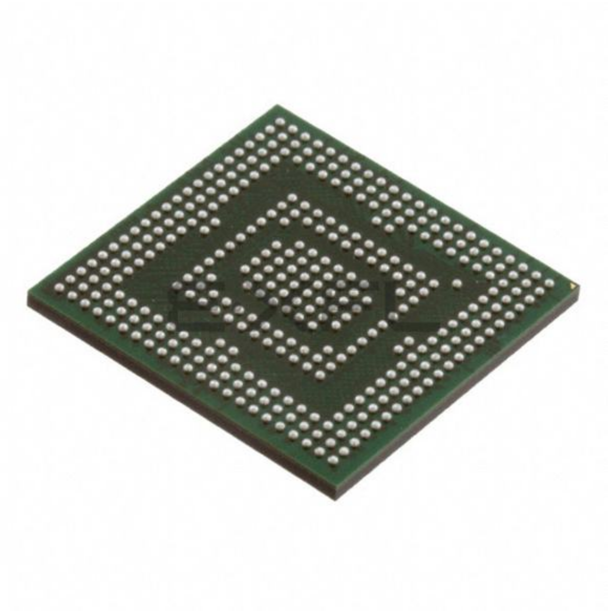
Analog Devices Inc. - ADSP-21584CBCZ-4A Datasheet



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	Active
Type	Floating Point
Interface	CAN, EBI/EMI, Ethernet, DAI, I²C, MMC/SD/SDIO, SPI, SPORT, UART/USART, USB OTG
Clock Rate	450MHz
Non-Volatile Memory	ROM (512kB)
On-Chip RAM	640kB
Voltage - I/O	3.30V
Voltage - Core	1.10V
Operating Temperature	-40°C ~ 95°C (TA)
Mounting Type	Surface Mount
Package / Case	349-LFBGA, CSPBGA
Supplier Device Package	349-CSPBGA (19x19)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/adsp-21584cbcz-4a

The memory map in Table 4 gives the L1 memory address space and shows multiple L1 memory blocks offering a configurable mix of SRAM and cache.

L1 Master and Slave Ports

Each SHARC+ core has two master and two slave ports to and from the system fabric. One master port fetches instructions. The second master port drives data to the system world. Both slave ports allow conflict free core/direct memory access (DMA) streams to the individual memory blocks. For slave port addresses, refer to the L1 memory address map in Table 4.

L1 On-Chip Memory Bandwidth

The internal memory architecture allows programs to have four accesses at the same time to any of the four blocks, assuming no block conflicts. The total bandwidth is realized using both the DMD and PMD buses.

Instruction and Data Cache

The ADSP-SC58x/ADSP-2158x processors also include a traditional instruction cache (I-cache) and two data caches (D-cache) (PM and DM caches). These caches support one instruction access and two data accesses over the DM and PM buses, per CCLK cycle. The cache controllers automatically manage the configured L1 memory. The system can configure part of the L1 memory for automatic management by the cache controllers. The sizes of these caches are independently configurable from 0 kB to a maximum of 128 kB each. The memory not managed by the cache controllers is directly addressable by the processors. The controllers ensure the data coherence between the two data caches. The caches provide user-controllable features such as full and partial locking, range-bound invalidation, and flushing.

System Event Controller (SEC) Input

The output of the system event controller (SEC) controller is forwarded to the core event controller (CEC) to respond directly to all unmasked system-based interrupts. The SEC also supports nesting including various SEC interrupt channel arbitration options. For all SEC channels, the processor automatically stacks the arithmetic status (ASTATx and ASTATy) registers and mode (MODE1) register in parallel with the interrupt servicing.

Core Memory-Mapped Registers (CMMR)

The core memory-mapped registers control the L1 instruction and data cache, BTB, L2 cache, parity error, system control, debug, and monitor functions.

SHARC+ CORE ARCHITECTURE

The ADSP-SC58x/ADSP-2158x processors are code compatible at the assembly level with the ADSP-2148x, ADSP-2147x, ADSP-2146x, ADSP-2137x, ADSP-2136x, ADSP-2126x, ADSP-2116x, and with the first-generation ADSP-2106x SHARC processors.

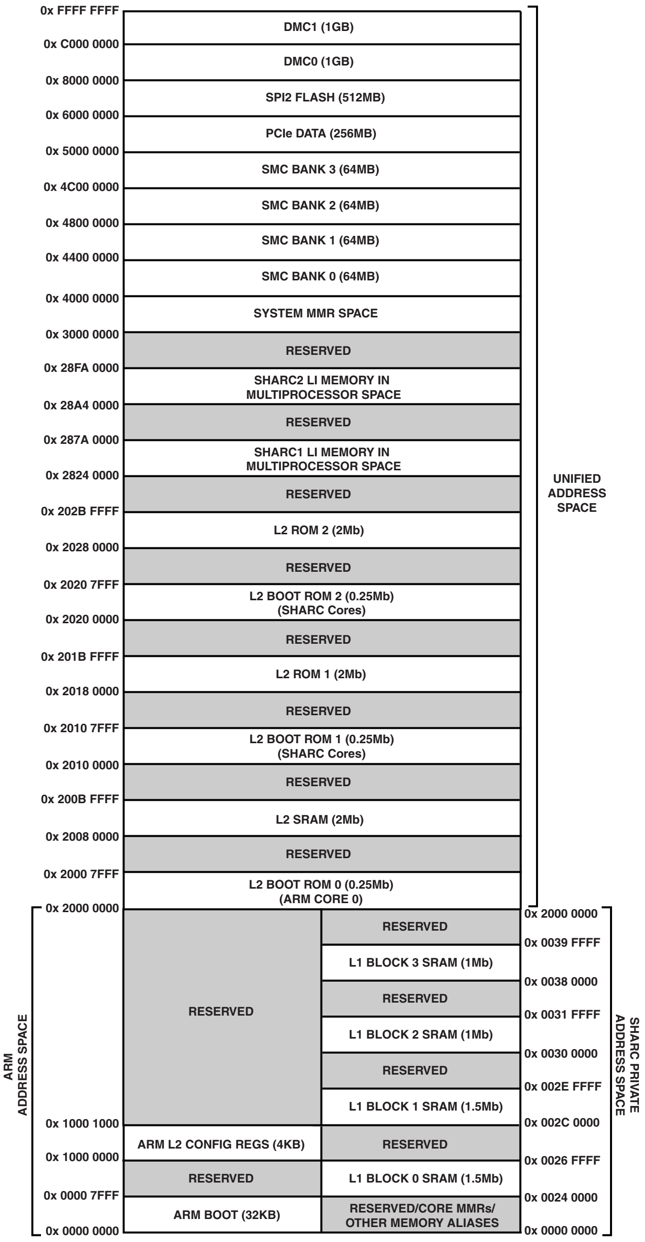


Figure 5. ADSP-SC58x/ADSP-2158x Memory Map

The ADSP-SC58x/ADSP-2158x processors share architectural features with the ADSP-2126x, ADSP-2136x, ADSP-2137x, ADSP-214xx, and ADSP-2116x SIMD SHARC processors, shown in Figure 4 and detailed in the following sections.

SIMD Computational Engine

The SHARC+ core contains two computational processing elements that operate as a single-instruction, multiple data (SIMD) engine.

The processing elements are referred to as PEx and PEy data registers and each contain an arithmetic logic unit (ALU), multiplier, shifter, and register file. PEx is always active and PEy is enabled by setting the PEYEN mode bit in the mode control register (MODE1).

Single instruction multiple data (SIMD) mode allows the processors to execute the same instruction in both processing elements, but each processing element operates on different data. This architecture efficiently executes math intensive DSP algorithms. In addition to all the features of previous generation SHARC cores, the SHARC+ core also provides a new and simpler way to execute an instruction only on the PEy data register.

SIMD mode also affects the way data transfers between memory and processing elements because to sustain computational operation in the processing elements requires twice the data bandwidth. Therefore, entering SIMD mode doubles the bandwidth between memory and the processing elements. When using the DAGs to transfer data in SIMD mode, two data values transfer with each memory or register file access.

Independent, Parallel Computation Units

Within each processing element is a set of pipelined computational units. The computational units consist of a multiplier, arithmetic/logic unit (ALU), and shifter. These units are arranged in parallel, maximizing computational throughput. These computational units support IEEE 32-bit single-precision floating-point, 40-bit extended-precision floating-point, IEEE 64-bit double-precision floating-point, and 32-bit fixed-point data formats.

A multifunction instruction set supports parallel execution of ALU and multiplier operations. In SIMD mode, the parallel ALU and multiplier operations occur in both processing elements per core.

