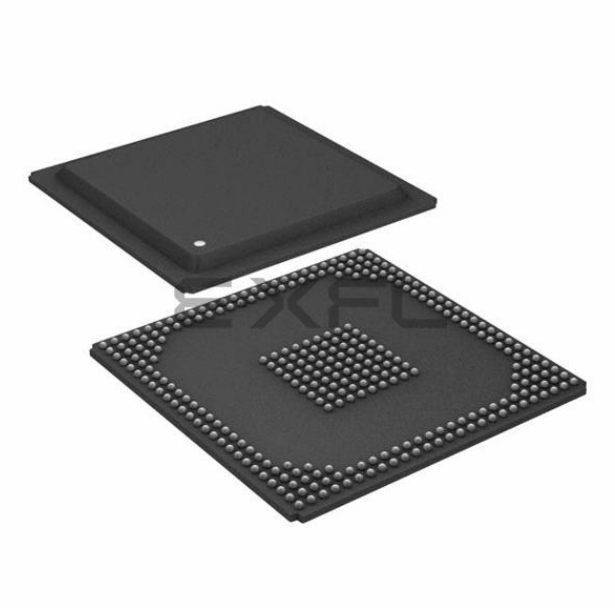

Welcome to E-XFL.COM

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	SPI, SSP, UART
Clock Rate	600MHz
Non-Volatile Memory	External
On-Chip RAM	328kB
Voltage - I/O	2.50V, 3.30V
Voltage - Core	1.35V
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	297-BGA
Supplier Device Package	297-PBGA (27x27)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/adsp-bf561sbb600

ADSP-BF561* PRODUCT PAGE QUICK LINKS

Last Content Update: 02/23/2017

COMPARABLE PARTS

View a parametric search of comparable parts.

EVALUATION KITS

- Low Cost ICE-1000 and High Performance ICE-2000 USB-based JTAG Emulators
- Multimedia Starter Kit
- The ADSP-BF561 EZ-Kit Lite evaluation hardware provides a low-cost hardware solution for evaluating the ADSP-BF561 Blackfin processor.
- USB-Based Emulator and High Performance USB-Based Emulator

DOCUMENTATION

Application Notes

- AN-813: Interfacing the ADSP-BF533/ADSP-BF561 Blackfin®; Processors to High Speed Parallel ADCs
- EE-120: Interfacing Assembly Language Programs to C
- EE-126: The ABCs of SDRAMemories
- EE-175: Emulator and Evaluation Hardware Troubleshooting Guide for VisualDSP++ Users
- EE-183: Rational Sample Rate Conversion with Blackfin® Processors
- EE-185: Fast Floating-Point Arithmetic Emulation on Blackfin® Processors
- EE-228: Switching Regulator Design Considerations for ADSP-BF533 Blackfin® Processors
- EE-261: Understanding Jitter Requirements of PLL-Based Processors
- EE-269: A Beginner's Guide to Ethernet 802.3
- EE-281: Hardware Design Checklist for the Blackfin® Processors
- EE-289: Implementing FAT32 File Systems on ADSP-BF533 Blackfin® Processors
- EE-293: Estimating Power for ADSP-BF561 Blackfin® Processors
- EE-294: Energy-Aware Programming on Blackfin Processors
- EE-300: Interfacing Blackfin® EZ-KIT Lite® Boards to CMOS Image Sensors
- EE-314: Booting the ADSP-BF561 Blackfin® Processor
- EE-323: Implementing Dynamically Loaded Software Modules
- EE-326: Blackfin® Processor and SDRAM Technology
- EE-330: Windows Vista Compatibility in VisualDSP++ 5.0 Development Tools
- EE-332: Cycle Counting and Profiling
- EE-336: Putting ADSP-BF54x Blackfin® Processor Booting into Practice
- EE-339: Using External Switching Regulators with Blackfin® Processors
- EE-340: Connecting SHARC® and Blackfin® Processors over SPI
- EE-356: Emulator and Evaluation Hardware Troubleshooting Guide for CCES Users

Data Sheet

GENERAL DESCRIPTION

The ADSP-BF561 processor is a high performance member of the Blackfin® family of products targeting a variety of multimedia, industrial, and telecommunications applications. At the heart of this device are two independent Analog Devices Blackfin processors. These Blackfin processors combine a dual-MAC state-of-the-art signal processing engine, the advantage of a clean, orthogonal RISC-like microprocessor instruction set, and single instruction, multiple data (SIMD) multimedia capabilities in a single instruction set architecture.

The ADSP-BF561 processor has 328K bytes of on-chip memory. Each Blackfin core includes:

- 16K bytes of instruction SRAM/cache
- 16K bytes of instruction SRAM
- 32K bytes of data SRAM/cache
- 32K bytes of data SRAM
- 4K bytes of scratchpad SRAM

Additional on-chip memory peripherals include:

- 128K bytes of low latency on-chip L2 SRAM
- Four-channel internal memory DMA controller
- External memory controller with glueless support for SDRAM, mobile SDRAM, SRAM, and flash.

PORTABLE LOW POWER ARCHITECTURE

Blackfin processors provide world-class power management and performance. Blackfin processors are designed in a low power and low voltage design methodology and feature dynamic power management, the ability to vary both the voltage and frequency of operation to significantly lower overall power consumption. Varying the voltage and frequency can result in a substantial reduction in power consumption, compared with just varying the frequency of operation. This translates into longer battery life for portable appliances.

BLACKFIN PROCESSOR CORE

As shown in Figure 2, each Blackfin core contains two multiplier/accumulators (MACs), two 40-bit ALUs, four video ALUs, and a single shifter. The computational units process 8-bit, 16-bit, or 32-bit data from the register file.

Each MAC performs a 16-bit by 16-bit multiply in every cycle, with accumulation to a 40-bit result, providing eight bits of extended precision. The ALUs perform a standard set of arithmetic and logical operations. With two ALUs capable of operating on 16-bit or 32-bit data, the flexibility of the computation units covers the signal processing requirements of a varied set of application needs.

Each of the two 32-bit input registers can be regarded as two 16-bit halves, so each ALU can accomplish very flexible single 16-bit arithmetic operations. By viewing the registers as pairs of 16-bit operands, dual 16-bit or single 32-bit operations can be accomplished in a single cycle. By further taking advantage of the second ALU, quad 16-bit operations can be accomplished simply, accelerating the per cycle throughput.

The powerful 40-bit shifter has extensive capabilities for performing shifting, rotating, normalization, extraction, and depositing of data. The data for the computational units is found in a multiported register file of sixteen 16-bit entries or eight 32-bit entries.

A powerful program sequencer controls the flow of instruction execution, including instruction alignment and decoding. The sequencer supports conditional jumps and subroutine calls, as well as zero overhead looping. A loop buffer stores instructions locally, eliminating instruction memory accesses for tight looped code.

Two data address generators (DAGs) provide addresses for simultaneous dual operand fetches from memory. The DAGs share a register file containing four sets of 32-bit Index, Modify, Length, and Base registers. Eight additional 32-bit registers provide pointers for general indexing of variables and stack locations.

Blackfin processors support a modified Harvard architecture in combination with a hierarchical memory structure. Level 1 (L1) memories are those that typically operate at the full processor speed with little or no latency. Level 2 (L2) memories are other memories, on-chip or off-chip, that may take multiple processor cycles to access. At the L1 level, the instruction memory holds instructions only. The two data memories hold data, and a dedicated scratchpad data memory stores stack and local variable information. At the L2 level, there is a single unified memory space, holding both instructions and data.

In addition, half of L1 instruction memory and half of L1 data memory may be configured as either Static RAMs (SRAMs) or caches. The Memory Management Unit (MMU) provides memory protection for individual tasks that may be operating on the core and may protect system registers from unintended access.

The architecture provides three modes of operation: user mode, supervisor mode, and emulation mode. User mode has restricted access to certain system resources, thus providing a protected software environment, while supervisor mode has unrestricted access to the system and core resources.

The Blackfin instruction set has been optimized so that 16-bit op-codes represent the most frequently used instructions, resulting in excellent compiled code density. Complex DSP instructions are encoded into 32-bit op-codes, representing fully featured multifunction instructions. Blackfin processors support a limited multi-issue capability, where a 32-bit instruction can be issued in parallel with two 16-bit instructions, allowing the programmer to use many of the core resources in a single instruction cycle.

The Blackfin assembly language uses an algebraic syntax for ease of coding and readability. The architecture has been optimized for use in conjunction with the VisualDSP C/C++ compiler, resulting in fast and efficient software implementations.

flexible configuration and upgradability of system memory while allowing the core to view all SDRAM as a single, contiguous, physical address space.

The asynchronous memory controller can also be programmed to control up to four banks of devices with very flexible timing parameters for a wide variety of devices. Each bank occupies a 64M byte segment regardless of the size of the devices used so that these banks will only be contiguous if fully populated with 64M bytes of memory.

I/O Memory Space

Blackfin processors do not define a separate I/O space. All resources are mapped through the flat 32-bit address space. On-chip I/O devices have their control registers mapped into memory mapped registers (MMRs) at addresses near the top of the 4G byte address space. These are separated into two smaller blocks, one which contains the control MMRs for all core functions, and the other which contains the registers needed for setup and control of the on-chip peripherals outside of the core. The core MMRs are accessible only by the core and only in supervisor mode and appear as reserved space by on-chip peripherals. The system MMRs are accessible by the core in supervisor mode and can be mapped as either visible or reserved to other devices, depending on the system protection model desired.

Booting

The ADSP-BF561 contains a small boot kernel, which configures the appropriate peripheral for booting. If the ADSP-BF561 is configured to boot from boot ROM memory space, the processor starts executing from the on-chip boot ROM.

Event Handling

The event controller on the ADSP-BF561 handles all asynchronous and synchronous events to the processor. The ADSP-BF561 provides event handling that supports both nesting and prioritization. Nesting allows multiple event service routines to be active simultaneously. Prioritization ensures that servicing of a higher priority event takes precedence over servicing of a lower priority event. The controller provides support for five different types of events:

- Emulation – An emulation event causes the processor to enter emulation mode, allowing command and control of the processor via the JTAG interface.
- Reset – This event resets the processor.
- Nonmaskable Interrupt (NMI) – The NMI event can be generated by the software watchdog timer or by the NMI input signal to the processor. The NMI event is frequently used as a power-down indicator to initiate an orderly shutdown of the system.
- Exceptions – Events that occur synchronously to program flow, i.e., the exception will be taken before the instruction is allowed to complete. Conditions such as data alignment violations or undefined instructions cause exceptions.
- Interrupts – Events that occur asynchronously to program flow. They are caused by timers, peripherals, input pins, and an explicit software instruction.

Each event has an associated register to hold the return address and an associated "return from event" instruction. When an event is triggered, the state of the processor is saved on the supervisor stack.

