = 2.5V, T_{AMB} = 25°C

[8]Refer to Table 23 on page 52 for definitions of operation types.

[9]Applies to all signal pins.

[10]Guaranteed, but not tested.

ABSOLUTE MAXIMUM RATINGS

V_{DDINT} Internal (Core) Supply Voltage[1,2] .....–0.3 to 3.0 V

V_{DDEXT} External (I/O) Supply Voltage .....–0.3 to 4.6 V

V_{IL}–V_{IH} Input Voltage .....–0.5 to V_{DDEXT}+0.5 V

V_{OL}–V_{OH} Output Voltage Swing .....–0.5 to V_{DDEXT}+0.5 V

C_L Load Capacitance..... 200 pF

t_{CCLK} Core Clock Period..... 6.25 ns

f_{CCLK} Core Clock Frequency..... 160 MHz

t_{HCLK} Peripheral Clock Period..... 10 ns

f_{HCLK} Peripheral Clock Frequency..... 100 MHz

T_{STORE} Storage Temperature Range .....–65 to 150°C

T_{LEAD} Lead Temperature (5 seconds)..... 185°C

[1]Specifications subject to change without notice.

[2]Stresses greater than those listed above may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions greater than those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD SENSITIVITY

CAUTION:

ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000V readily accumulate on the human body and test equipment and can discharge without detection. Although the ADSP-2195 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high-energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

TIMING SPECIFICATIONS

This section contains timing information for the DSP's external signals.

Timer PWM_OUT Cycle Timing

Table 10 and Figure 13 describe timer expired operations. The input signal is asynchronous in "width capture mode" and has an absolute maximum input frequency of 50 MHz.

Table 10. Timer PWM_OUT Cycle Timing

Parameter	Description	Min	Max	Unit
Switching Characteristic				
t_{HTO}	Timer pulsewidth output[1]	6.25	(2^{32}–1) cycles	ns

[1]The minimum time for t_{HTO} is one cycle, and the maximum time for t_{HTO} equals (2^{32}–1) cycles.