All processing operations take one cycle to complete. For all floating-point operations, the processor takes two cycles to complete in case of data dependency. Double-precision floating-point data take two to six cycles to complete. The processor stalls for the appropriate number of cycles for an interlocked pipeline plus data dependency check.

Core Timer

Each SHARC+ processor core also has a timer. This extra timer is clocked by the internal processor clock and is typically used as a system tick clock for generating periodic operating system interrupts.

Data Register File

Each processing element contains a general-purpose data register file. The register files transfer data between the computation units and the data buses, and store intermediate results. These 10-port, 32-register register files (16 primary, 16 secondary), combined with the enhanced Harvard architecture of the processor, allow unconstrained data flow between computation units and internal memory. The registers in the PEx data register file are referred to as R0–R15 and in the PEy data register file as S0–S15.

Context Switch

Many of the registers of the processor have secondary registers that can activate during interrupt servicing for a fast context switch. The data, DAG, and multiplier result registers have secondary registers. The primary registers are active at reset, while control bits in MODE1 activate the secondary registers.

Universal Registers (USTAT)

General-purpose tasks use the universal registers. The four USTAT registers allow easy bit manipulations (set, clear, toggle, test, XOR) for all control and status peripheral registers.

The data bus exchange register (PX) permits data to pass between the 64-bit PM data bus and the 64-bit DM data bus or between the 40-bit register file and the PM or DM data bus. These registers contain hardware to handle the data width difference.

Data Address Generators With Zero-Overhead Hardware Circular Buffer Support

For indirect addressing and implementing circular data buffers in hardware, the ADSP-SC58x/ADSP-2158x processor uses the two data address generators (DAGs). Circular buffers allow efficient programming of delay lines and other data structures required in digital signal processing, and are commonly used in digital filters and Fourier transforms. The two DAGs of the processors contain sufficient registers to allow the creation of up to 32 circular buffers (16 primary register sets and 16 secondary sets). The DAGs automatically handle address pointer wraparound, reduce overhead, increase performance, and simplify implementation. Circular buffers can start and end at any memory location.

Flexible Instruction Set Architecture (ISA)

The ISA, a 48-bit instruction word, accommodates various parallel operations for concise programming. For example, the processors can conditionally execute a multiply, an add, and a subtract in both processing elements while branching and fetching up to four 32-bit values from memory—all in a single instruction. Additionally, the double-precision floating-point instruction set is an addition to the SHARC+ core.

Variable Instruction Set Architecture (VISA)

In addition to supporting the standard 48-bit instructions from previous SHARC processors, the SHARC+ core processors support 16-bit and 32-bit opcodes for many instructions, formerly 48-bit in the ISA. This feature, called variable instruction set architecture (VISA), drops redundant or unused bits within the 48-bit instruction to create more efficient and compact code. The program sequencer supports fetching these 16-bit and 32-bit instructions from both internal and external memories. VISA is not an operating mode; it is only address dependent (refer to memory map ISA/VISA address spaces in Table 7). Furthermore, it allows jumps between ISA and VISA instruction fetches.

Memory Direct Memory Access (MDMA)

The processor supports various MDMA operations, including,

- Standard bandwidth MDMA channels with CRC protection (32-bit bus width, runs on SCLK0)
- Enhanced bandwidth MDMA channel (32-bit bus width, runs on SYSCLK)
- Maximum bandwidth MDMA channels (64-bit bus width, run on SYCLK, one channel can be assigned to the FFT accelerator)

Extended Memory DMA

Extended memory DMA supports various operating modes such as delay line (which allows processor reads and writes to external delay line buffers and to the external memory) with limited core interaction and scatter/gather DMA (writes to and from noncontiguous memory blocks).

Cyclic Redundant C ode (CRC) Protection

The cyclic redundant codes (CRC) protection modules allow system software to calculate the signature of code, data, or both in memory, the content of memory-mapped registers, or periodic communication message objects. Dedicated hardware circuitry compares the signature with precalculated values and triggers appropriate fault events.

For example, every 100 ms the system software initiates the signature calculation of the entire memory contents and compares these contents with expected, precalculated values. If a mismatch occurs, a fault condition is generated through the processor core or the trigger routing unit.

The CRC is a hardware module based on a CRC32 engine that computes the CRC value of the 32-bit data-words presented to it. The source channel of the memory to memory DMA (in memory scan mode) provides data. The data can be optionally forwarded to the destination channel (memory transfer mode). The main features of the CRC peripheral are as follows:

- Memory scan mode
- Memory transfer mode
- Data verify mode
- Data fill mode
- User-programmable CRC32 polynomial
- Bit/byte mirroring option (endianness)
- Fault/error interrupt mechanisms
- 1D and 2D fill block to initialize an array with constants
- 32-bit CRC signature of a block of a memory or an MMR block

Event Handling

The processors provide event handling that supports both nesting and prioritization. Nesting allows multiple event service routines to be active simultaneously. Prioritization ensures that servicing a higher priority event takes precedence over servicing a lower priority event.

The processors provide support for five different types of events:

- An emulation event causes the processors to enter emulation mode, allowing command and control of the processors through the JTAG interface.
- A reset event resets the processors.
- An exceptions event occur synchronously to program flow (in other words, the exception is taken before the instruction is allowed to complete). Conditions triggered on the one side by the SHARC+ core, such as data alignment (SIMD/long word) or compute violations (fixed or floating point), and illegal instructions cause core exceptions. Conditions triggered on the other side by the SEC, such as error correcting codes (ECC)/parity/watchdog/system clock, cause system exceptions.
- An interrupts event occurs asynchronously to program flow. They are caused by input signals, timers, and other peripherals, as well as by an explicit software instruction.

System Event Controller (SEC)

Both SHARC+ cores feature a system event controller. The SEC features include the following:

- Comprehensive system event source management including interrupt enable, fault enable, priority, core mapping, and source grouping
- A distributed programming model where each system event source control and all status fields are independent of each other
- Determinism where all system events have the same propagation delay and provide unique identification of a specific system event source
- A slave control port that provides access to all SEC registers for configuration, status, and interrupt/fault services
- Global locking that supports a register level protection model to prevent writes to locked registers
- Fault management including fault action configuration, time out, external indication, and system reset

Trigger Routing Unit (TRU)

The trigger routing unit (TRU) provides system-level sequence control without core intervention. The TRU maps trigger masters (generators of triggers) to trigger slaves (receivers of triggers). Slave endpoints can be configured to respond to triggers in various ways. Common applications enabled by the TRU include,

- Automatically triggering the start of a DMA sequence after a sequence from another DMA channel completes
- Software triggering
- Synchronization of concurrent activities

blocks on the processor. The digital audio interface carries three types of information: audio data, nonaudio data (compressed data), and timing information.

The S/PDIF interface supports one stereo channel or compressed audio streams. The S/PDIF transmitter and receiver are AES3 compliant and support the sample rate from 24 KHz to 192 KHz. The S/PDIF receiver supports professional jitter standards.

The S/PDIF receiver/transmitter has no separate DMA channels. It receives audio data in serial format and converts it into a biphase encoded signal. The serial data input to the receiver/transmitter can be formatted as left justified, I^2S, or right justified with word widths of 16, 18, 20, or 24 bits. The serial data, clock, and frame sync inputs to the S/PDIF receiver/transmitter are routed through the signal routing unit (SRU). They can come from various sources, such as the SPORTs, external pins, and the precision clock generators (PCGs), and are controlled by the SRU control registers.

Precision Clock Generators (PCG)

The precision clock generators (PCG) consist of four units: units A/B located in the DAI0 block, and units C/D located in the DAI1 block. The PCG can generate a pair of signals (clock and frame sync) derived from a clock input signal (CLKIN1-0, SCLK0, or DAI pin buffer). Each unit can also access the opposite DAI unit. All units are identical in functionality and operate independently of each other. The two signals generated by each unit are normally used as a serial bit clock/frame sync pair.

Enhanced Parallel Peripheral Interface (EPPI)

The processors provide an enhanced parallel peripheral interface (EPPI) that supports data widths up to 24 bits. The EPPI supports direct connection to TFT LCD panels, parallel ADCs and DACs, video encoders and decoders, image sensor modules, and other general-purpose peripherals.