The ADSP-BF561 event controller consists of two stages: the Core Event Controller (CEC) and the System Interrupt Controller (SIC). The Core Event Controller works with the System Interrupt Controller to prioritize and control all system events. Conceptually, interrupts from the peripherals enter into the SIC, and are then routed directly into the general-purpose interrupts of the CEC.

Core Event Controller (CEC)

The CEC supports nine general-purpose interrupts (IVG15–7), in addition to the dedicated interrupt and exception events. Of these general-purpose interrupts, the two lowest priority interrupts (IVG15–14) are recommended to be reserved for software interrupt handlers, leaving seven prioritized interrupt inputs to support the peripherals of the ADSP-BF561. Table 1 describes the inputs to the CEC, identifies their names in the Event Vector Table (EVT), and lists their priorities.

Table 1. Core Event Controller (CEC)

Priority (0 is Highest)	Event Class	EVT Entry
0	Emulation/Test Control	EMU
1	Reset	RST
2	Nonmaskable Interrupt	NMI
3	Exceptions	EVX
4	Global Enable	
5	Hardware Error	IVHW
6	Core Timer	IVTMR
7	General Interrupt 7	IVG7
8	General Interrupt 8	IVG8
9	General Interrupt 9	IVG9
10	General Interrupt 10	IVG10
11	General Interrupt 11	IVG11
12	General Interrupt 12	IVG12
13	General Interrupt 13	IVG13
14	General Interrupt 14	IVG14
15	General Interrupt 15	IVG15

System Interrupt Controller (SIC)

The System Interrupt Controller provides the mapping and routing of events from the many peripheral interrupt sources to the prioritized general-purpose interrupt inputs of the CEC. Although the ADSP-BF561 provides a default mapping, the user can alter the mappings and priorities of interrupt events by

even though the event may be latched in the ILAT register. This register may be read from or written to while in supervisor mode.

Note that general-purpose interrupts can be globally enabled and disabled with the STI and CLI instructions, respectively.

- CEC Interrupt Pending Register (IPEND) – The IPEND register keeps track of all nested events. A set bit in the IPEND register indicates the event is currently active or nested at some level. This register is updated automatically by the controller but may be read while in supervisor mode.

The SIC allows further control of event processing by providing six 32-bit interrupt control and status registers. Each register contains a bit corresponding to each of the peripheral interrupt events shown in Table 2.

- SIC Interrupt Mask Registers (SIC_IMASKx) – These registers control the masking and unmasking of each peripheral interrupt event. When a bit is set in these registers, that peripheral event is unmasked and will be processed by the system when asserted. A cleared bit in these registers masks the peripheral event, thereby preventing the processor from servicing the event.
- SIC Interrupt Status Registers (SIC_ISRx) – As multiple peripherals can be mapped to a single event, these registers allow the software to determine which peripheral event source triggered the interrupt. A set bit indicates the peripheral is asserting the interrupt; a cleared bit indicates the peripheral is not asserting the event.
- SIC Interrupt Wakeup Enable Registers (SIC_IWRx) – By enabling the corresponding bit in these registers, each peripheral can be configured to wake up the processor, should the processor be in a powered-down mode when the event is generated.

Because multiple interrupt sources can map to a single general-purpose interrupt, multiple pulse assertions can occur simultaneously, before or during interrupt processing for an interrupt event already detected on this interrupt input. The IPEND register contents are monitored by the SIC as the interrupt acknowledgement.

The appropriate ILAT register bit is set when an interrupt rising edge is detected (detection requires two core clock cycles). The bit is cleared when the respective IPEND register bit is set. The IPEND bit indicates that the event has entered into the processor pipeline. At this point the CEC will recognize and queue the next rising edge event on the corresponding event input. The minimum latency from the rising edge transition of the general-purpose interrupt to the IPEND output asserted is three core clock cycles; however, the latency can be much higher, depending on the activity within and the mode of the processor.

DMA CONTROLLERS

The ADSP-BF561 has two independent DMA controllers that support automated data transfers with minimal overhead for the DSP cores. DMA transfers can occur between the ADSP-BF561 internal memories 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 interfaces, including the SDRAM controller and the asynchronous memory controller. DMA-capable peripherals include the SPORTs, SPI port, UART, and PPIs. Each individual DMA-capable peripheral has at least one dedicated DMA channel.

The ADSP-BF561 DMA controllers support both 1-dimensional (1-D) and 2-dimensional (2-D) DMA transfers. DMA transfer initialization can be implemented from registers or from sets of parameters called descriptor blocks.

The 2-D DMA capability supports arbitrary row and column sizes up to 64K elements by 64K elements, and arbitrary row and column step sizes up to ± 32K elements. Furthermore, the column step size can be less than the row step size, allowing implementation of interleaved data streams. This feature is especially useful in video applications where data can be de-interleaved on the fly.

Examples of DMA types supported by the ADSP-BF561 DMA controllers include:

- A single linear buffer that stops upon completion.
- A circular autorefreshing buffer that interrupts on each full or fractionally full buffer.
- 1-D or 2-D DMA using a linked list of descriptors.
- 2-D DMA using an array of descriptors, specifying only the base DMA address within a common page.

In addition to the dedicated peripheral DMA channels, each DMA Controller has four memory DMA channels provided for transfers between the various memories of the ADSP-BF561 system. These enable transfers of blocks of data between any of the memories—including external SDRAM, ROM, SRAM, and flash memory—with minimal processor intervention. Memory DMA transfers can be controlled by a very flexible descriptor-based methodology or by a standard register-based autobuffer mechanism.

Further, the ADSP-BF561 has a four channel Internal Memory DMA (IMDMA) Controller. The IMDMA Controller allows data transfers between any of the internal L1 and L2 memories.

WATCHDOG TIMER

Each ADSP-BF561 core includes a 32-bit timer, which can be used to implement a software watchdog function. A software watchdog can improve system availability by forcing the processor to a known state, via generation of a hardware reset, nonmaskable interrupt (NMI), or general-purpose interrupt, 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 to zero from the programmed value. This protects the system from remaining in an unknown state where software, which would normally reset the timer, has stopped running due to an external noise condition or software error.

The core clock (CCLK) frequency can also be dynamically changed by means of the CSEL1–0 bits of the PLL_DIV register. Supported CCLK divider ratios are 1, 2, 4, and 8, as shown in Table 6. This programmable core clock capability is useful for fast core frequency modifications.

Table 6. Core Clock Ratios

Signal Name CSEL1–0	Divider Ratio VCO/CCLK	Example Frequency Ratios (MHz)	
		VCO	CCLK
00	1:1	500	500
01	2:1	500	250
10	4:1	200	50
11	8:1	200	25

The maximum PLL clock time when a change is programmed via the PLL_CTL register is 40 µs. The maximum time to change the internal voltage via the internal voltage regulator is also 40 µs. The reset value for the PLL_LOCKCNT register is 0x200. This value should be programmed to ensure a 40 µs wakeup time when either the voltage is changed or a new MSEL value is programmed. The value should be programmed to ensure an 80 µs wakeup time when both voltage and the MSEL value are changed. The time base for the PLL_LOCKCNT register is the period of CLKIN.

BOOTING MODES

The ADSP-BF561 has three mechanisms (listed in Table 7) for automatically loading internal L1 instruction memory, L2, or external memory after a reset. A fourth mode is provided to execute from external memory, bypassing the boot sequence.

Table 7. Booting Modes

BMODE1–0	Description
00	Execute from 16-bit external memory (Bypass Boot ROM)
01	Boot from 8-bit/16-bit flash
10	Boot from SPI host slave mode
11	Boot from SPI serial EEPROM (16-, 24-bit address range)

The BMODE pins of the reset configuration register, sampled during power-on resets and software initiated resets, implement the following modes:

- Execute from 16-bit external memory – Execution starts from address 0x2000 0000 with 16-bit packing. The boot ROM is bypassed in this mode. All configuration settings are set for the slowest device possible (3-cycle hold time, 15-cycle R/W access times, 4-cycle setup). Note that, in bypass mode, only Core A can execute instructions from external memory.
- Boot from 8-bit/16-bit external flash memory – The 8-bit/16-bit flash boot routine located in boot ROM memory space is set up using Asynchronous Memory Bank 0. All configuration settings are set for the slowest device possible (3-cycle hold time; 15-cycle R/W access times; 4-cycle setup).
- Boot from SPI host device – The Blackfin processor operates in SPI slave mode and is configured to receive the bytes of the .LDR file from an SPI host (master) agent. To hold off the host device from transmitting while the boot ROM is busy, the Blackfin processor asserts a GPIO pin, called host wait (HWAIT), to signal the host device not to send any more bytes until the flag is deasserted. The flag is chosen by the user and this information is transferred to the Blackfin processor via bits 10:5 of the FLAG header.
- Boot from SPI serial EEPROM (16-, 24-bit addressable) – The SPI uses the PF2 output pin to select a single SPI EPROM device, submits a read command at address 0x0000, and begins clocking data into the beginning of L1 instruction memory. A 16-, 24-bit addressable SPI-compatible EPROM must be used.

For each of the boot modes, a boot loading protocol is used to transfer program and data blocks from an external memory device to their specified memory locations. Multiple memory blocks may be loaded by any boot sequence. Once all blocks are loaded, Core A program execution commences from the start of L1 instruction SRAM (0xFFA0 0000). Core B remains in a held-off state until Bit 5 of SICA_SYSCR is cleared by Core A. After that, Core B will start execution at address 0xFF60 0000.

In addition, Bit 4 of the reset configuration register can be set by application code to bypass the normal boot sequence during a software reset. For this case, the processor jumps directly to the beginning of L1 instruction memory.

INSTRUCTION SET DESCRIPTION

The Blackfin processor family assembly language instruction set employs an algebraic syntax that was designed for ease of coding and readability. The instructions have been specifically tuned to provide a flexible, densely encoded instruction set that compiles to a very small final memory size. The instruction set also provides fully featured multifunction instructions that allow the programmer to use many of the processor core resources in a single instruction. Coupled with many features more often seen on microcontrollers, this instruction set is very efficient when compiling C and C++ source code. In addition, the architecture supports both a user (algorithm/application code) and a supervisor (O/S kernel, device drivers, debuggers, ISRs) mode of operation—allowing multiple levels of access to core processor resources.

The assembly language, which takes advantage of the processor's unique architecture, offers the following advantages:

- Seamlessly integrated DSP/CPU features are optimized for both 8-bit and 16-bit operations.
- A multi-issue load/store modified Harvard architecture, which supports two 16-bit MAC or four 8-bit ALU plus two load/store plus two pointer updates per cycle.