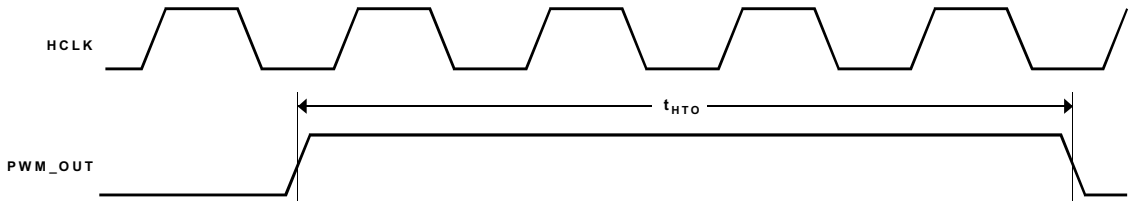

Figure 13. Timer PWM_OUT Cycle Timing

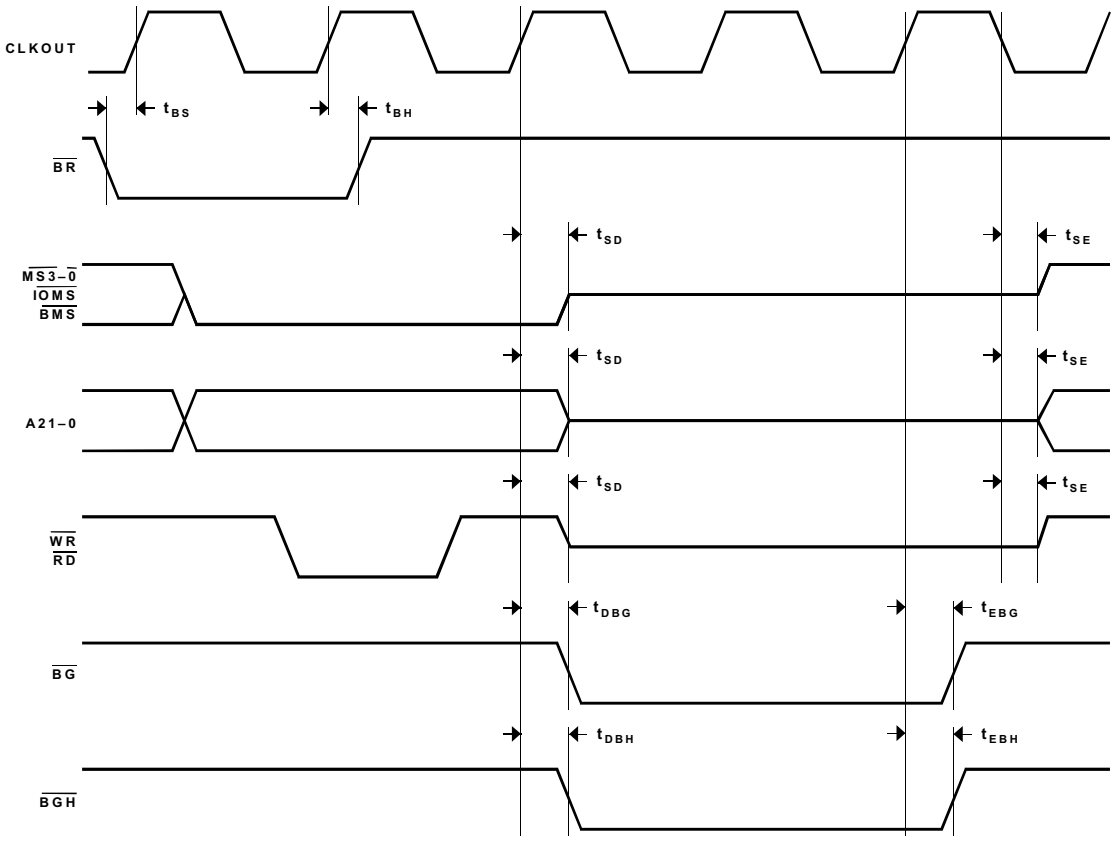

Figure 16. External Port Bus Request and Grant Cycle Timing

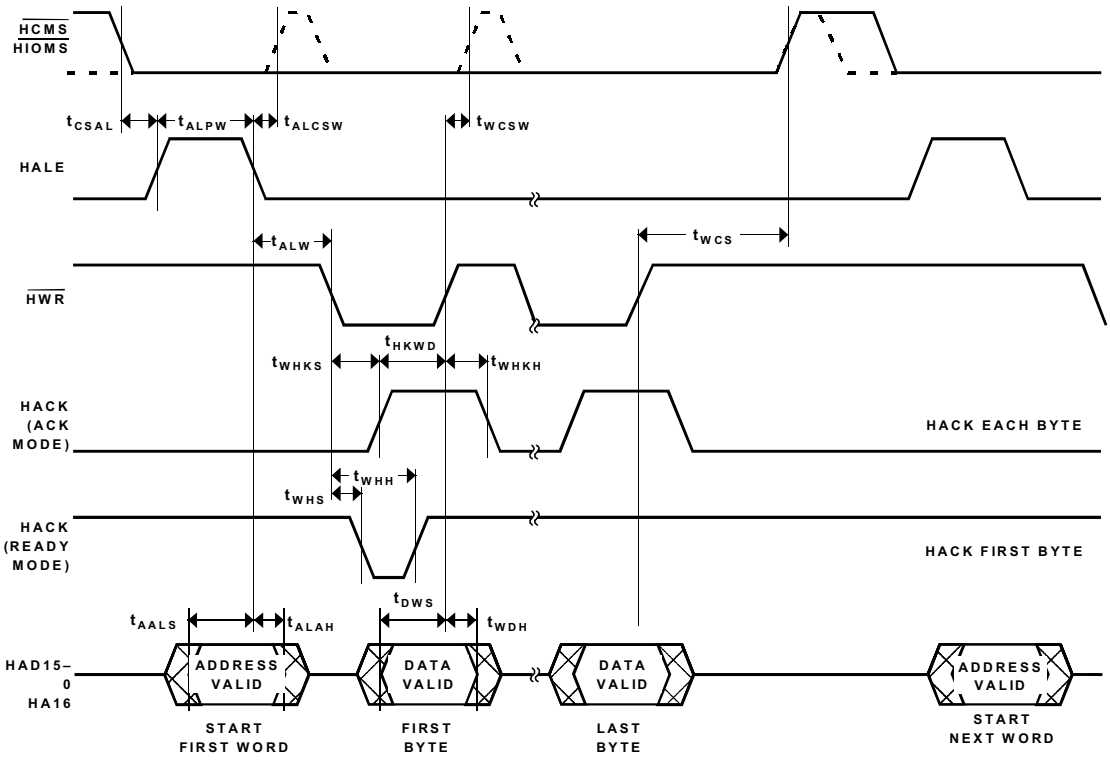

Figure 17. Host Port ALE Mode Write Cycle Timing

Host Port ALE Mode Read Cycle Timing

Table 16 and Figure 19 describe host port read operations in Address Latch Enable (ALE) mode. For more information on ACK, Ready, ALE, and ACC mode selection, see the Host port modes description on page 10.