The features supported in the EPPI module include the following:

- Programmable data length of 8 bits, 10 bits, 12 bits, 14 bits, 16 bits, 18 bits, and 24 bits per clock.
- Various framed, nonframed, and general-purpose operating modes. Frame syncs can be generated internally or can be supplied by an external device.
- ITU-656 status word error detection and correction for ITU-656 receive modes and ITU-656 preamble and status word decoding.
- Optional packing and unpacking of data to/from 32 bits from/to 8 bits, 16 bits, and 24 bits. If packing/unpacking is enabled, configure endianness to change the order of packing/unpacking of the bytes/words.
- RGB888 can be converted to RGB666 or RGB565 for transmit modes.
- Various deinterleaving/interleaving modes for receiving/transmitting 4:2:2 YCrCb data.
- Configurable LCD data enable output available on Frame Sync 3.

Universal Asynchronous Receiver/Transmitter (UART) Ports

The processors provide three full-duplex universal asynchronous receiver/transmitter (UART) ports, fully compatible with PC standard UARTs. Each UART port provides a simplified UART interface to other peripherals or hosts, supporting full-duplex, DMA supported, asynchronous transfers of serial data. A UART port includes support for five to eight data bits as well as no parity, even parity, or odd parity.

Optionally, an additional address bit can be transferred to interrupt only addressed nodes in multidrop bus (MDB) systems. A frame is terminated by a configurable number of stop bits.

The UART ports support automatic hardware flow control through the clear to send (CTS) input and request to send (RTS) output with programmable assertion first in, first out (FIFO) levels.

To help support the Local Interconnect Network (LIN) protocols, a special command causes the transmitter to queue a break command of programmable bit length into the transmit buffer. Similarly, the number of stop bits can be extended by a programmable interframe space.

Serial Peripheral Interface (SPI) Ports

The processors have three industry-standard SPI-compatible ports that allow the processors to communicate with multiple SPI-compatible devices.

The baseline SPI peripheral is a synchronous, four-wire interface consisting of two data pins, one device select pin, and a gated clock pin. The two data pins allow full-duplex operation to other SPI-compatible devices. An extra two (optional) data pins are provided to support quad SPI operation. Enhanced modes of operation, such as flow control, fast mode, and dual-I/O mode (DIOM), are also supported. A direct memory access (DMA) mode allows for transferring several words with minimal central processing unit (CPU) interaction.

With a range of configurable options, the SPI ports provide a glueless hardware interface with other SPI-compatible devices in master mode, slave mode, and multimaster environments. The SPI peripheral includes programmable baud rates, clock phase, and clock polarity. The peripheral can operate in a multimaster environment by interfacing with several other devices, acting as either a master device or a slave device. In a multimaster environment, the SPI peripheral uses open-drain outputs to avoid data bus contention. The flow control features enable slow slave devices to interface with fast master devices by providing an SPI ready pin (SPI_RDY) which flexibly controls the transfers.

The baud rate and clock phase/polarities of the SPI port are programmable. The port has integrated DMA channels for both transmit and receive data streams.

Link Ports (LP)

Two 8-bit wide link ports (LP) can connect to the link ports of other DSPs or peripherals. LP are bidirectional ports that have eight data lines, an acknowledge line, and a clock line.

- Automatic detection of IPv4 and IPv6 packets, as well as PTP messages
- Multiple input clock sources (SCLK0, RGMII, RMII, RMII clock, and external clock)
- Programmable pulse per second (PPS) output
- Auxiliary snapshot to time stamp external events

Controller Area Network (CAN)

There are two controller area network (CAN) modules. A CAN controller implements the CAN 2.0B (active) protocol. This protocol is an asynchronous communications protocol used in both industrial and automotive control systems. The CAN protocol is well suited for control applications due to the capability to communicate reliably over a network. This is because the protocol incorporates CRC checking, message error tracking, and fault node confinement.

The CAN controller offers the following features:

- 32 mailboxes (8 receive only, 8 transmit only, 16 configurable for receive or transmit)
- Dedicated acceptance masks for each mailbox
- Additional data filtering on the first two bytes
- Support for both the standard (11-bit) and extended (29-bit) identifier (ID) message formats
- Support for remote frames
- Active or passive network support
- Interrupts, including transmit and receive complete, error, and global

An additional crystal is not required to supply the CAN clock because it is derived from a system clock through a programmable divider.

Timers

The processors include several timers that are described in the following sections.

General-Purpose (GP) Timers (TIMER)

There is one general-purpose (GP) timer unit, providing eight general-purpose programmable timers. Each timer has an external pin that can be configured either as PWM or timer output, as an input to clock the timer, or as a mechanism for measuring pulse widths and periods of external events. These timers can be synchronized to an external clock input on the TM_TMR[n] pins, an external TM_CLK input pin, or to the internal SCLK0.

These timer units can be used in conjunction with the UARTs and the CAN controller to measure the width of the pulses in the data stream to provide a software autobaud detect function for the respective serial channels.

The GP timers can generate interrupts to the processor core, providing periodic events for synchronization to either the system clock or to external signals. Timer events can also trigger other peripherals via the TRU (for instance, to signal a fault). Each timer can also be started and/or stopped by any TRU master without core intervention.

Watchdog Timer (WDT)

Two on-chip software watchdog timers (WDT) can be used by the ARM Cortex-A5 and/or SHARC+ cores. A software watchdog can improve system availability by forcing the processors to a known state, via a general-purpose interrupt, or a fault, if the timer expires before being reset by software.

The programmer initializes the count value of the timer, enables the appropriate interrupt, then enables the timer. Thereafter, the software must reload the counter before it counts down to zero from the programmed value, protecting the system from remaining in an unknown state where software that normally resets the timer stops running due to an external noise condition or software error.

General-Purpose Counters (CNT)

A 32-bit counter (CNT) is provided that can operate in general-purpose up/down count modes and can sense 2-bit quadrature or binary codes as typically emitted by industrial drives or manual thumbwheels. Count direction is either controlled by a level-sensitive input pin or by two edge detectors.

A third counter input can provide flexible zero marker support and can input the push button signal of thumbwheel devices. All three CNT0 pins have a programmable debouncing circuit.

Internal signals forwarded to a GP timer enable this timer to measure the intervals between count events. Boundary registers enable auto-zero operation or simple system warning by interrupts when programmed count values are exceeded.

PCI Express (PCIe)

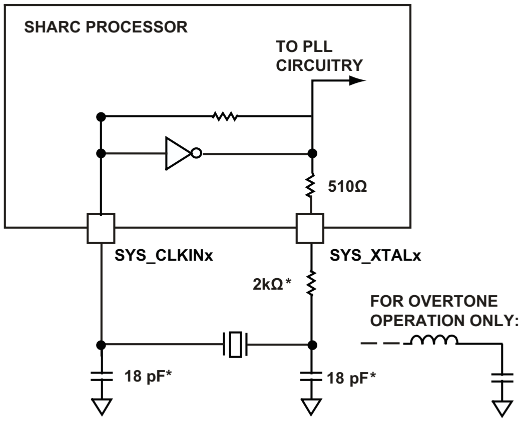
A PCI express interface (PCIe) is available on some product variants (see Table 2 and Table 3). This single, bidirectional lane can be configured to be either a root complex (RC) or end point (EP) system. The PCIe interface has the following features:

- Compliance with the *PCI Express Base Specification 3.0*
- Support for transfers at either 2.5 Gbps (Gen 1) or 5.0 Gbps (Gen 2) in each direction
- Support for 8b/10b encode and decode
- Lane reversal and lane polarity inversion
- Flow control of data in both the transmit and receive directions
- Support for removal of corrupted packets for error detection and recovery
- Maximum transaction payload of 256 bytes

Housekeeping Analog-to-Digital Converter (HADC)

The housekeeping analog-to-digital converter (HADC) provides a general-purpose, multichannel successive approximation ADC. It supports the following set of features:

- 12-bit ADC core (10-bit accuracy) with built in sample and hold.
- Eight single-ended input channels that can be extended to 15 channels by adding an external channel multiplexer.
- Throughput rates up to 1 MSPS.



NOTE: VALUES MARKED WITH * MUST BE CUSTOMIZED, DEPENDING ON THE CRYSTAL AND LAYOUT. ANALYZE CAREFULLY. FOR FREQUENCIES ABOVE 33 MHz, THE SUGGESTED CAPACITOR VALUE OF 18 pF MUST BE TREATED AS A MAXIMUM.

Figure 7. External Crystal Connection

A third overtone crystal can be used for frequencies above 25 MHz. The circuit is then modified to ensure crystal operation only at the third overtone by adding a tuned inductor circuit, shown in Figure 7. A design procedure for the third overtone operation is discussed in detail in "Using Third Overtone Crystals with the ADSP-218x DSP" (EE-168). The same recommendations can be used for the USB crystal oscillator.

Clock Distribution Unit (CDU)

The two CGUs each provide outputs which feed a clock distribution unit (CDU). The clock outputs CLKO0–CLKO9 are connected to various targets. For more information, refer to the ADSP-SC58x/ADSP-2158x SHARC+ Processor Hardware Reference.