- All registers, I/O, and memory are mapped into a unified 4G byte memory space providing a simplified programming model.
- Microcontroller features, such as arbitrary bit and bit-field manipulation, insertion, and extraction; integer operations on 8-, 16-, and 32-bit data types; and separate user and kernel stack pointers.
- Code density enhancements, which include intermixing of 16-bit and 32-bit instructions (no mode switching, no code segregation). Frequently used instructions are encoded as 16-bits.

DEVELOPMENT TOOLS

The ADSP-BF561 is supported with a complete set of CROSSCORE®† software and hardware development tools, including Analog Devices emulators and the VisualDSP++®‡ development environment. The same emulator hardware that supports other Analog Devices processors also fully emulates the ADSP-BF561.

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++ runtime library that includes DSP and mathematical functions. A key point for these tools is C/C++ code efficiency. The compiler has been developed for efficient translation of C/C++ code to Blackfin assembly. The Blackfin processor has architectural features that improve the efficiency of compiled C/C++ code.

The VisualDSP++ debugger has a number of important features. Data visualization is enhanced by a plotting package that offers a significant level of flexibility. This graphical representation of user data enables the programmer to quickly determine the performance of an algorithm. As algorithms grow in complexity, this capability can have increasing significance on the designer's development schedule, increasing productivity. Statistical profiling enables the programmer to nonintrusively poll the processor as it is running the program. This feature, unique to VisualDSP++, enables the software developer to passively gather important code execution metrics without interrupting the real-time characteristics of the program. Essentially, the developer can identify bottlenecks in software quickly and efficiently. By using the profiler, the programmer can focus on those areas in the program that impact performance and take corrective action.

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 breakpoints.
- 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.
- Perform source level debugging.
- Create custom debugger windows.

The VisualDSP++ IDE lets programmers define and manage software development. Its dialog boxes and property pages let programmers configure and manage all development tools, including color syntax highlighting in the VisualDSP++ editor. These capabilities permit programmers to:

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

The VisualDSP++ Kernel (VDK) incorporates scheduling and resource management tailored specifically to address the memory and timing constraints of embedded, real-time programming. These capabilities enable engineers to develop code more effectively, eliminating the need to start from the very beginning when developing new application code. The VDK features include threads, critical and unscheduled regions, semaphores, events, and device flags. The VDK also supports priority-based, pre-emptive, cooperative, and time-sliced scheduling approaches. In addition, the VDK was designed to be scalable. If the application does not use a specific feature, the support code for that feature is excluded from the target system.

Because the VDK is a library, a developer can decide whether to use it or not. The VDK is integrated into the VisualDSP++ development environment, but can also be used with standard command line tools. When the VDK is used, the development environment assists the developer with many error prone tasks and assists in managing system resources, automating the generation of various VDK-based objects, and visualizing the system state when debugging an application that uses the VDK.

The Expert Linker can be used to visually manipulate the placement of code and data in the embedded system. Memory utilization can be viewed in a color-coded graphical form. Code and data can be easily moved to different areas of the processor or external memory with the drag of the mouse. Runtime stack and heap usage can be examined. The Expert Linker is fully compatible with existing Linker Definition File (LDF), allowing the developer to move between the graphical and textual environments.

Analog Devices emulators use the IEEE 1149.1 JTAG test access port of the ADSP-BF561 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 the loading or timing of the target system.

† CROSSCORE is a registered trademark of Analog Devices, Inc.
‡ VisualDSP++ is a registered trademark of Analog Devices, Inc.

Table 8. Pin Descriptions (Continued)

Pin Name	Type	Function	Driver Type[1]
PF/SPI/TIMER			
PF0/*SPISS*/*TMR0*	I/O	Programmable Flag/*Slave SPI Select/Timer*	C
PF1/*SPISEL1*/*TMR1*	I/O	Programmable Flag/*SPI Select/Timer*	C
PF2/*SPISEL2*/*TMR2*	I/O	Programmable Flag/*SPI Select/Timer*	C
PF3/*SPISEL3*/*TMR3*	I/O	Programmable Flag/*SPI Select/Timer*	C
PF4/*SPISEL4*/*TMR4*	I/O	Programmable Flag/*SPI Select/Timer*	C
PF5/*SPISEL5*/*TMR5*	I/O	Programmable Flag/*SPI Select/Timer*	C
PF6/*SPISEL6*/*TMR6*	I/O	Programmable Flag/*SPI Select/Timer*	C
PF7/*SPISEL7*/*TMR7*	I/O	Programmable Flag/*SPI Select/Timer*	C
PF8	I/O	Programmable Flag	C
PF9	I/O	Programmable Flag	C
PF10	I/O	Programmable Flag	C
PF11	I/O	Programmable Flag	C
PF12	I/O	Programmable Flag	C
PF13	I/O	Programmable Flag	C
PF14	I/O	Programmable Flag	C
PF15/*EXT CLK*	I/O	Programmable Flag/*External Timer Clock Input*	C
PPI0			
PPI0D15–8/*PF47–40*	I/O	PPI Data/*Programmable Flag Pins*	C
PPI0D7–0	I/O	PPI Data Pins	C
PPI0CLK	I	PPI Clock	
PPI0SYNC1/*TMR8*	I/O	PPI Sync/*Timer*	C
PPI0SYNC2/*TMR9*	I/O	PPI Sync/*Timer*	C
PPI0SYNC3	I/O	PPI Sync	C
PPI1			
PPI1D15–8/*PF39–32*	I/O	PPI Data/*Programmable Flag Pins*	C
PPI1D7–0	I/O	PPI Data Pins	C
PPI1CLK	I	PPI Clock	
PPI1SYNC1/*TMR10*	I/O	PPI Sync/*Timer*	C
PPI1SYNC2/*TMR11*	I/O	PPI Sync/*Timer*	C
PPI1SYNC3	I/O	PPI Sync	C
SPORT0			
RSCLK0/*PF28*	I/O	Sport0 Receive Serial Clock/*Programmable Flag*	D
RFS0/*PF19*	I/O	Sport0 Receive Frame Sync/*Programmable Flag*	C
DR0PRI	I	Sport0 Receive Data Primary	
DR0SEC/*PF20*	I/O	Sport0 Receive Data Secondary/*Programmable Flag*	C
TSCLK0/*PF29*	I/O	Sport0 Transmit Serial Clock/*Programmable Flag*	D
TFS0/*PF16*	I/O	Sport0 Transmit Frame Sync/*Programmable Flag*	C
DT0PRI/*PF18*	I/O	Sport0 Transmit Data Primary/*Programmable Flag*	C
DT0SEC/*PF17*	I/O	Sport0 Transmit Data Secondary/*Programmable Flag*	C

ABSOLUTE MAXIMUM RATINGS

Stresses greater than those listed in Table 13 may cause permanent damage to the device. These are stress ratings only. 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.

Table 13. Absolute Maximum Ratings

Parameter	Value
Internal (Core) Supply Voltage (V_{DDINT})	–0.3 V to +1.42 V
External (I/O) Supply Voltage (V_{DDEXT})	–0.5 V to +3.8 V
Input Voltage[1]	–0.5 V to +3.8 V
Output Voltage Swing	–0.5 V to V_{DDEXT} + 0.5 V
Load Capacitance[2]	200 pF
Storage Temperature Range	–65°C to +150°C
Junction Temperature Under Bias	125°C

[1] Applies to 100% transient duty cycle. For other duty cycles see Table 14.
[2] For proper SDRAM controller operation, the maximum load capacitance is 50 pF (at 3.3 V) or 30 pF (at 2.5 V) for ADDR19–1, DATA15–0, ABE1–0/SDQM1–0, CLKOUT, SCKE, SA10, SRAS, SCAS, SWE, and SMS.

Table 14. Maximum Duty Cycle for Input Transient Voltage[1]

V_{IN} Min (V)	V_{IN} Max (V)[2]	Maximum Duty Cycle
–0.50	3.80	100%
–0.70	4.00	40%
–0.80	4.10	25%
–0.90	4.20	15%
–1.00	4.30	10%

[1] Applies to all signal pins with the exception of CLKIN, XTAL, VROUT1–0.
[2] Only one of the listed options can apply to a particular design.

PACKAGE INFORMATION

The information presented in Figure 7 and Table 15 provides details about the package branding for the Blackfin processors. For a complete listing of product availability, see the Ordering Guide on Page 63.

Figure 7. Product Information on Package

Table 15. Package Brand Information

Brand Key	Field Description
t	Temperature Range
pp	Package Type
Z	RoHS Compliant Part
ccc	See Ordering Guide
vvvvvv.x	Assembly Lot Code
n.n	Silicon Revision
yyww	Date Code

ESD SENSITIVITY

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

TIMING SPECIFICATIONS

Clock and Reset Timing

Table 16 and Figure 8 describe clock and reset operations. Per Absolute Maximum Ratings on Page 22, combinations of CLKIN and clock multipliers must not result in core/system clocks exceeding the maximum limits allowed for the processor, including system clock restrictions related to supply voltage.

Table 16. Clock and Normal Reset Timing

Parameter		Min	Max	Unit
Timing Requirements				
t_{CKIN}	CLKIN (to PLL) Period[1, 2, 3]	25.0	100.0	ns
t_{CKINL}	CLKIN Low Pulse	10.0		ns
t_{CKINH}	CLKIN High Pulse	10.0		ns
t_{WRST}	$\overline{\text{RESET}}$ Asserted Pulse Width Low[4]	$11 \times t_{CKIN}$		ns

[1] If DF bit in PLL_CTL register is set t_{CLKIN} is divided by two before going to PLL, then the t_{CLKIN} maximum period is 50 ns and the t_{CLKIN} minimum period is 12.5 ns.
[2] Applies to PLL bypass mode and PLL nonbypass mode.
[3] Combinations of the CLKIN frequency and the PLL clock multiplier must not exceed the allowed f_{VCO}, f_{CCLK}, and f_{SCLK} settings discussed in Table 9 on Page 20 through Table 12 on Page 21.
[4] Applies after power-up sequence is complete. See Table 17 and Figure 9 for power-up reset timing.