Table 16. Host Port ALE Mode Read Cycle Timing

Parameter	Description	Min	Max	Unit
Switching Characteristics				
t_{RHKS}	$\overline{HRD}$ asserted to HACK asserted (setup, ACK Mode)	2	$2+t_{NH}$[1]	ns
t_{RHKH}	$\overline{HRD}$ de-asserted to HACK de-asserted (hold, ACK Mode)		2	ns
t_{RHS}	$\overline{HRD}$ asserted to HACK asserted (setup, Ready Mode)		2	ns
t_{RHH}	$\overline{HRD}$ asserted to HACK de-asserted (hold, Ready Mode)		$2+t_{NH}$[1]	ns
Timing Requirements				
t_{CSAL}	$\overline{HCMS}$ or $\overline{HCIOMS}$ asserted to HALE asserted (delay)	0		ns
t_{ALCS}	HALE de-asserted to optional $\overline{HCMS}$ or $\overline{HCIOMS}$ de-asserted	1		ns
t_{RCSW}	$\overline{HRD}$ de-asserted to $\overline{HCMS}$ or $\overline{HCIOMS}$ de-asserted	1		ns
t_{ALR}	HALE de-asserted to $\overline{HRD}$ asserted	1		ns
t_{RCS}	$\overline{HRD}$ de-asserted (after last byte) to $\overline{HCMS}$ or $\overline{HCIOMS}$ de-asserted (ready for next read)	1		ns
t_{ALPW}	HALE asserted pulsewidth	4		ns
t_{HKRD}	HACK asserted to $\overline{HRD}$ de-asserted (hold, ACK Mode)	1.5		ns
t_{AALS}	Address valid to HALE de-asserted (setup)	4		ns
t_{ALAH}	HALE de-asserted to address invalid (hold)	1		ns
t_{RDH}	$\overline{HRD}$ de-asserted to data invalid (hold)	1		ns

[1] t_{NH} are peripheral bus latencies ($n \times t_{HCLK}$); these are internal DSP latencies related to the number of peripherals attempting to access DSP memory at the same time.

Serial Port (SPORT) Frame Synch Timing

Table 19 and Figure 22 describe SPORT frame synch operations.

To determine whether communication is possible between two devices at clock speed n, the following specifications must be confirmed: 1) frame sync delay and frame sync setup and hold, 2) data delay and data setup and hold, and 3) R/TCLK width.

Table 19. Serial Port (SPORT) Frame Synch Timing

Parameter	Description	Min	Max	Unit
Switching Characteristics				
t_{HOFSE}	RFS Hold after RCLK (Internally Generated RFS)[1]		12.4	ns
t_{HOFSI}	TFS Hold after TCLK (Internally Generated TFS)[1]		12.2	ns
$t_{DDTENFS}$	Data Enable from late FS or MCE = 1, MFD = 0[2]		4.7	ns
$t_{DDTLFSE}$	Data Delay from Late External TFS or External RFS with MCE = 1, MFD = 0[3]		4.7	ns
t_{HDTE}	Transmit Data Hold after TCLK (external clk)[1]		12.4	ns
t_{HDTI}	Transmit Data Hold after TCLK (internal clk)[1]	0	12.2	ns
t_{DDTE}	Transmit Data Delay after TCLK (external clk)[1]	0	12.2	ns
t_{DDTI}	Transmit Data Delay after TCLK (internal clk)[1]	0	11.1	ns
Timing Requirements				
t_{SFSE}	TFS/RFS Setup before TCLK/RCLK (external clk)[3]	–0.6	TBD	ns
t_{SFSI}	TFS/RFS Setup before TCLK/RCLK (internal clk)[3]	–0.6	TBD	ns

[1]Referenced to drive edge.

[2]MCE = 1, TFS enable and TFS valid follow $t_{DDTLFSE}$ and $t_{DDTENFS}$.

[3]Referenced to sample edge.