Power-Up

SYS_XTALx oscillations (SYS_CLKINx) start when power is applied to the VDD_EXT pins. The rising edge of $\overline{\text{SYS\_HWRST}}$ starts on-chip PLL locking (PLL lock counter). The deassertion must apply only if all voltage supplies and SYS_CLKINx oscillations are valid (refer to the Power-Up Reset Timing section).

Clock Out/External Clock

The SYS_CLKOUT output pin has programmable options to output divided-down versions of the on-chip clocks. By default, the SYS_CLKOUT pin drives a buffered version of the SYS_ CLKIN0 input. Refer to the ADSP-SC58x/ADSP-2158x SHARC+ Processor Hardware Reference to change the default mapping of clocks.

Booting

The processors have several mechanisms for automatically loading internal and external memory after a reset. The boot mode is defined by the SYS_BMODE[n] input pins. There are two categories of boot modes. In master boot mode, the processors actively load data from serial memories. In slave boot modes, the processors receive data from external host devices.

The boot modes are shown in Table 9. These modes are implemented by the SYS_BMODE[n] bits of the reset configuration register and are sampled during power-on resets and software initiated resets.

In the ADSP-SC58x processors, the ARM Cortex-A5 (Core 0) controls the boot process, including loading all internal and external memory. Likewise, in the ADSP-2158x processors, the SHARC+ (Core 1) controls the boot function. The option for secure boot is available on all models.

Table 9. Boot Modes

SYS_BMODE[n] Setting	Boot Mode
000	No boot
001	SPI2 master
010	SPI2 slave
011	Reserved
100	Reserved
101	Reserved
110	Link0 slave
111	UART0 slave

Thermal Monitoring Unit (TMU)

The thermal monitoring unit (TMU) provides on-chip temperature measurement which is important in applications that require substantial power consumption. The TMU is integrated into the processor die and digital infrastructure using an MMR-based system access to measure the die temperature variations in real-time.

TMU features include the following:

- On-chip temperature sensing
- Programmable over temperature and under temperature limits
- Programmable conversion rate
- Averaging feature available

Power Supplies

The processors have separate power supply connections for:

- Internal (VDD_INT)
- External (VDD_EXT)
- USB (VDD_USB)
- HADC (VDD_HADC)
- RTC (VDD_RTC)

GPIO MULTIPLEXING FOR THE 349-BALL CSP_BGA PACKAGE

Table 13 through Table 17 identify the pin functions that are multiplexed on the general-purpose I/O pins of the 349-ball CSP_BGA package.

Table 13. Signal Multiplexing for Port A

Signal Name	Multiplexed Function 0	Multiplexed Function 1	Multiplexed Function 2	Multiplexed Function 3	Multiplexed Function Input Tap
PA_00	ETH0_TXD0			SMC0_A21	
PA_01	ETH0_TXD1			SMC0_A20	
PA_02	ETH0_MDC			SMC0_A24	
PA_03	ETH0_MDIO			SMC0_A23	
PA_04	ETH0_RXD0			SMC0_A19	
PA_05	ETH0_RXD1			SMC0_A18	
PA_06	ETH0_RXCLK_REFCLK			SMC0_A17	
PA_07	ETH0_CRS			SMC0_A16	
PA_08	ETH0_RXD2			SMC0_A12	
PA_09	ETH0_RXD3			SMC0_A11	
PA_10	ETH0_TXEN			SMC0_A22	
PA_11	ETH0_TXCLK			SMC0_A15	
PA_12	ETH0_TXD2			SMC0_A14	
PA_13	ETH0_TXD3			SMC0_A13	
PA_14	ETH0_PTPPPS3	SINC0_D0		SMC0_A10	
PA_15	ETH0_PTPPPS2	SINC0_D1		SMC0_A09	

Table 14. Signal Multiplexing for Port B

Signal Name	Multiplexed Function 0	Multiplexed Function 1	Multiplexed Function 2	Multiplexed Function 3	Multiplexed Function Input Tap
PB_00	ETH0_PTPPPS1	SINC0_D2	PPI0_D14	SMC0_A08	TM0_ACLK3
PB_01	ETH0_PTPPPS0	SINC0_CLK0	PPI0_D15	SMC0_A07	TM0_ACLK4
PB_02	ETH0_PTPCLKIN0	UART1_TX	PPI0_D16	SMC0_A04	
PB_03	ETH0_PTPAUXIN0	UART1_RX	PPI0_D17	SMC0_A03	TM0_ACI1
PB_04	MLB0_CLK	SINC0_D3	PPI0_D12	SMC0_ARDY	ETH0_PTPAUXIN1
PB_05	MLB0_SIG		PPI0_D13	SMC0_A01	ETH0_PTPAUXIN2
PB_06	MLB0_DAT		PWM0_BH	SMC0_A02	ETH0_PTPAUXIN3
PB_07	LP1_D0	PWM0_AH	TM0_TMR3	SMC0_D15	
PB_08	LP1_D1	PWM0_AL	TM0_TMR4	SMC0_D14	
PB_09	LP1_D2		CAN1_TX	SMC0_D13	
PB_10	LP1_D3	TM0_TMR2	CAN1_RX	SMC0_D12	TM0_ACI4
PB_11	LP1_D4		PWM0_DH	SMC0_D11	CNT0_ZM
PB_12	LP1_D5		PWM0_DL	SMC0_D10	CNT0_UD
PB_13	LP1_D6		PWM0_CH	SMC0_D09	
PB_14	LP1_D7	TM0_TMR5	PWM0_CL	SMC0_D08	CNT0_DG
PB_15	LP1_ACK	PWM0_TRIP0	TM0_TMR1	SMC0_AWE	

Table 15. Signal Multiplexing for Port C

Signal Name	Multiplexed Function 0	Multiplexed Function 1	Multiplexed Function 2	Multiplexed Function 3	Multiplexed Function Input Tap
PC_00	LP1_CLK	PWM0_BL	$\overline{\text{SPI0\_SEL4}}$	$\overline{\text{SMC0\_ARE}}$	
PC_01	SPI2_CLK				
PC_02	SPI2_MISO				
PC_03	SPI2_MOSI				
PC_04	SPI2_D2				
PC_05	SPI2_D3				
PC_06	$\overline{\text{SPI2\_SEL1}}$				$\overline{\text{SPI2\_SS}}$
PC_07	CAN0_RX	$\overline{\text{SPI0\_SEL1}}$		$\overline{\text{SMC0\_AMS2}}$	TM0_ACI3
PC_08	CAN0_TX			$\overline{\text{SMC0\_AMS3}}$	
PC_09	SPI0_CLK				
PC_10	SPI0_MISO				
PC_11	SPI0_MOSI				TM0_CLK
PC_12	$\overline{\text{SPI0\_SEL3}}$	SPI0_RDY	ACM0_T0	SMC0_A25	
PC_13	$\overline{\text{UART0\_TX}}$	$\overline{\text{SPI1\_SEL1}}$	ACM0_A0		
PC_14	$\overline{\text{UART0\_RX}}$		ACM0_A1		TM0_ACI0
PC_15	$\overline{\text{UART0\_RTS}}$	PPI0_FS3	ACM0_A2	$\overline{\text{SMC0\_AMS0}}$	

Table 16. Signal Multiplexing for Port D

Signal Name	Multiplexed Function 0	Multiplexed Function 1	Multiplexed Function 2	Multiplexed Function 3	Multiplexed Function Input Tap
PD_00	$\overline{\text{UART0\_CTS}}$	PPI0_D23	ACM0_A3	SMC0_D07	
PD_01	$\overline{\text{SPI0\_SEL2}}$		ACM0_A4	$\overline{\text{SMC0\_AOE}}$	$\overline{\text{SPI0\_SS}}$
PD_02	LP0_D0	$\overline{\text{PWM1\_TRIP0}}$	TRACE0_D00		
PD_03	LP0_D1	PWM1_AH	TRACE0_D01		
PD_04	LP0_D2	PWM1_AL	TRACE0_D02		
PD_05	LP0_D3	PWM1_BH	TRACE0_D03		
PD_06	LP0_D4	PWM1_BL	TRACE0_D04		
PD_07	LP0_D5	PWM1_CH	TRACE0_D05		
PD_08	LP0_D6	PWM1_CL	TRACE0_D06		TM0_ACLK1
PD_09	LP0_D7	PWM1_DH	TRACE0_D07		TM0_ACLK2
PD_10	LP0_CLK	PWM1_DL	TRACE0_CLK		
PD_11	LP0_ACK	PWM1_SYNC			
PD_12	$\overline{\text{UART2\_TX}}$		PPI0_D19	SMC0_A06	
PD_13	$\overline{\text{UART2\_RX}}$		PPI0_D18	SMC0_A05	TM0_ACI2
PD_14	PPI0_D11	$\overline{\text{PWM2\_TRIP0}}$	MLB0_CLKOUT	SMC0_D06	
PD_15	PPI0_D10	PWM2_CH		SMC0_D05	