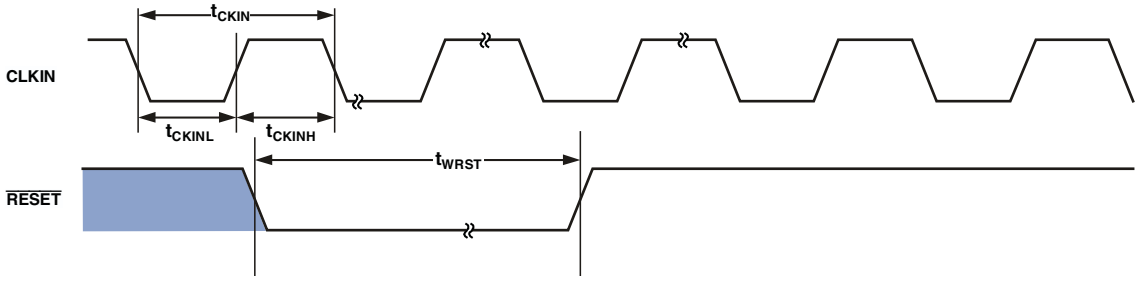

Figure 8. Clock and Normal Reset Timing

Table 17. Power-Up Reset Timing

Parameter		Min	Max	Unit
Timing Requirements				
$t_{RST\_IN\_PWR}$	$\overline{\text{RESET}}$ Deasserted after the V_{DDINT}, V_{DDEXT}, and CLKIN Pins are Stable and Within Specification	$3500 \times t_{CKIN}$		µs

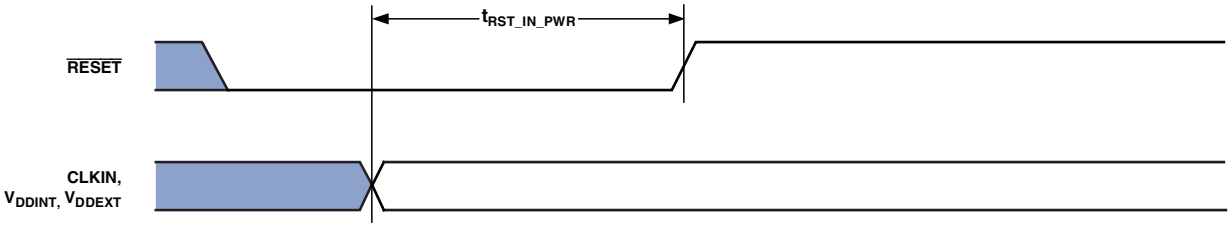

Figure 9. Power-Up Reset Timing

Parallel Peripheral Interface Timing

Table 22, and Figure 14 through Figure 17 on Page 30, describe default Parallel Peripheral Interface operations.

If bit 4 of the PLL_CTL register is set, then Figure 18 on Page 30 and Figure 19 on Page 31 apply.

Table 22. Parallel Peripheral Interface Timing

Parameter		Min	Max	Unit
Timing Requirements				
t_{PCLKW}	PPIxCLK Width[1]	5.0		ns
t_{PCLK}	PPIxCLK Period[1]	13.3		ns
t_{SFSPE}	External Frame Sync Setup Before PPIxCLK	4.0		ns
t_{HFSPE}	External Frame Sync Hold After PPIxCLK	1.0		ns
t_{SDRPE}	Receive Data Setup Before PPIxCLK	3.5		ns
t_{HDRPE}	Receive Data Hold After PPIxCLK	2.0		ns
Switching Characteristics				
t_{DFSPE}	Internal Frame Sync Delay After PPIxCLK		8.0	ns
t_{HOFSPE}	Internal Frame Sync Hold After PPIxCLK	1.7		ns
t_{DDTPE}	Transmit Data Delay After PPIxCLK		8.0	ns
t_{HDTPE}	Transmit Data Hold After PPIxCLK	2.0		ns

[1] For PPI modes that use an internally generated frame sync, the PPIxCLK frequency cannot exceed $f_{SCLK}/2$. For modes with no frame syncs or external frame syncs, PPIxCLK cannot exceed 75 MHz and f_{SCLK} should be equal to or greater than PPIxCLK.

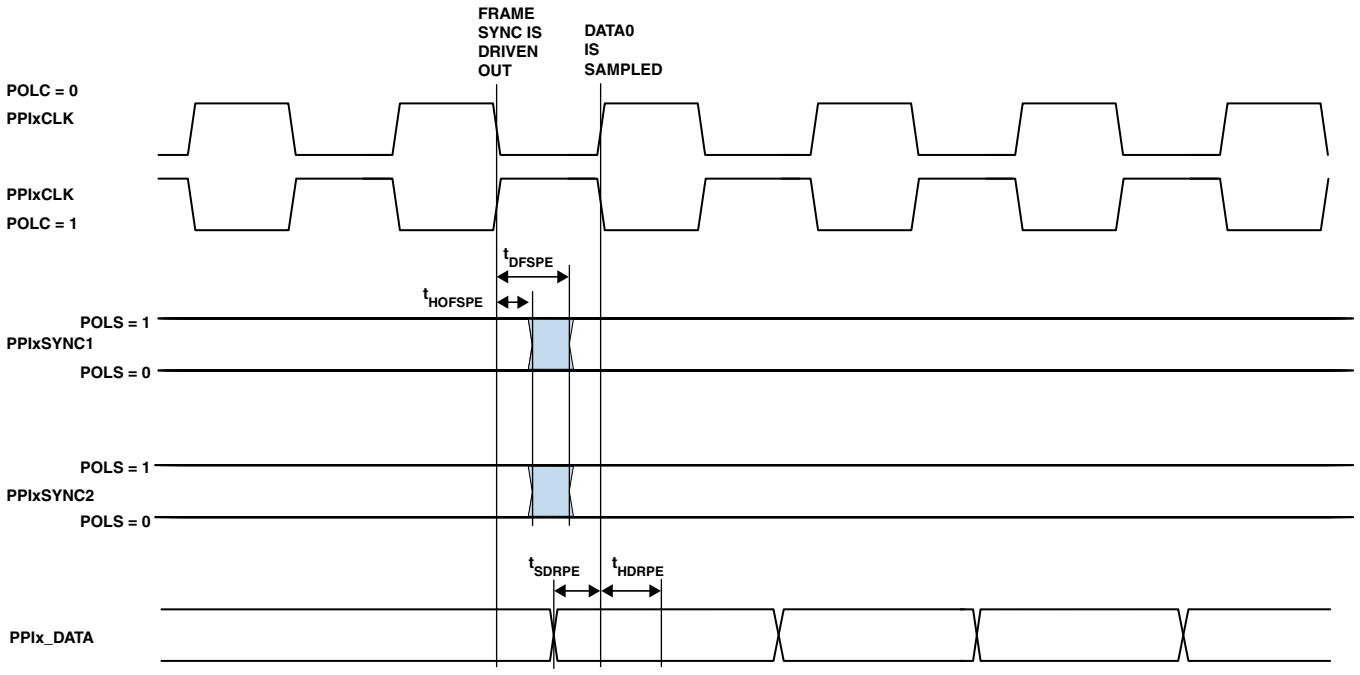

Figure 14. PPI GP Rx Mode with Internal Frame Sync Timing (Default)

Table 25. Serial Ports—Enable and Three-State

Parameter		Min	Max	Unit
Switching Characteristics				
t_{DTENE}	Data Enable Delay from External TSCLKx[1]	0		ns
t_{DDTTE}	Data Disable Delay from External TSCLKx[1]		10.0	ns
t_{DTENI}	Data Enable Delay from Internal TSCLKx[1]	–2.0		ns
t_{DDTTI}	Data Disable Delay from Internal TSCLKx[1]		3.0	ns

[1] Referenced to drive edge.

Table 26. External Late Frame Sync

Parameter		Min	Max	Unit
Switching Characteristics				
$t_{DDTLFSE}$	Data Delay from Late External TFSx or External RFSx with MCMEN = 1, MFD = 0[1, 2]		10.0	ns
$t_{DTENLFS}$	Data Enable from Late FS or MCMEN = 1, MFD = 0[1, 2]	0		ns

[1] MCMEN = 1, TFSx enable and TFSx valid follow $t_{DTENLFS}$ and $t_{DDTLFSE}$.
[2] If external RFSx/TFSx setup to RSCLKx/TSCLKx > $t_{SCLKE}/2$, then $t_{DDTTE/I}$ and $t_{DTENE/I}$ apply; otherwise $t_{DDTLFSE}$ and $t_{DTENLFS}$ apply.

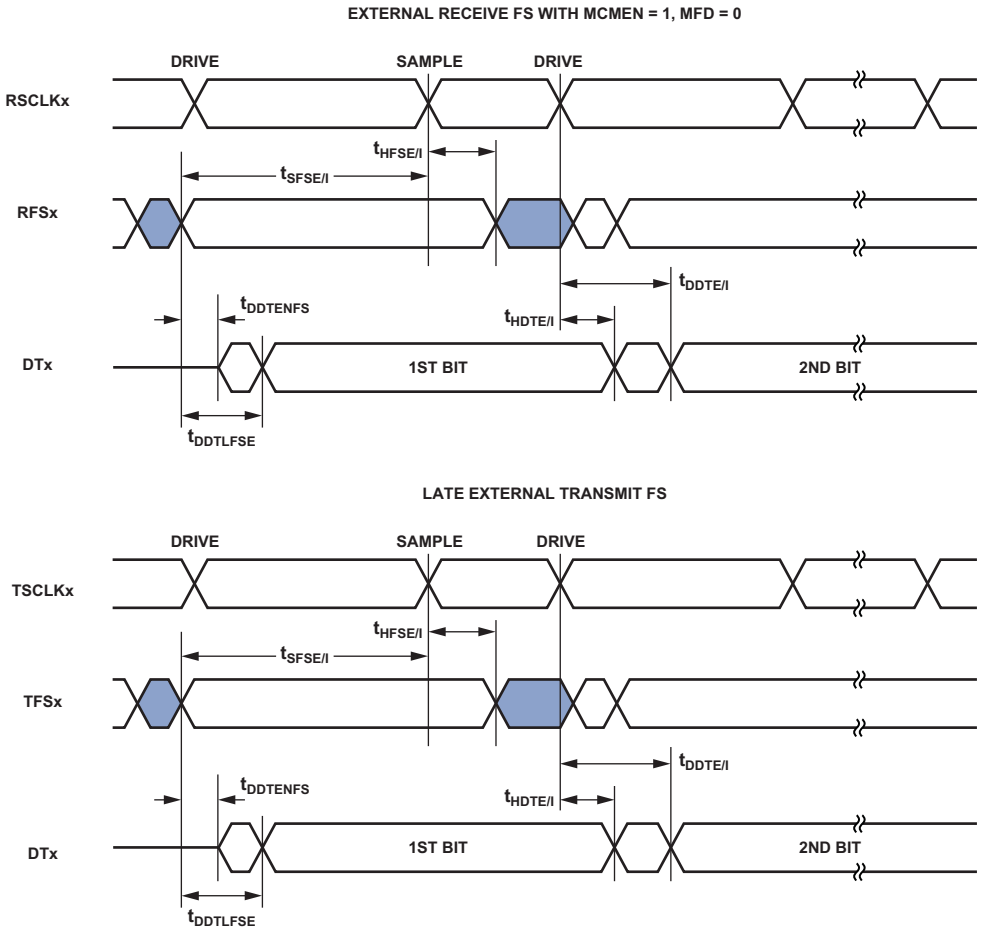

Figure 22. External Late Frame Sync

OUTPUT DRIVE CURRENTS

Figure 29 through Figure 36 on Page 42 show typical current voltage characteristics for the output drivers of the ADSP-BF561 processor. The curves represent the current drive capability of the output drivers as a function of output voltage. Refer to Table 8 on Page 17 to identify the driver type for a pin.