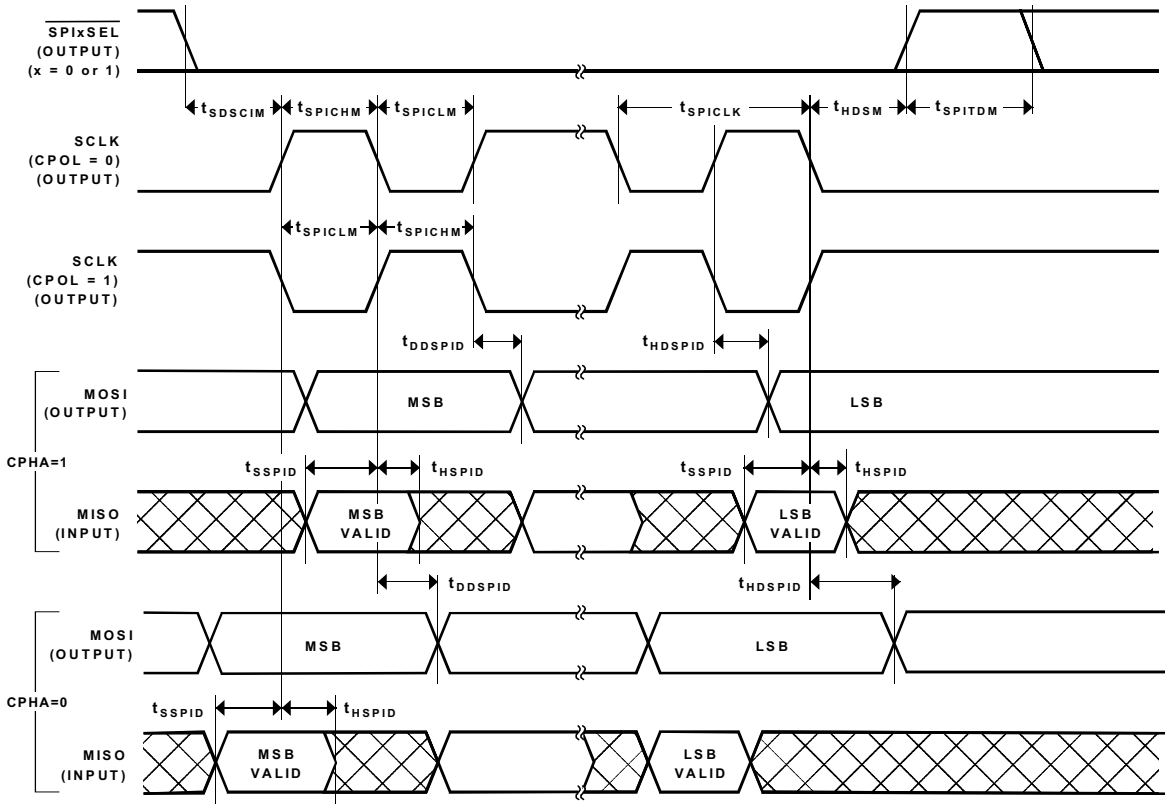

Figure 23. Serial Peripheral Interface (SPI) Port—Master

Universal Asynchronous Receiver-Transmitter (UART) Port—Receive and Transmit Timing

Figure 25 describes UART port receive and transmit operations. The maximum baud rate is HCLK/16. As shown in Figure 25 there is some latency between the generation internal UART interrupts and the external data operations. These latencies are negligible at the data transmission rates for the UART.