Table 19. ADSP-SC58x/ADSP-2158x 529-Ball CSP_BGA Signal Descriptions (Continued)

Signal Name	Description	Port	Pin Name
$\overline{\text{DMC0\_CK}}$	DMC0 Clock (complement)	Not Muxed	$\overline{\text{DMC0\_CK}}$
$\overline{\text{DMC0\_CS0}}$	DMC0 Chip Select 0	Not Muxed	$\overline{\text{DMC0\_CS0}}$
DMC0_DQ00	DMC0 Data 0	Not Muxed	DMC0_DQ00
DMC0_DQ01	DMC0 Data 1	Not Muxed	DMC0_DQ01
DMC0_DQ02	DMC0 Data 2	Not Muxed	DMC0_DQ02
DMC0_DQ03	DMC0 Data 3	Not Muxed	DMC0_DQ03
DMC0_DQ04	DMC0 Data 4	Not Muxed	DMC0_DQ04
DMC0_DQ05	DMC0 Data 5	Not Muxed	DMC0_DQ05
DMC0_DQ06	DMC0 Data 6	Not Muxed	DMC0_DQ06
DMC0_DQ07	DMC0 Data 7	Not Muxed	DMC0_DQ07
DMC0_DQ08	DMC0 Data 8	Not Muxed	DMC0_DQ08
DMC0_DQ09	DMC0 Data 9	Not Muxed	DMC0_DQ09
DMC0_DQ10	DMC0 Data 10	Not Muxed	DMC0_DQ10
DMC0_DQ11	DMC0 Data 11	Not Muxed	DMC0_DQ11
DMC0_DQ12	DMC0 Data 12	Not Muxed	DMC0_DQ12
DMC0_DQ13	DMC0 Data 13	Not Muxed	DMC0_DQ13
DMC0_DQ14	DMC0 Data 14	Not Muxed	DMC0_DQ14
DMC0_DQ15	DMC0 Data 15	Not Muxed	DMC0_DQ15
DMC0_LDM	DMC0 Data Mask for Lower Byte	Not Muxed	DMC0_LDM
DMC0_LDQS	DMC0 Data Strobe for Lower Byte	Not Muxed	DMC0_LDQS
$\overline{\text{DMC0\_LDQS}}$	DMC0 Data Strobe for Lower Byte (complement)	Not Muxed	$\overline{\text{DMC0\_LDQS}}$
DMC0_ODT	DMC0 On-die termination	Not Muxed	DMC0_ODT
$\overline{\text{DMC0\_RAS}}$	DMC0 Row Address Strobe	Not Muxed	$\overline{\text{DMC0\_RAS}}$
$\overline{\text{DMC0\_RESET}}$	DMC0 Reset (DDR3 only)	Not Muxed	$\overline{\text{DMC0\_RESET}}$
DMC0_RZQ	DMC0 External calibration resistor connection	Not Muxed	DMC0_RZQ
DMC0_UDM	DMC0 Data Mask for Upper Byte	Not Muxed	DMC0_UDM
DMC0_UDQS	DMC0 Data Strobe for Upper Byte	Not Muxed	DMC0_UDQS
$\overline{\text{DMC0\_UDQS}}$	DMC0 Data Strobe for Upper Byte (complement)	Not Muxed	$\overline{\text{DMC0\_UDQS}}$
DMC0_VREF	DMC0 Voltage Reference	Not Muxed	DMC0_VREF
$\overline{\text{DMC0\_WE}}$	DMC0 Write Enable	Not Muxed	$\overline{\text{DMC0\_WE}}$
DMC1_A00	DMC1 Address 0	Not Muxed	DMC1_A00
DMC1_A01	DMC1 Address 1	Not Muxed	DMC1_A01
DMC1_A02	DMC1 Address 2	Not Muxed	DMC1_A02
DMC1_A03	DMC1 Address 3	Not Muxed	DMC1_A03
DMC1_A04	DMC1 Address 4	Not Muxed	DMC1_A04
DMC1_A05	DMC1 Address 5	Not Muxed	DMC1_A05
DMC1_A06	DMC1 Address 6	Not Muxed	DMC1_A06
DMC1_A07	DMC1 Address 7	Not Muxed	DMC1_A07
DMC1_A08	DMC1 Address 8	Not Muxed	DMC1_A08
DMC1_A09	DMC1 Address 9	Not Muxed	DMC1_A09
DMC1_A10	DMC1 Address 10	Not Muxed	DMC1_A10
DMC1_A11	DMC1 Address 11	Not Muxed	DMC1_A11
DMC1_A12	DMC1 Address 12	Not Muxed	DMC1_A12
DMC1_A13	DMC1 Address 13	Not Muxed	DMC1_A13
DMC1_A14	DMC1 Address 14	Not Muxed	DMC1_A14
DMC1_A15	DMC1 Address 15	Not Muxed	DMC1_A15
DMC1_BA0	DMC1 Bank Address 0	Not Muxed	DMC1_BA0
DMC1_BA1	DMC1 Bank Address 1	Not Muxed	DMC1_BA1

Table 19. ADSP-SC58x/ADSP-2158x 529-Ball CSP_BGA Signal Descriptions (Continued)

Signal Name	Description	Port	Pin Name
PWM2_AL	PWM2 Channel A Low Side	F	PF_06
PWM2_BH	PWM2 Channel B High Side	F	PF_09
PWM2_BL	PWM2 Channel B Low Side	F	PF_08
PWM2_CH	PWM2 Channel C High Side	D	PD_15
PWM2_CL	PWM2 Channel C Low Side	E	PE_00
PWM2_DH	PWM2 Channel D High Side	E	PE_04
PWM2_DL	PWM2 Channel D Low Side	E	PE_10
PWM2_SYNC	PWM2 PWMTMR Grouped	E	PE_05
PWM2_TRIP0	PWM2 Shutdown Input 0	D	PD_14
GND	Ground	Not Muxed	GND
VDD_EXT	External Voltage Domain	Not Muxed	VDD_EXT
VDD_INT	Internal Voltage Domain	Not Muxed	VDD_INT
RTC0_CLKIN	RTC0 Crystal input / external oscillator connection	Not Muxed	RTC0_CLKIN
RTC0_XTAL	RTC0 Crystal output	Not Muxed	RTC0_XTAL
SINC0_CLK0	SINC0 Clock 0	B	PB_01
SINC0_D0	SINC0 Data 0	A	PA_14
SINC0_D1	SINC0 Data 1	A	PA_15
SINC0_D2	SINC0 Data 2	B	PB_00
SINC0_D3	SINC0 Data 3	B	PB_04
SMC0_A01	SMC0 Address 1	B	PB_05
SMC0_A02	SMC0 Address 2	B	PB_06
SMC0_A03	SMC0 Address 3	B	PB_03
SMC0_A04	SMC0 Address 4	B	PB_02
SMC0_A05	SMC0 Address 5	D	PD_13
SMC0_A06	SMC0 Address 6	D	PD_12
SMC0_A07	SMC0 Address 7	B	PB_01
SMC0_A08	SMC0 Address 8	B	PB_00
SMC0_A09	SMC0 Address 9	A	PA_15
SMC0_A10	SMC0 Address 10	A	PA_14
SMC0_A11	SMC0 Address 11	A	PA_09
SMC0_A12	SMC0 Address 12	A	PA_08
SMC0_A13	SMC0 Address 13	A	PA_13
SMC0_A14	SMC0 Address 14	A	PA_12
SMC0_A15	SMC0 Address 15	A	PA_11
SMC0_A16	SMC0 Address 16	A	PA_07
SMC0_A17	SMC0 Address 17	A	PA_06
SMC0_A18	SMC0 Address 18	A	PA_05
SMC0_A19	SMC0 Address 19	A	PA_04
SMC0_A20	SMC0 Address 20	A	PA_01
SMC0_A21	SMC0 Address 21	A	PA_00
SMC0_A22	SMC0 Address 22	A	PA_10
SMC0_A23	SMC0 Address 23	A	PA_03
SMC0_A24	SMC0 Address 24	A	PA_02
SMC0_A25	SMC0 Address 25	C	PC_12
SMC0_ABE0	SMC0 Byte Enable 0	E	PE_14
SMC0_ABE1	SMC0 Byte Enable 1	E	PE_15
SMC0_AMS0	SMC0 Memory Select 0	C	PC_15
SMC0_AMS1	SMC0 Memory Select 1	E	PE_13