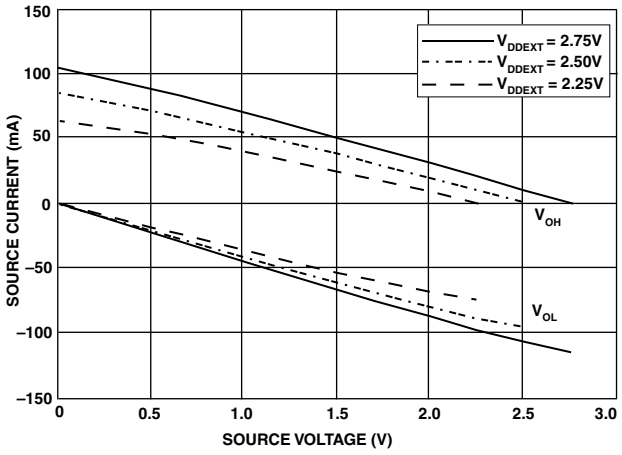

Figure 29. Drive Current A (Low V_{DDEXT})

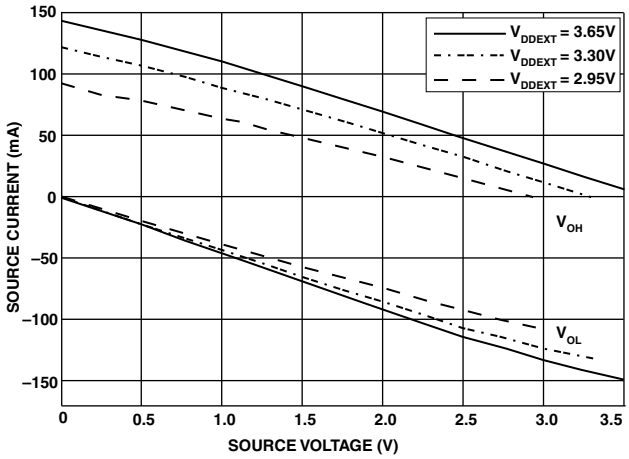

Figure 30. Drive Current A (High V_{DDEXT})

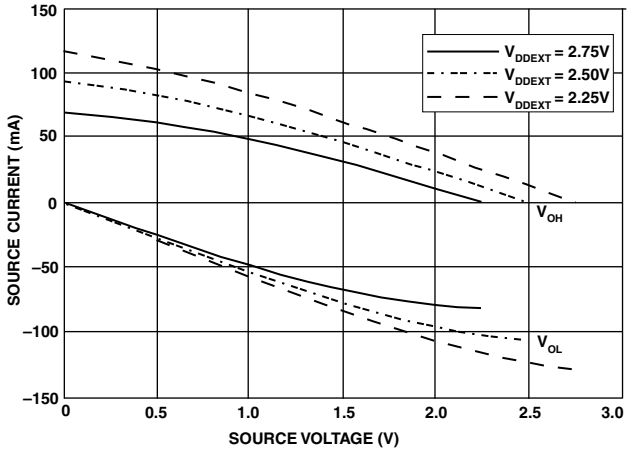

Figure 31. Drive Current B (Low V_{DDEXT})

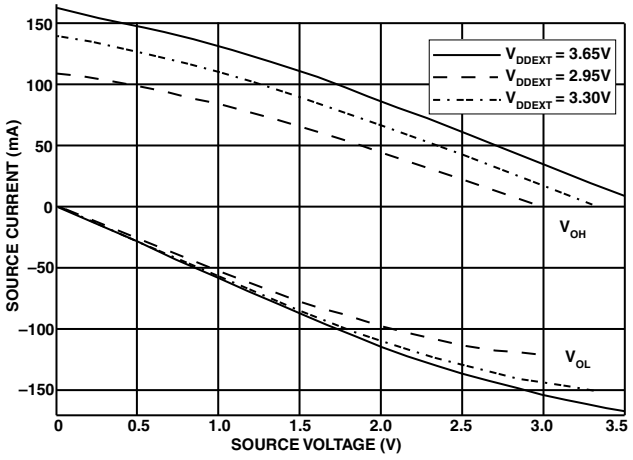

Figure 32. Drive Current B (High V_{DDEXT})

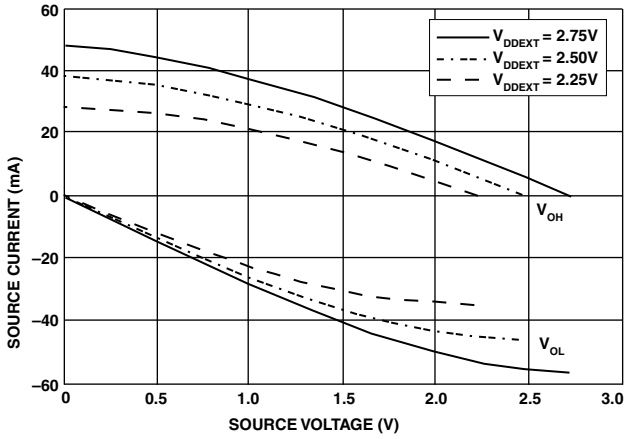

Figure 33. Drive Current C (Low V_{DDEXT})

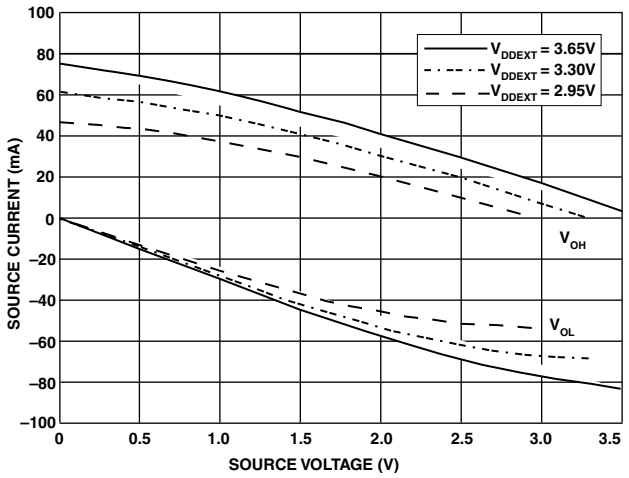

Figure 34. Drive Current C (High V_{DDEXT})

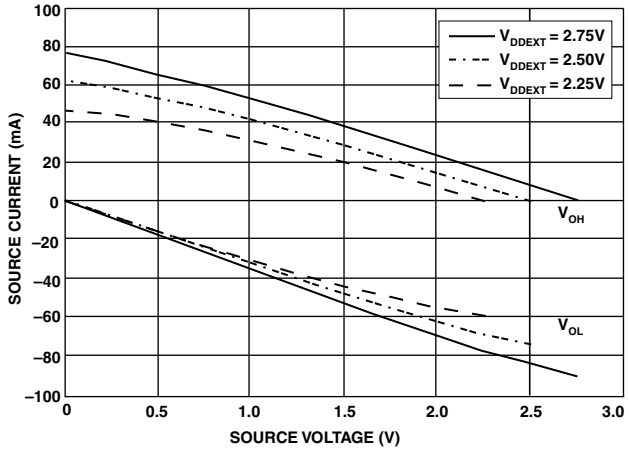

Figure 35. Drive Current D (Low V_{DDEXT})

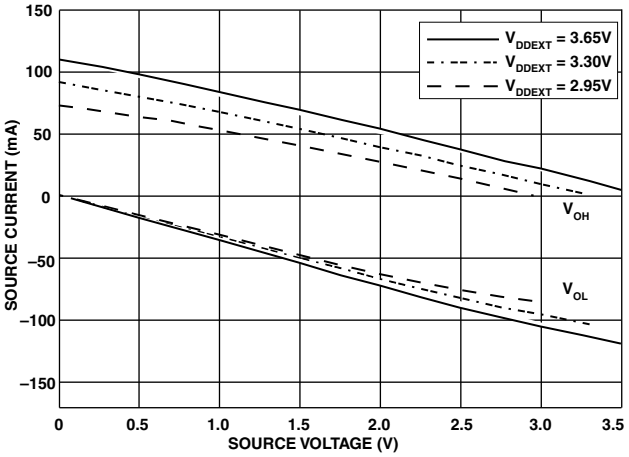

Figure 36. Drive Current D (High V_{DDEXT})

POWER DISSIPATION

Many operating conditions can affect power dissipation. System designers should refer to *Estimating Power for ADSP-BF561 Blackfin Processors (EE-293)* on the Analog Devices website (www.analog.com)—use site search on "EE-293." This document provides detailed information for optimizing your design for lowest power.

See the *ADSP-BF561 Blackfin Processor Hardware Reference Manual* for definitions of the various operating modes and for instructions on how to minimize system power.

TEST CONDITIONS

All timing parameters appearing in this data sheet were measured under the conditions described in this section. Figure 37 shows the measurement point for ac measurements (except output enable/disable). The measurement point V_{MEAS} is 1.5 V for V_{DDEXT} (nominal) = 2.5 V/3.3 V.

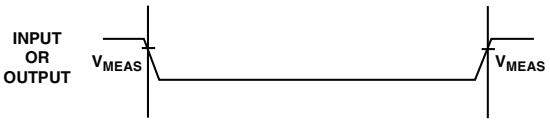

Figure 37. Voltage Reference Levels for AC Measurements (Except Output Enable/Disable)

Output Enable Time Measurement

Output pins are considered to be enabled when they have made a transition from a high impedance state to the point when they start driving.

The output enable time t_{ENA} is the interval from the point when a reference signal reaches a high or low voltage level to the point when the output starts driving as shown on the right side of Figure 38 on Page 43.

The time $t_{ENA\_MEASURED}$ is the interval, from when the reference signal switches, to when the output voltage reaches V_{TRIP}(high) or V_{TRIP}(low). V_{TRIP}(high) is 2.0 V and V_{TRIP}(low) is 1.0 V for V_{DDEXT} (nominal) = 2.5 V/3.3 V. Time t_{TRIP} is the interval from when the output starts driving to when the output reaches the V_{TRIP}(high) or V_{TRIP}(low) trip voltage.

Time t_{ENA} is calculated as shown in the equation:

$$t_{ENA} = t_{ENA\_MEASURED} - t_{TRIP}$$

If multiple pins (such as the data bus) are enabled, the measurement value is that of the first pin to start driving.