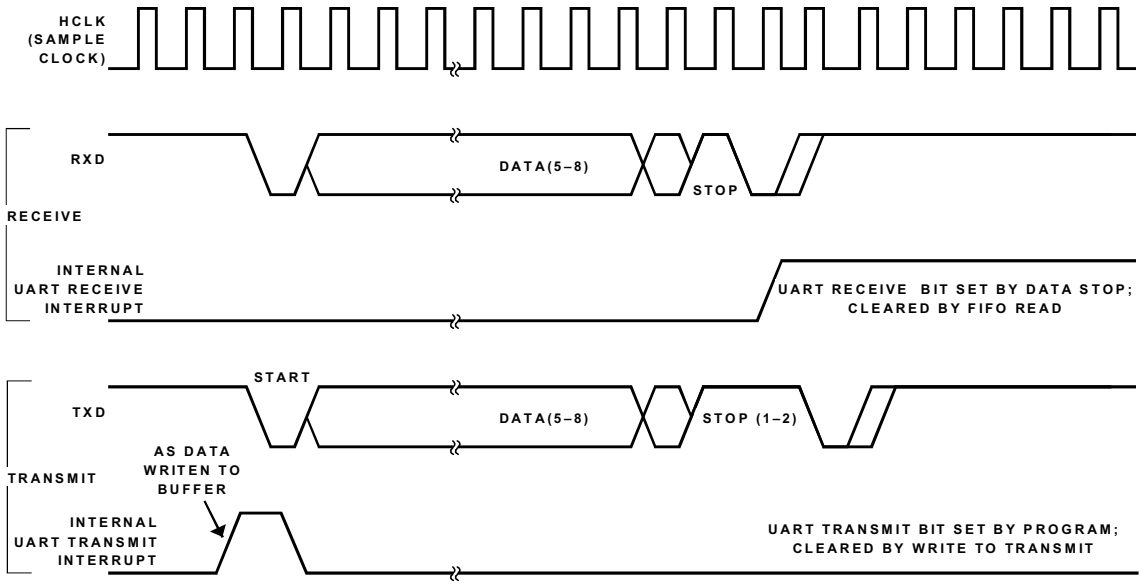

Figure 25. UART Port—Receive and Transmit Timing

JTAG Test And Emulation Port Timing

Table 22 and Figure 26 describe JTAG port operations.

Table 22. JTAG Port Timing

Parameter	Description	Min	Max	Unit
Switching Characteristics				
t_{DTDO}	TDO Delay from TCK Low		4	ns
t_{DSYS}	System Outputs Delay After TCK Low[1]	0	5	ns
Timing Parameters				
t_{TCK}	TCK Period	20		ns
t_{STAP}	TDI, TMS Setup Before TCK High		4	ns
t_{HTAP}	TDI, TMS Hold After TCK High		4	ns
t_{SSYS}	System Inputs Setup Before TCK Low[2]		4	ns
t_{HSYS}	System Inputs Hold After TCK Low[2]		5	ns
t_{TRSTW}	$\overline{TRST}$ Pulsewidth[3]	4		ns

[1]System Outputs = DATA15–0, ADDR21–0, $\overline{MS3–0}$, $\overline{RD}$, $\overline{WR}$, ACK, CLKOUT, $\overline{BG}$, PF7–0, TIMEXP, DT0, DT1, TCLK0, TCLK1, RCLK0, RCLK1, TFS0, TFS1, RFS0, RFS1, $\overline{BMS}$.

[2]System Inputs = DATA15–0, ADDR21–0, $\overline{RD}$, $\overline{WR}$, ACK, $\overline{BR}$, $\overline{BG}$, PF7–0, DR0, DR1, TCLK0, TCLK1, RCLK0, RCLK1, TFS0, TFS1, RFS0, RFS1, CLKIN, $\overline{RESET}$.

[3]50 MHz max.