Table 19. ADSP-SC58x/ADSP-2158x 529-Ball CSP_BGA Signal Descriptions (Continued)

Signal Name	Description	Port	Pin Name
$\overline{\text{SMC0\_AMS2}}$	SMC0 Memory Select 2	C	PC_07
$\overline{\text{SMC0\_AMS3}}$	SMC0 Memory Select 3	C	PC_08
$\overline{\text{SMC0\_AOE}}$	SMC0 Output Enable	D	PD_01
SMC0_ARDY	SMC0 Asynchronous Ready	B	PB_04
$\overline{\text{SMC0\_ARE}}$	SMC0 Read Enable	C	PC_00
$\overline{\text{SMC0\_AWE}}$	SMC0 Write Enable	B	PB_15
SMC0_D00	SMC0 Data 0	E	PE_12
SMC0_D01	SMC0 Data 1	E	PE_11
SMC0_D02	SMC0 Data 2	E	PE_10
SMC0_D03	SMC0 Data 3	E	PE_09
SMC0_D04	SMC0 Data 4	E	PE_00
SMC0_D05	SMC0 Data 5	D	PD_15
SMC0_D06	SMC0 Data 6	D	PD_14
SMC0_D07	SMC0 Data 7	D	PD_00
SMC0_D08	SMC0 Data 8	B	PB_14
SMC0_D09	SMC0 Data 9	B	PB_13
SMC0_D10	SMC0 Data 10	B	PB_12
SMC0_D11	SMC0 Data 11	B	PB_11
SMC0_D12	SMC0 Data 12	B	PB_10
SMC0_D13	SMC0 Data 13	B	PB_09
SMC0_D14	SMC0 Data 14	B	PB_08
SMC0_D15	SMC0 Data 15	B	PB_07
SPI0_CLK	SPI0 Clock	C	PC_09
SPI0_MISO	SPI0 Master In, Slave Out	C	PC_10
SPI0_MOSI	SPI0 Master Out, Slave In	C	PC_11
SPI0_RDY	SPI0 Ready	C	PC_12
$\overline{\text{SPI0\_SEL1}}$	SPI0 Slave Select Output 1	C	PC_07
$\overline{\text{SPI0\_SEL2}}$	SPI0 Slave Select Output 2	D	PD_01
$\overline{\text{SPI0\_SEL3}}$	SPI0 Slave Select Output 3	C	PC_12
$\overline{\text{SPI0\_SEL4}}$	SPI0 Slave Select Output 4	C	PC_00
$\overline{\text{SPI0\_SEL5}}$	SPI0 Slave Select Output 5	E	PE_01
$\overline{\text{SPI0\_SEL6}}$	SPI0 Slave Select Output 6	E	PE_02
$\overline{\text{SPI0\_SEL7}}$	SPI0 Slave Select Output 7	E	PE_03
$\overline{\text{SPI0\_SS}}$	SPI0 Slave Select Input	D	PD_01
SPI1_CLK	SPI1 Clock	E	PE_13
SPI1_MISO	SPI1 Master In, Slave Out	E	PE_14
SPI1_MOSI	SPI1 Master Out, Slave In	E	PE_15
SPI1_RDY	SPI1 Ready	E	PE_08
$\overline{\text{SPI1\_SEL1}}$	SPI1 Slave Select Output 1	C	PC_13
$\overline{\text{SPI1\_SEL2}}$	SPI1 Slave Select Output 2	E	PE_07
$\overline{\text{SPI1\_SEL3}}$	SPI1 Slave Select Output 3	E	PE_11
$\overline{\text{SPI1\_SEL4}}$	SPI1 Slave Select Output 4	E	PE_12
$\overline{\text{SPI1\_SEL5}}$	SPI1 Slave Select Output 5	E	PE_08
$\overline{\text{SPI1\_SEL6}}$	SPI1 Slave Select Output 6	F	PF_00
$\overline{\text{SPI1\_SEL7}}$	SPI1 Slave Select Output 7	F	PF_01
$\overline{\text{SPI1\_SS}}$	SPI1 Slave Select Input	E	PE_11
SPI2_CLK	SPI2 Clock	C	PC_01
SPI2_D2	SPI2 Data 2	C	PC_04

Table 27. ADSP-SC58x/ADSP-2158x Designer Quick Reference (Continued)

Signal Name	Type	Driver Type	Int Term	Reset Term	Reset Drive	Power Domain	Description and Notes
DMC0_CKE	Output	B	none	none	L	VDD_DMC	Desc: DMC0 Clock enable Notes: No notes
$\overline{\text{DMC0\_CK}}$	Output	C	none	none	L	VDD_DMC	Desc: DMC0 Clock (complement) Notes: No notes
$\overline{\text{DMC0\_CS0}}$	Output	B	none	none	none	VDD_DMC	Desc: DMC0 Chip Select 0 Notes: No notes
DMC0_DQ00	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 0 Notes: No notes
DMC0_DQ01	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 1 Notes: No notes
DMC0_DQ02	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 2 Notes: No notes
DMC0_DQ03	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 3 Notes: No notes
DMC0_DQ04	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 4 Notes: No notes
DMC0_DQ05	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 5 Notes: No notes
DMC0_DQ06	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 6 Notes: No notes
DMC0_DQ07	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 7 Notes: No notes
DMC0_DQ08	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 8 Notes: No notes
DMC0_DQ09	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 9 Notes: No notes
DMC0_DQ10	InOut	B	Internal logic ensures that input signal does not float	none	none	VDD_DMC	Desc: DMC0 Data 10 Notes: No notes

Asynchronous Flash Write

Table 49 and Figure 16 show asynchronous flash memory write timing, related to the SMC.

Table 49. Asynchronous Flash Write

Parameter		Min	Max	Unit
Switching Characteristics				
t_{AMSADV}	SMC0_Ax/$\overline{\text{SMC0\_AMSx}}$ Assertion Before ADV Low[1]	PREST × t_{SCLK0} − 2		ns
$t_{DADVAWE}$	$\overline{\text{SMC0\_AWE}}$ Low Delay From ADV High[2]	PREAT × t_{SCLK0} − 2		ns
t_{WADV}	NR_ADV Active Low Width[3]	WST × t_{SCLK0} − 2		ns
t_{HAWE}	Output[4] Hold After $\overline{\text{SMC0\_AWE}}$ High[5]	WHT × t_{SCLK0} − 3.5		ns
t_{WAWE}[6]	$\overline{\text{SMC0\_AWE}}$ Active Low Width[7]	WAT × t_{SCLK0} − 2		ns

[1] PREST value set using the SMC_BxETIM.PREST bits.
[2] PREAT value set using the SMC_BxETIM.PREAT bits.
[3] WST value set using the SMC_BxTIM.WST bits.
[4] Output signals are DATA, SMC0_Ax, $\overline{\text{SMC0\_AMSx}}$, $\overline{\text{SMC0\_ABEx}}$.
[5] WHT value set using the SMC_BxTIM.WHT bits.
[6] SMC_BxCTL.ARDYEN bit = 0.
[7] WAT value set using the SMC_BxTIM.WAT bits.

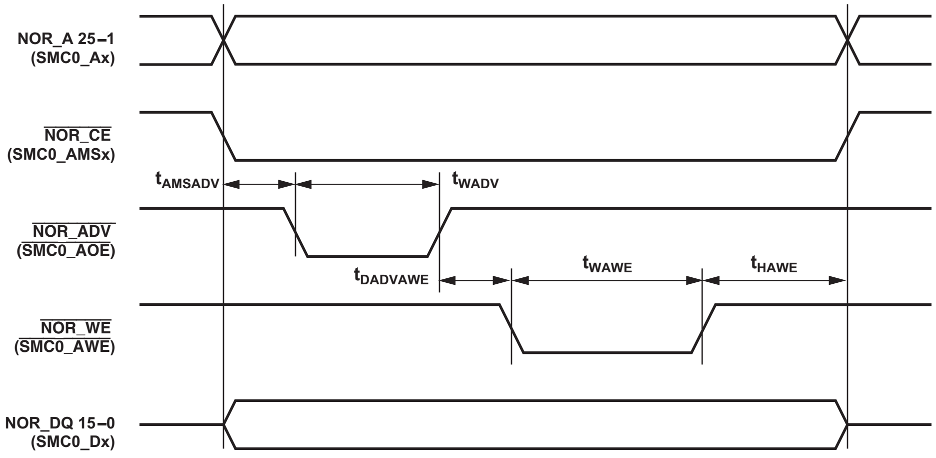


Figure 16. Asynchronous Flash Write

All Accesses

Table 50 describes timing that applies to all memory accesses, related to the SMC.