Output Disable Time Measurement

Output pins are considered to be disabled when they stop driving, go into a high impedance state, and start to decay from their output high or low voltage. The output disable time t_{DIS} is the difference between $t_{DIS\_MEASURED}$ and t_{DECAY} as shown on the left side of Figure 38 on Page 43.

$$t_{DIS} = t_{DIS\_MEASURED} - t_{DECAY}$$

The time for the voltage on the bus to decay by ΔV is dependent on the capacitive load C_L and the load current I_L. This decay time can be approximated by the equation:

$$t_{DECAY} = (C_L \Delta V)/I_L$$

The time t_{DECAY} is calculated with test loads C_L and I_L, and with ΔV equal to 0.5 V for V_{DDEXT} (nominal) = 2.5 V/3.3 V.

The time $t_{DIS\_MEASURED}$ is the interval from when the reference signal switches, to when the output voltage decays ΔV from the measured output high or output low voltage.

Example System Hold Time Calculation

To determine the data output hold time in a particular system, first calculate t_{DECAY} using the equation given above. Choose ΔV to be the difference between the ADSP-BF561 processor's output voltage and the input threshold for the device requiring the hold time. C_L is the total bus capacitance (per data line), and I_L is the total leakage or three-state current (per data line). The hold time will be t_{DECAY} plus the various output disable times as specified in the Timing Specifications on Page 23 (for example t_{DSDAT} for an SDRAM write cycle as shown in SDRAM Interface Timing on Page 26).

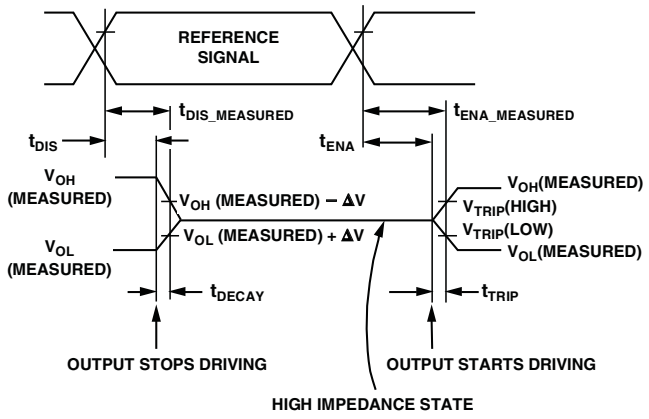

Figure 38. Output Enable/Disable

Capacitive Loading

Output delays and holds are based on standard capacitive loads: 30 pF on all pins (see Figure 39). V_{LOAD} is 1.5 V for V_{DDEXT} (nominal) = 2.5 V/3.3 V. Figure 40 through Figure 47 on Page 44 show how output rise time varies with capacitance. The delay and hold specifications given should be derated by a factor derived from these figures. The graphs in these figures may not be linear outside the ranges shown.

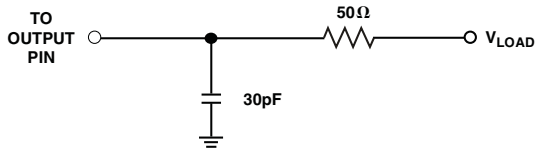

Figure 39. Equivalent Device Loading for AC Measurements (Includes All Fixtures)

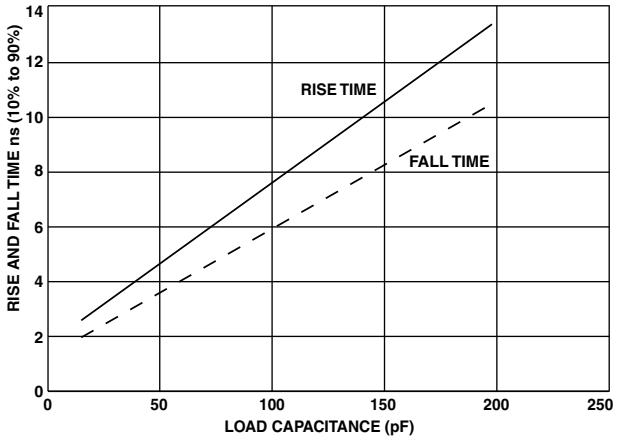

Figure 40. Typical Rise and Fall Times (10% to 90%) versus Load Capacitance for Driver A at V_{DDEXT} (min)

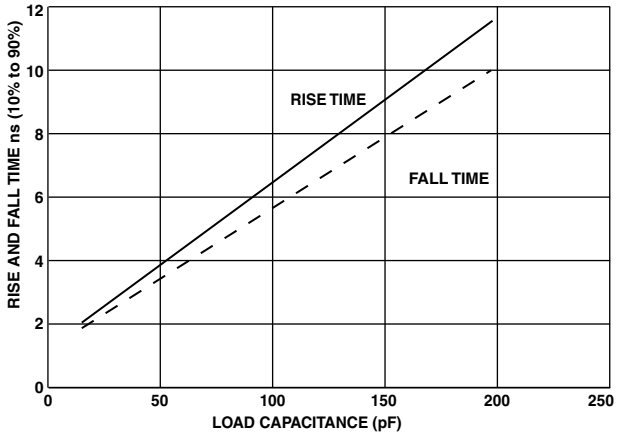

Figure 41. Typical Rise and Fall Times (10% to 90%) versus Load Capacitance for Driver A at V_{DDEXT} (max)

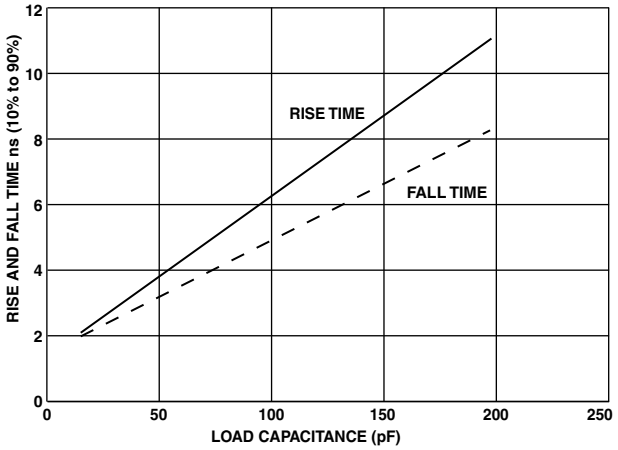

Figure 42. Typical Rise and Fall Times (10% to 90%) versus Load Capacitance for Driver B at V_{DDEXT} (min)

256-BALL CSP_BGA (17 mm) BALL ASSIGNMENT

Table 35 lists the 256-Ball CSP_BGA (17 mm × 17 mm) ball assignment by ball number. Table 36 on Page 48 lists the ball assignment alphabetically by signal.

Table 35. 256-Ball CSP_BGA (17 mm × 17 mm) Ball Assignment (Numerically by Ball Number)

Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal
A1	VDDEXT	C9	$\overline{\text{SMS3}}$	F1	CLKIN	H9	GND	L1	PPI0D3
A2	ADDR22	C10	$\overline{\text{SWE}}$	F2	PPI0D10	H10	GND	L2	PPI0D2
A3	ADDR18	C11	SA10	F3	$\overline{\text{RESET}}$	H11	GND	L3	PPI0D1
A4	ADDR14	C12	$\overline{\text{ABE0}}$	F4	BYPASS	H12	GND	L4	PPI0D0
A5	ADDR11	C13	ADDR07	F5	VDDEXT	H13	GND	L5	VDDEXT
A6	$\overline{\text{AMS3}}$	C14	ADDR04	F6	VDDEXT	H14	DATA21	L6	VDDEXT
A7	$\overline{\text{AMS0}}$	C15	DATA0	F7	VDDEXT	H15	DATA19	L7	VDDEXT
A8	ARDY	C16	DATA05	F8	GND	H16	DATA23	L8	VDDEXT
A9	$\overline{\text{SMS2}}$	D1	PPI0D15	F9	GND	J1	VROUT1	L9	GND
A10	SCLK0	D2	PPI0SYNC3	F10	VDDEXT	J2	PPI0D8	L10	VDDEXT
A11	SCLK1	D3	PPI0SYNC2	F11	VDDEXT	J3	PPI0D7	L11	VDDEXT
A12	$\overline{\text{ABE2}}$	D4	ADDR21	F12	VDDEXT	J4	PPI0D9	L12	VDDEXT
A13	$\overline{\text{ABE3}}$	D5	ADDR15	F13	DATA11	J5	GND	L13	NC
A14	ADDR06	D6	ADDR09	F14	DATA08	J6	GND	L14	DT0PRI
A15	ADDR03	D7	$\overline{\text{AWE}}$	F15	DATA10	J7	GND	L15	DATA31
A16	VDDEXT	D8	$\overline{\text{SMS0}}$	F16	DATA16	J8	GND	L16	DATA28
B1	ADDR24	D9	$\overline{\text{SRAS}}$	G1	XTAL	J9	GND	M1	PPI1SYNC2
B2	ADDR23	D10	$\overline{\text{SCAS}}$	G2	VDDEXT	J10	GND	M2	PPI1D15
B3	ADDR19	D11	$\overline{\text{BGH}}$	G3	VDDEXT	J11	GND	M3	PPI1D14
B4	ADDR17	D12	$\overline{\text{ABE1}}$	G4	GND	J12	VDDINT	M4	PPI1D9
B5	ADDR12	D13	DATA02	G5	GND	J13	VDDINT	M5	VDDINT
B6	ADDR10	D14	DATA01	G6	VDDEXT	J14	DATA20	M6	VDDINT
B7	$\overline{\text{AMS1}}$	D15	DATA03	G7	GND	J15	DATA22	M7	GND
B8	$\overline{\text{AOE}}$	D16	DATA07	G8	GND	J16	DATA24	M8	VDDINT
B9	$\overline{\text{SMS1}}$	E1	PPI0D11	G9	GND	K1	PPI0D6	M9	GND
B10	SCKE	E2	PPI0D13	G10	GND	K2	PPI0D5	M10	VDDINT
B11	$\overline{\text{BR}}$	E3	PPI0D12	G11	VDDEXT	K3	PPI0D4	M11	GND
B12	$\overline{\text{BG}}$	E4	PPI0D14	G12	VDDEXT	K4	PPI1SYNC3	M12	VDDINT
B13	ADDR08	E5	PPI1CLK	G13	DATA17	K5	VDDEXT	M13	RSCLK0
B14	ADDR05	E6	VDDINT	G14	DATA14	K6	VDDEXT	M14	DR0PRI
B15	ADDR02	E7	GND	G15	DATA15	K7	GND	M15	TSCLK0
B16	DATA04	E8	VDDINT	G16	DATA18	K8	GND	M16	DATA29
C1	PPI0SYNC1	E9	GND	H1	VROUT0	K9	GND	N1	PPI1SYNC1
C2	ADDR25	E10	VDDINT	H2	GND	K10	GND	N2	PPI1D10
C3	PPI0CLK	E11	GND	H3	GND	K11	VDDEXT	N3	PPI1D7
C4	ADDR20	E12	VDDINT	H4	VDDINT	K12	GND	N4	PPI1D5
C5	ADDR16	E13	DATA06	H5	VDDINT	K13	GND	N5	PF0
C6	ADDR13	E14	DATA13	H6	GND	K14	DATA26	N6	PF04
C7	$\overline{\text{AMS2}}$	E15	DATA09	H7	GND	K15	DATA25	N7	PF09
C8	$\overline{\text{ARE}}$	E16	DATA12	H8	GND	K16	DATA27	N8	PF12

256-BALL CSP_BGA (12 mm) BALL ASSIGNMENT

Table 37 lists the 256-Ball CSP_BGA (12 mm × 12 mm) ball assignment by ball number. Table 38 on Page 53 lists the ball assignment alphabetically by signal.