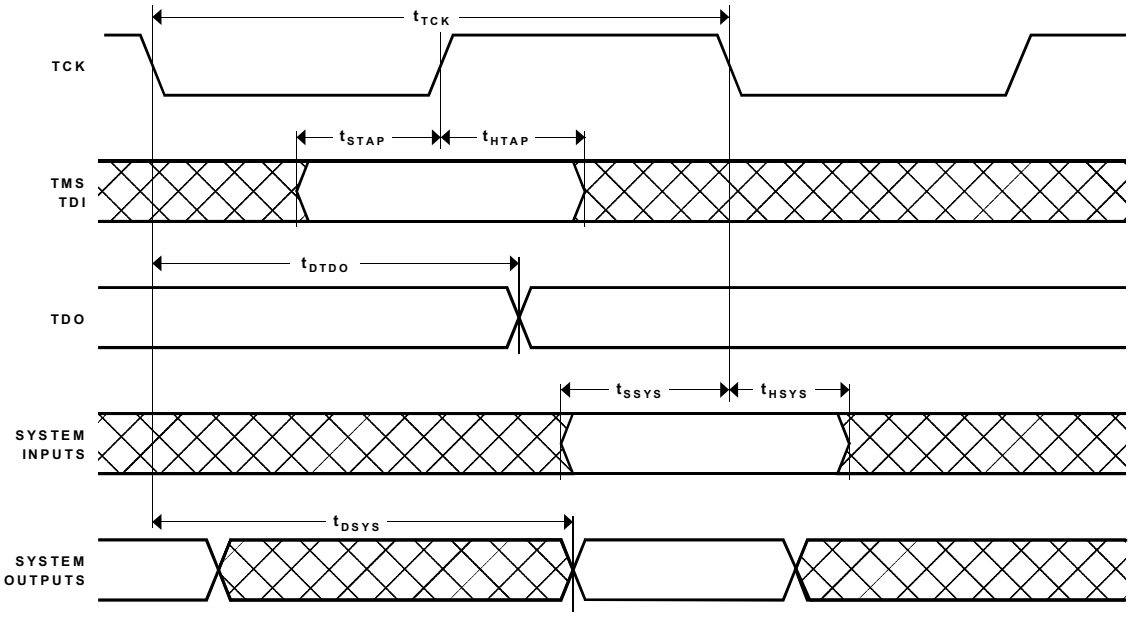

Figure 26. JTAG Port Timing

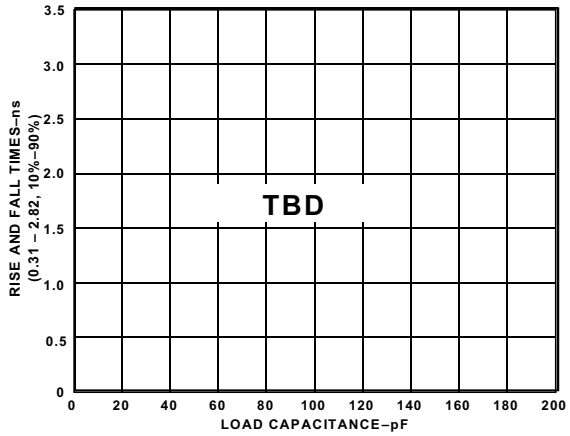

Figure 36. Typical Output Rise Time (10%-90%, V_{DDEXT}=Min) vs. Load Capacitance

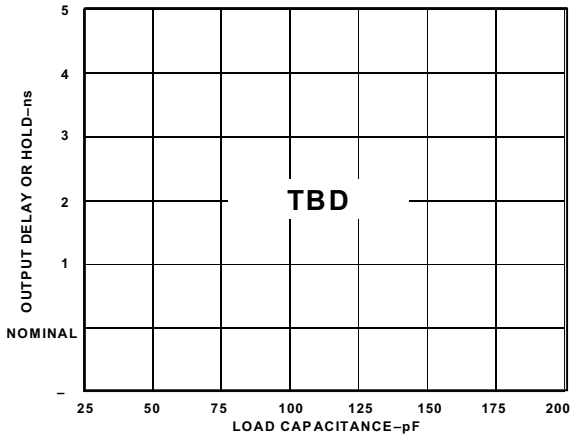

Figure 37. Typical Output Delay or Hold vs. Load Capacitance (at Max Case Temperature)

$$T_{AMB} = T_{CASE} - PD \times \theta_{CA}$$

Figure 38. T_{CASE} Calculation

Table 25. θ_{CA} Values[1]

Airflow (Linear Ft./Min.)	0	100	200	400	600
Airflow (Meters/Second)	0	0.5	1	2	3
LQFP: θ_{CA} (°C/W)	44.3	41.4	38.5	35.3	32.1
Mini-BGA: θ_{CA} (°C/W)	26	24	22	20.9	19.8

Table 27. 144-Lead LQFP Pins (Numerically By Pin Number (Continued)

SIGNAL	PIN #
A5	89
V_{DDEXT}	90
A6	91
A7	92
A8	93
GND	94
A9	95
A10	96
A11	97
A12	98
A13	99
V_{DDEXT}	100
A14	101
A15	102
A16	103
A17	104
GND	105
A18	106
A19	107
A20	108
A21	109
$\overline{BGH}$	110
$\overline{BG}$	111
$\overline{BR}$	112
$\overline{BMS}$	113
$\overline{IOMS}$	114
$\overline{MS0}$	115
$\overline{MS1}$	116
$\overline{MS2}$	117
V_{DDEXT}	118

ORDERING GUIDE

Part Number[1, 2]	Ambient Temperature Range	Instruction Rate	On-Chip SRAM	Operating Voltage
ADSP-2195MKST-160X	0°C to 70°C	160 MHz	1.3M bit	2.5 Int./3.3 Ext. V
ADSP-2195MBST-140X	-40°C to 85°C	140 MHz	1.3M bit	2.5 Int./3.3 Ext. V
ADSP-2195MKCA-160X	0°C to 70°C	160 MHz	1.3M bit	2.5 Int./3.3 Ext. V
ADSP-2195MBCA-140X	-40°C to 85°C	140 MHz	1.3M bit	2.5 Int./3.3 Ext. V

[1]ST = Plastic Thin Quad Flatpack (LQFP).

[2]CA = Mini Ball Grid Array