Table 50. All Accesses

Parameter		Min	Max	Unit
Switching Characteristic				
t_{TURN}	$\overline{\text{SMC0\_AMSx}}$ Inactive Width	(IT + TT) × t_{SCLK0} − 2		ns

DDR3 SDRAM Read Cycle Timing

Table 58 and Figure 24 show mobile DDR3 SDRAM read cycle timing, related to the DMC.

Table 58. DDR3 SDRAM Read Cycle Timing VDD_DMCx Nominal 1.5 V[1]

Parameter		450 MHz[2] Min	450 MHz[2] Max	Unit
Timing Requirements				
t_{DQSQ}	DMCx_DQS to DMCx_DQ Skew for DMCx_DQS and Associated DMCx_DQ Signals		0.2	ns
t_{QH}	DMCx_DQ, DMCx_DQS Output Hold Time From DMCx_DQS	0.38		t_{CK}
t_{RPRE}	Read Preamble	0.9		t_{CK}
t_{RPST}	Read Postamble	0.3		t_{CK}

[1] Specifications apply to both DMC0 and DMC1.

[2] To ensure proper operation of the DDR3, all the DDR3 guidelines must be strictly followed. See "Interfacing DDR3/DDR2/LPDDR Memory to ADSP-SC5xx/215xx Processors" (EE-387).

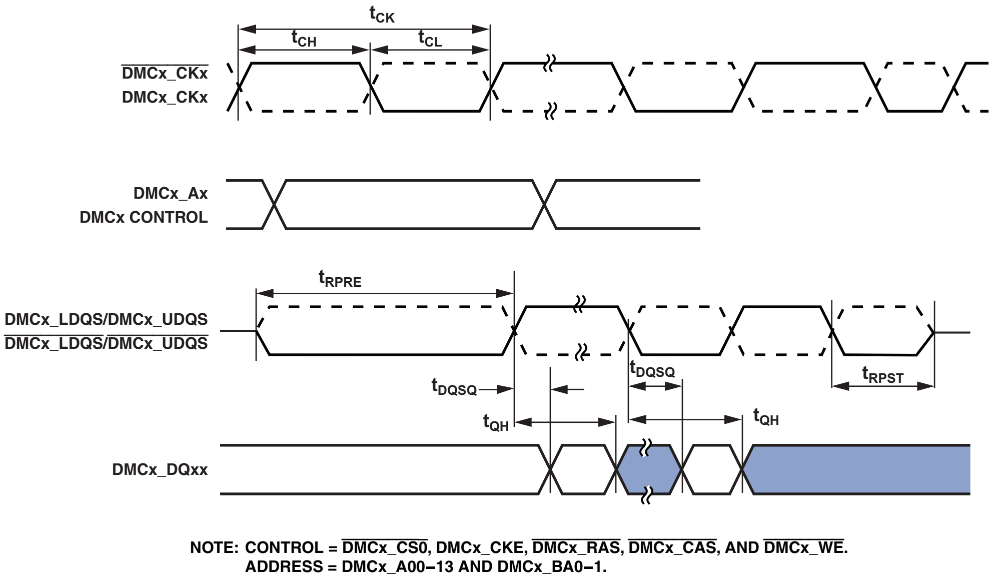


Figure 24. DDR3 SDRAM Controller Input AC Timing

The SPTx_TDV output signal becomes active in SPORT multichannel mode. During transmit slots (enabled with active channel selection registers) the SPTx_TDV is asserted for communication with external devices.

Table 67. Serial Ports—TDV (Transmit Data Valid)[1]

Parameter		Min	Max	Unit
Switching Characteristics				
t_{DRDVEN}	Data Valid Enable Delay from Drive Edge of External Clock[2]	2		ns
t_{DFDVEN}	Data Valid Disable Delay from Drive Edge of External Clock[2]		14	ns
t_{DRDVIN}	Data Valid Enable Delay from Drive Edge of Internal Clock[2]	–2.5		ns
t_{DFDVIN}	Data Valid Disable Delay from Drive Edge of Internal Clock[2]		3.5	ns

[1] Specifications apply to all eight SPORTs.
[2] Referenced to drive edge.

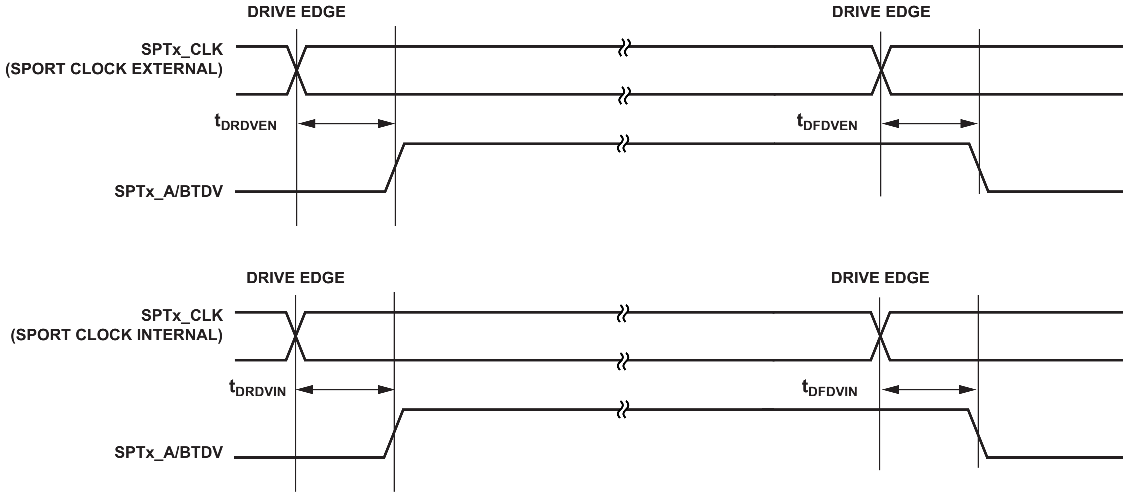


Figure 39. Serial Ports—Transmit Data Valid Internal and External Clock

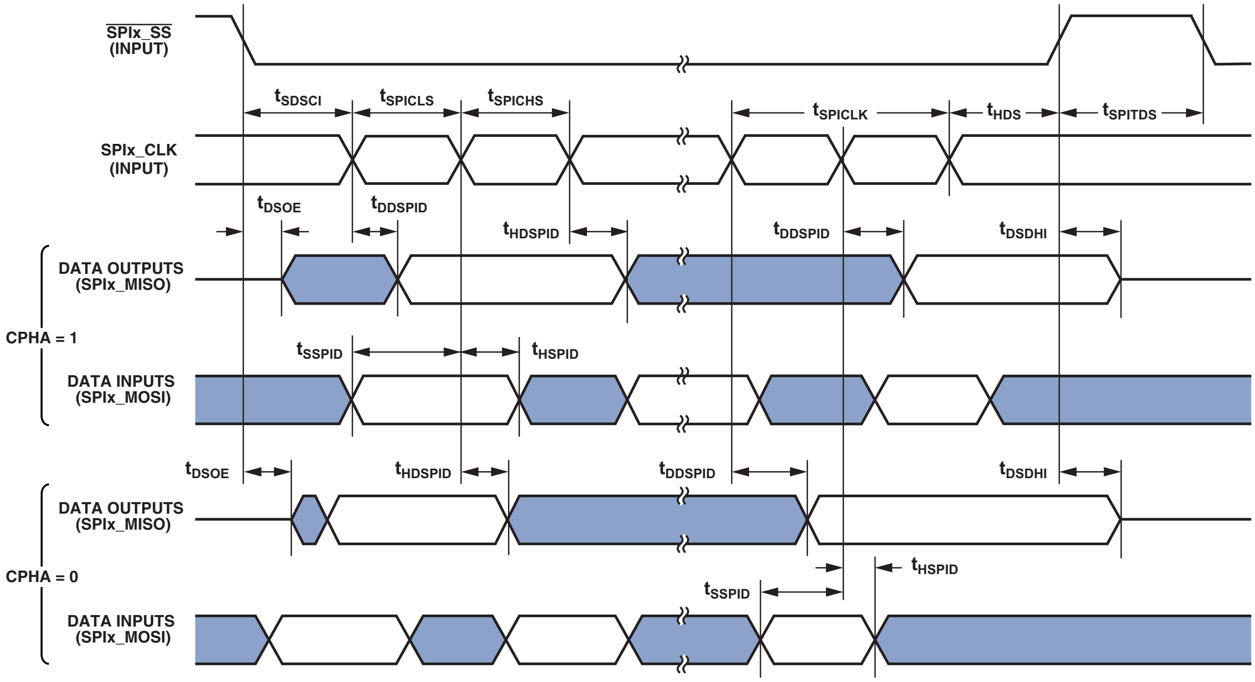


Figure 44. SPI Port—Slave Timing

SPI Port—SPIx_RDY Slave Timing

SPIx_RDY is used to provide flow control. CPOL, CPHA, and FCCH are configuration bits in the SPIx_CTL register.