Table 37. 256-Ball CSP_BGA (12 mm × 12 mm) Ball Assignment (Numerically by Ball Number)

Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal
A01	VDDEXT	C09	$\overline{\text{SMS2}}$	F01	CLKIN	H09	GND	L01	PPI0D0
A02	ADDR24	C10	$\overline{\text{SRAS}}$	F02	VDDEXT	H10	GND	L02	PPI1SYNC2
A03	ADDR20	C11	GND	F03	$\overline{\text{RESET}}$	H11	VDDINT	L03	GND
A04	VDDEXT	C12	$\overline{\text{BGH}}$	F04	PPI0D10	H12	DATA16	L04	PPI1SYNC3
A05	ADDR14	C13	GND	F05	ADDR21	H13	DATA18	L05	VDDEXT
A06	ADDR10	C14	ADDR07	F06	ADDR17	H14	DATA20	L06	PPI1D11
A07	$\overline{\text{AMS3}}$	C15	DATA1	F07	VDDINT	H15	DATA17	L07	GND
A08	$\overline{\text{AWE}}$	C16	DATA3	F08	GND	H16	DATA19	L08	VDDINT
A09	VDDEXT	D01	PPI0D13	F09	VDDINT	J01	VROUT0	L09	GND
A10	$\overline{\text{SMS3}}$	D02	PPI0D15	F10	GND	J02	VROUT1	L10	VDDEXT
A11	SCLK0	D03	PPI0SYNC3	F11	ADDR08	J03	PPI0D2	L11	GND
A12	SCLK1	D04	ADDR23	F12	DATA10	J04	PPI0D3	L12	DR0PRI
A13	$\overline{\text{BG}}$	D05	GND	F13	DATA8	J05	PPI0D1	L13	TFS0
A14	$\overline{\text{ABE2}}$	D06	GND	F14	DATA12	J06	VDDEXT	L14	GND
A15	$\overline{\text{ABE3}}$	D07	ADDR09	F15	DATA9	J07	GND	L15	DATA27
A16	VDDEXT	D08	GND	F16	DATA11	J08	VDDINT	L16	DATA29
B01	PPI1CLK	D09	ARDY	G01	XTAL	J09	VDDINT	M01	PPI1D15
B02	ADDR22	D10	$\overline{\text{SCAS}}$	G02	GND	J10	VDDINT	M02	PPI1D13
B03	ADDR18	D11	SA10	G03	VDDEXT	J11	GND	M03	PPI1D9
B04	ADDR16	D12	VDDEXT	G04	BYPASS	J12	DATA30	M04	GND
B05	ADDR12	D13	ADDR02	G05	PPI0D14	J13	DATA22	M05	NC
B06	VDDEXT	D14	GND	G06	GND	J14	GND	M06	PF3
B07	$\overline{\text{AMS1}}$	D15	DATA5	G07	GND	J15	DATA21	M07	PF7
B08	$\overline{\text{ARE}}$	D16	DATA6	G08	GND	J16	DATA23	M08	VDDINT
B09	$\overline{\text{SMS1}}$	E01	GND	G09	VDDINT	K01	PPI0D6	M09	GND
B10	SCKE	E02	PPI0D11	G10	ADDR05	K02	PPI0D4	M10	BMODE0
B11	VDDEXT	E03	PPI0D12	G11	ADDR03	K03	PPI0D8	M11	SCK
B12	$\overline{\text{BR}}$	E04	PPI0SYNC1	G12	DATA15	K04	PPI1SYNC1	M12	DR1PRI
B13	$\overline{\text{ABE1}}$	E05	ADDR15	G13	DATA14	K05	PPI1D14	M13	NC
B14	ADDR06	E06	ADDR13	G14	GND	K06	VDDEXT	M14	VDDEXT
B15	ADDR04	E07	$\overline{\text{AMS2}}$	G15	DATA13	K07	GND	M15	DATA31
B16	DATA0	E08	VDDINT	G16	VDDEXT	K08	VDDINT	M16	DT0PRI
C01	PPI0SYNC2	E09	$\overline{\text{SMS0}}$	H01	GND	K09	GND	N01	PPI1D12
C02	PPI0CLK	E10	$\overline{\text{SWE}}$	H02	GND	K10	GND	N02	PPI1D10
C03	ADDR25	E11	$\overline{\text{ABE0}}$	H03	PPI0D9	K11	VDDINT	N03	PPI1D3
C04	ADDR19	E12	DATA2	H04	PPI0D7	K12	DATA28	N04	PPI1D1
C05	GND	E13	GND	H05	PPI0D5	K13	DATA26	N05	PF1
C06	ADDR11	E14	DATA4	H06	VDDINT	K14	DATA24	N06	PF9
C07	$\overline{\text{AOE}}$	E15	DATA7	H07	VDDINT	K15	DATA25	N07	GND
C08	$\overline{\text{AMS0}}$	E16	VDDEXT	H08	GND	K16	VDDEXT	N08	PF13

297-BALL PBGA BALL ASSIGNMENT

Table 39 lists the 297-Ball PBGA ball assignment numerically by ball number. Table 40 on Page 58 lists the ball assignment alphabetically by signal.

Table 39. 297-Ball PBGA Ball Assignment (Numerically by Ball Number)

Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal
A01	GND	B15	$\overline{\text{SMS1}}$	G01	PPI0D11	L14	GND
A02	ADDR25	B16	$\overline{\text{SMS3}}$	G02	PPI0D10	L15	GND
A03	ADDR23	B17	SCKE	G25	DATA4	L16	GND
A04	ADDR21	B18	$\overline{\text{SWE}}$	G26	DATA7	L17	GND
A05	ADDR19	B19	SA10	H01	BYPASS	L18	VDDINT
A06	ADDR17	B20	$\overline{\text{BR}}$	H02	$\overline{\text{RESET}}$	L25	DATA12
A07	ADDR15	B21	$\overline{\text{BG}}$	H25	DATA6	L26	DATA15
A08	ADDR13	B22	$\overline{\text{ABE1}}$	H26	DATA9	M01	VROUT0
A09	ADDR11	B23	$\overline{\text{ABE3}}$	J01	CLKIN	M02	GND
A10	ADDR09	B24	ADDR07	J02	GND	M10	VDDEXT
A11	$\overline{\text{AMS3}}$	B25	GND	J10	VDDEXT	M11	GND
A12	$\overline{\text{AMS1}}$	B26	ADDR05	J11	VDDEXT	M12	GND
A13	$\overline{\text{AWE}}$	C01	PPI0SYNC3	J12	VDDEXT	M13	GND
A14	$\overline{\text{ARE}}$	C02	PPI0CLK	J13	VDDEXT	M14	GND
A15	$\overline{\text{SMS0}}$	C03	GND	J14	VDDEXT	M15	GND
A16	$\overline{\text{SMS2}}$	C04	GND	J15	VDDEXT	M16	GND
A17	$\overline{\text{SRAS}}$	C05	GND	J16	VDDINT	M17	GND
A18	$\overline{\text{SCAS}}$	C22	GND	J17	VDDINT	M18	VDDINT
A19	SCLK0	C23	GND	J18	VDDINT	M25	DATA14
A20	SCLK1	C24	GND	J25	DATA8	M26	DATA17
A21	$\overline{\text{BGH}}$	C25	ADDR04	J26	DATA11	N01	VROUT1
A22	$\overline{\text{ABE0}}$	C26	ADDR03	K01	XTAL	N02	PPI0D9
A23	$\overline{\text{ABE2}}$	D01	PPI0SYNC1	K02	NC	N10	VDDEXT
A24	ADDR08	D02	PPI0SYNC2	K10	VDDEXT	N11	GND
A25	ADDR06	D03	GND	K11	VDDEXT	N12	GND
A26	GND	D04	GND	K12	VDDEXT	N13	GND
B01	PPI1CLK	D23	GND	K13	VDDEXT	N14	GND
B02	GND	D24	GND	K14	VDDEXT	N15	GND
B03	ADDR24	D25	ADDR02	K15	VDDEXT	N16	GND
B04	ADDR22	D26	DATA1	K16	VDDINT	N17	GND
B05	ADDR20	E01	PPI0D15	K17	VDDINT	N18	VDDINT
B06	ADDR18	E02	PPI0D14	K18	VDDINT	N25	DATA16
B07	ADDR16	E03	GND	K25	DATA10	N26	DATA19
B08	ADDR14	E24	GND	K26	DATA13	P01	PPI0D7
B09	ADDR12	E25	DATA0	L01	NC	P02	PPI0D8
B10	ADDR10	E26	DATA3	L02	NC	P10	VDDEXT
B11	$\overline{\text{AMS2}}$	F01	PPI0D13	L10	VDDEXT	P11	GND
B12	$\overline{\text{AMS0}}$	F02	PPI0D12	L11	GND	P12	GND
B13	$\overline{\text{AOE}}$	F25	DATA2	L12	GND	P13	GND
B14	ARDY	F26	DATA5	L13	GND	P14	GND

Table 39. 297-Ball PBGA Ball Assignment (Numerically by Ball Number) (Continued)

Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal
P15	GND	U11	VDDEXT	AC04	GND	AE21	RX
P16	GND	U12	VDDEXT	AC23	GND	AE22	RFS1
P17	GND	U13	VDDEXT	AC24	GND	AE23	DR1SEC
P18	VDDINT	U14	GND	AC25	DR0SEC	AE24	TFS1
P25	DATA18	U15	VDDINT	AC26	RFS0	AE25	GND
P26	DATA21	U16	VDDINT	AD01	PPI1D7	AE26	NC
R01	PPI0D5	U17	VDDINT	AD02	PPI1D6	AF01	GND
R02	PPI0D6	U18	VDDINT	AD03	GND	AF02	PPI1D4
R10	VDDEXT	U25	DATA24	AD04	GND	AF03	PPI1D2
R11	GND	U26	DATA27	AD05	GND	AF04	PPI1D0
R12	GND	V01	PPI1SYNC3	AD22	GND	AF05	PF1
R13	GND	V02	PPI0D0	AD23	GND	AF06	PF3
R14	GND	V25	DATA26	AD24	GND	AF07	PF5
R15	GND	V26	DATA29	AD25	NC	AF08	PF7
R16	GND	W01	PPI1SYNC1	AD26	RSCLK0	AF09	PF9
R17	GND	W02	PPI1SYNC2	AE01	PPI1D5	AF10	PF11
R18	VDDINT	W25	DATA28	AE02	GND	AF11	PF13
R25	DATA20	W26	DATA31	AE03	PPI1D3	AF12	PF15
R26	DATA23	Y01	PPI1D15	AE04	PPI1D1	AF13	NMI1
T01	PPI0D3	Y02	PPI1D14	AE05	PF0	AF14	TCK
T02	PPI0D4	Y25	DATA30	AE06	PF2	AF15	TDI
T10	VDDEXT	Y26	DT0PRI	AE07	PF4	AF16	TMS
T11	GND	AA01	PPI1D13	AE08	PF6	AF17	SLEEP
T12	GND	AA02	PPI1D12	AE09	PF8	AF18	NMI0
T13	GND	AA25	DT0SEC	AE10	PF10	AF19	SCK
T14	GND	AA26	TSCLK0	AE11	PF12	AF20	TX
T15	GND	AB01	PPI1D11	AE12	PF14	AF21	RSCLK1
T16	GND	AB02	PPI1D10	AE13	NC	AF22	DR1PRI
T17	GND	AB03	GND	AE14	TDO	AF23	TSCLK1
T18	VDDINT	AB24	GND	AE15	$\overline{\text{TRST}}$	AF24	DT1SEC
T25	DATA22	AB25	TFS0	AE16	$\overline{\text{EMU}}$	AF25	DT1PRI
T26	DATA25	AB26	DR0PRI	AE17	BMODE1	AF26	GND
U01	PPI0D1	AC01	PPI1D9	AE18	BMODE0		
U02	PPI0D2	AC02	PPI1D8	AE19	MISO		
U10	VDDEXT	AC03	GND	AE20	MOSI		

Figure 52 lists the top view of the 297-Ball PBGA ball configuration. Figure 53 lists the bottom view.

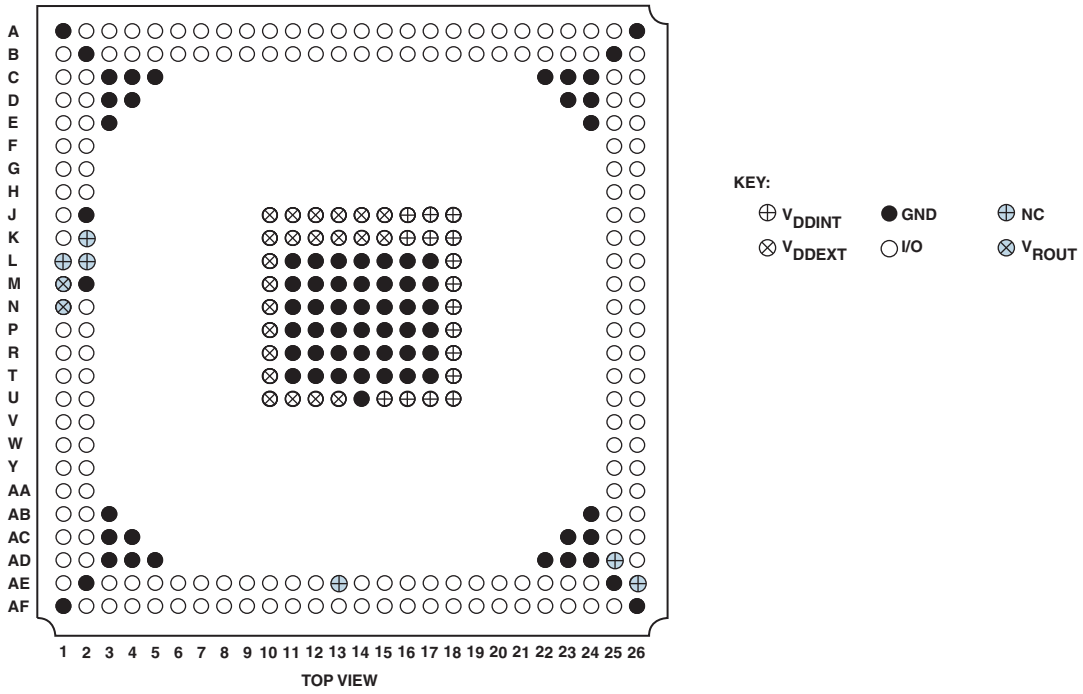

Figure 52. 297-Ball PBGA Ball Configuration (Top View)

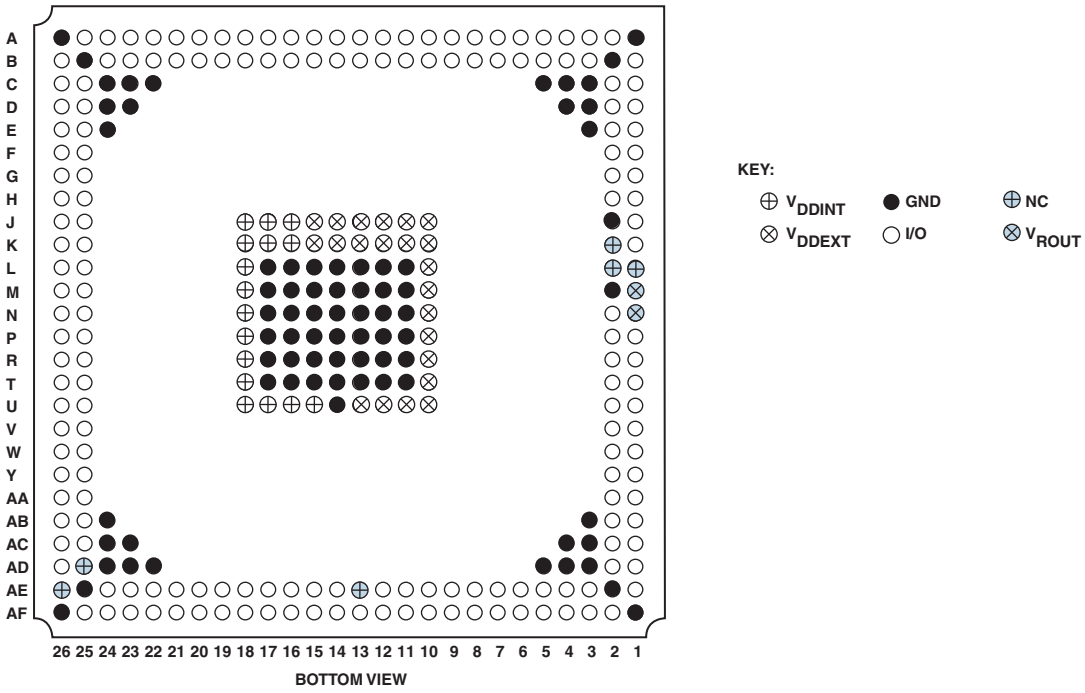

Figure 53. 297-Ball PBGA Ball Configuration (Bottom View)

SURFACE-MOUNT DESIGN

Table 41 is provided as an aid to PCB design. For industry-standard design recommendations, refer to IPC-7351, *Generic Requirements for Surface Mount Design and Land Pattern Standard.*

Table 41. BGA Data for Use with Surface-Mount Design

Package	Ball Attach Type	Solder Mask Opening	Ball Pad Size
256-Ball CSP_BGA (BC-256-1)	Solder Mask Defined	0.30 mm diameter	0.43 mm diameter
256-Ball CSP_BGA (BC-256-4)	Solder Mask Defined	0.43 mm diameter	0.55 mm diameter
297-Ball PBGA (B-297)	Solder Mask Defined	0.43 mm diameter	0.58 mm diameter

AUTOMOTIVE PRODUCTS

Some ADSP-BF561 models are available for automotive applications with controlled manufacturing. Note that these special models may have specifications that differ from the general release models.

The automotive grade products shown in Table 42 are available for use in automotive applications. Contact your local ADI account representative or authorized ADI product distributor for specific product ordering information. Note that all automotive products are RoHS compliant.

Table 42. Automotive Products

Product Family[1]	Temperature Range[2]	Speed Grade (Max)[3]	Package Description	Package Option
ADBF561WBBZ5xx	–40°C to +85°C	533 MHz	297-Ball PBGA	B-297
ADBF561WBBCZ5xx	–40°C to +85°C	533 MHz	256-Ball CSP_BGA	BC-256-4

[1] xx denotes silicon revision.
[2] Referenced temperature is ambient temperature.
[3] The internal voltage regulation feature is not available. External voltage regulation is required to ensure correct operation.

ORDERING GUIDE

Model	Temperature Range[1]	Speed Grade (Max)	Package Description	Package Option
ADSP-BF561SKBCZ-6V[2]	0°C to +70°C	600 MHz	256-Ball CSP_BGA	BC-256-1
ADSP-BF561SKBCZ-5V[2]	0°C to +70°C	533 MHz	256-Ball CSP_BGA	BC-256-1
ADSP-BF561SKBCZ500[2]	0°C to +70°C	500 MHz	256-Ball CSP_BGA	BC-256-1
ADSP-BF561SKB500	0°C to +70°C	500 MHz	297-Ball PBGA	B-297
ADSP-BF561SKB600	0°C to +70°C	600 MHz	297-Ball PBGA	B-297
ADSP-BF561SKBZ500[2]	0°C to +70°C	500 MHz	297-Ball PBGA	B-297
ADSP-BF561SKBZ600[2]	0°C to +70°C	600 MHz	297-Ball PBGA	B-297
ADSP-BF561SBB600	–40°C to +85°C	600 MHz	297-Ball PBGA	B-297
ADSP-BF561SBB500	–40°C to +85°C	500 MHz	297-Ball PBGA	B-297
ADSP-BF561SBBZ600[2]	–40°C to +85°C	600 MHz	297-Ball PBGA	B-297
ADSP-BF561SBBZ500[2]	–40°C to +85°C	500 MHz	297-Ball PBGA	B-297
ADSP-BF561SKBCZ-6A[2]	0°C to +70°C	600 MHz	256-Ball CSP_BGA	BC-256-4
ADSP-BF561SKBCZ-5A[2]	0°C to +70°C	500 MHz	256-Ball CSP_BGA	BC-256-4
ADSP-BF561SBBCZ-5A[2]	–40°C to +85°C	500 MHz	256-Ball CSP_BGA	BC-256-4

[1] Referenced temperature is ambient temperature.
[2] Z = RoHS compliant part.