Table 73. SPI Port—SPIx_RDY Slave Timing[1]

Parameter		Conditions	Min	Max	Unit
Switching Characteristic					
$t_{DSPISCKRDYS}$	SPIx_RDY Deassertion from Last Valid Input SPIx_CLK Edge	FCCH = 0	$3 \times t_{SCLK1}$	$4 \times t_{SCLK1} + 10$	ns
		FCCH = 1	$4 \times t_{SCLK1}$	$5 \times t_{SCLK1} + 10$	ns

[1] All specifications apply to all three SPIs.

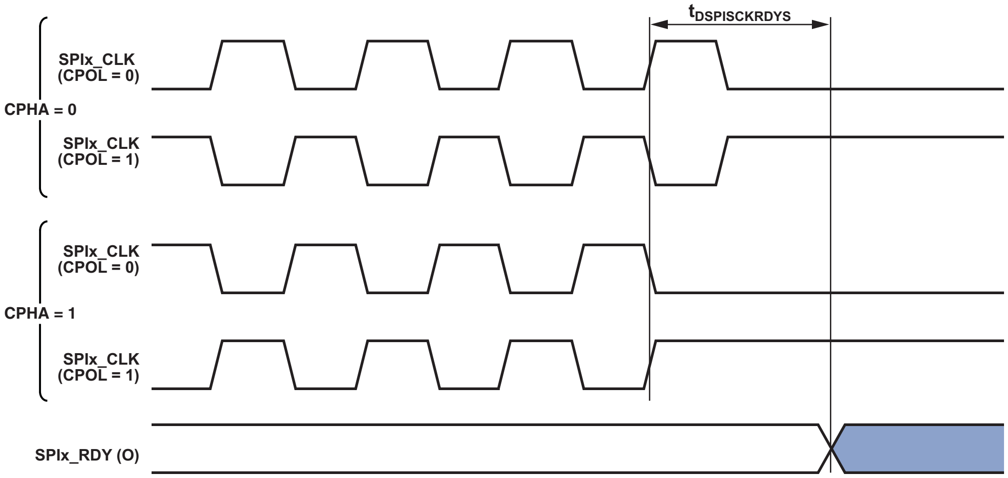


Figure 45. SPIx_RDY Deassertion from Valid Input SPIx_CLK Edge in Slave Mode

10/100 EMAC Timing (ETH0 and ETH1)

Table 88 through Table 90 and Figure 59 through Figure 61 describe the 10/100 EMAC operations.

Table 88. 10/100 EMAC Timing—RMII Receive Signal[1]

Parameter[2]		Min	Max	Unit
Timing Requirements				
$t_{REFCLKF}$	ETHx_REFCLK Frequency (f_{SCLK0} = SCLK0 Frequency)		50 + 1%	MHz
$t_{REFCLKW}$	ETHx_REFCLK Width ($t_{REFCLKF}$ = ETHx_REFCLK Period)	$t_{REFCLKF} \times 35\%$	$t_{REFCLKF} \times 65\%$	ns
$t_{REFCLKIS}$	Rx Input Valid to RMII ETHx_REFCLK Rising Edge (Data In Setup)	1.75		ns
$t_{REFCLKIH}$	RMII ETHx_REFCLK Rising Edge to Rx Input Invalid (Data In Hold)	1.6		ns

[1] These specifications apply to ETH0 and ETH1.

[2] RMII inputs synchronous to RMII ETHx_REFCLK are ETHx_RXD1–0, RMII ETHx_CRS, and ERxER.

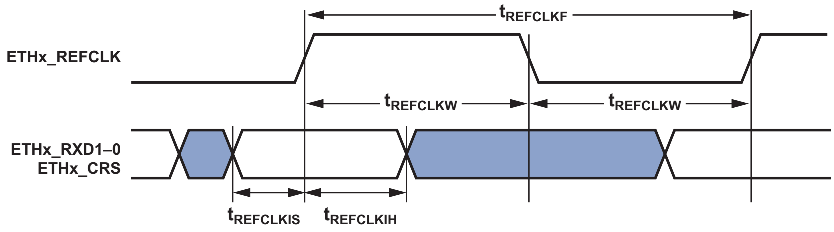


Figure 59. 10/100 EMAC Controller Timing—RMII Receive Signal

Table 89. 10/100 EMAC Timing—RMII Transmit Signal[1]

Parameter[2]		Min	Max	Unit
Switching Characteristics				
$t_{REFCLKOV}$	RMII ETHx_REFCLK Rising Edge to Transmit Output Valid (Data Out Valid)		11.9	ns
$t_{REFCLKOH}$	RMII ETHx_REFCLK Rising Edge to Transmit Output Invalid (Data Out Hold)	2		ns

[1] These specifications apply to ETH0 and ETH1.

[2] RMII outputs synchronous to RMII ETHx_REFCLK are ETHx_TXD1–0.

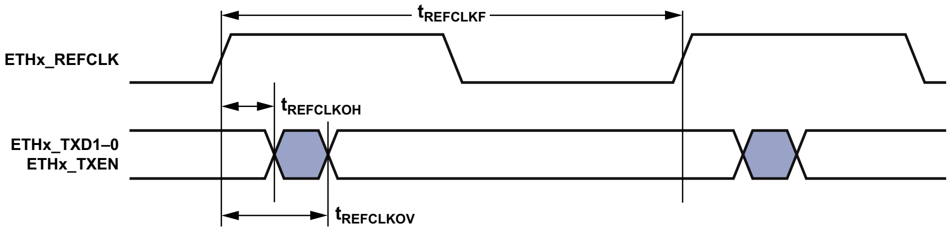


Figure 60. 10/100 EMAC Controller Timing—RMII Transmit Signal

Media Local Bus (MLB)

All the numbers shown in Table 99 are applicable for all MLB speed modes (1024 FS, 512 FS, and 256 FS) for the 3-pin protocol, unless otherwise specified. Refer to the *Media Local Bus Specification version 4.2* for more details.

Table 99. 3-Pin MLB Interface Specifications

Parameter		Min	Typ	Max	Unit
t_{MLBCLK}	MLB Clock Period				
	1024 FS		20.3		ns
	512 FS		40		ns
	256 FS		81		ns
t_{MCKL}	MLBCLK Low Time				
	1024 FS	6.1			ns
	512 FS	14			ns
	256 FS	30			ns
t_{MCKH}	MLBCLK High Time				
	1024 FS	9.3			ns
	512 FS	14			ns
	256 FS	30			ns
t_{MCKR}	MLBCLK Rise Time (V_{IL} to V_{IH})				
	1024 FS			1	ns
	512 FS/256 FS			3	ns
t_{MCKF}	MLBCLK Fall Time (V_{IH} to V_{IL})				
	1024 FS			1	ns
	512 FS/256 FS			3	ns
t_{MPWV}[1]	MLBCLK Pulse Width Variation				
	1024 FS			0.7	nspp
	512 FS/256			2.0	nspp
t_{DSMCF}	DAT/SIG Input Setup Time	1			ns
t_{DHMCF}	DAT/SIG Input Hold Time	2			ns
t_{MCFDZ}	DAT/SIG Output Time to Three-State	0		15	ns
t_{MCDRV}	DAT/SIG Output Data Delay From MLBCLK Rising Edge			8	ns
t_{MDZH}[2]	Bus Hold Time				
	1024 FS	2			ns
	512 FS/256	4			ns
C_{MLB}	DAT/SIG Pin Load				
	1024 FS			40	pf
	512 FS/256			60	pf

[1] Pulse width variation is measured at 1.25 V by triggering on one edge of MLBCLK and measuring the spread on the other edge, measured in ns peak-to-peak.

[2] Board designs must ensure the high impedance bus does not leave the logic state of the final driven bit for this time period. Therefore, coupling must be minimized while meeting the maximum capacitive load listed.

D13317-0-10/16(0)

ANALOG DEVICES

www.analog.com