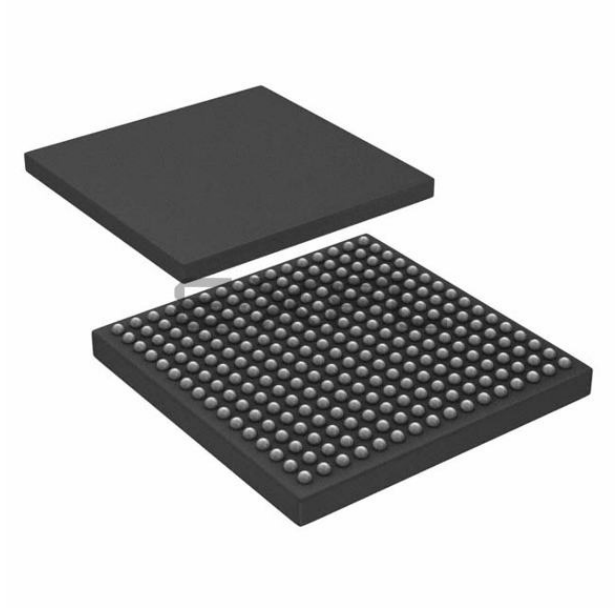

Welcome to E-XFL.COM

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	Fixed Point
Interface	SPI, SSP, UART
Clock Rate	533MHz
Non-Volatile Memory	External
On-Chip RAM	328kB
Voltage - I/O	2.50V, 3.30V
Voltage - Core	1.25V
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	256-BGA, CSPBGA
Supplier Device Package	256-CSPBGA (17x17)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/adsp-bf561sbbcz-5a

- ADSP-BF561: Blackfin Embedded Symmetric Multiprocessor Data Sheet

Emulator Manuals

- HPUSB, USB, and HPPCI Emulator User's Guide
- ICE-1000/ICE-2000 Emulator User's Guide
- ICE-100B Emulator User's Guide

Evaluation Kit Manuals

- ADSP-BF561 EZ-KIT Lite® Evaluation System Manual
- Blackfin® A-V EZ-Extender® Manual
- Blackfin® EZ-Extender® Manual
- Blackfin® FPGA EZ-Extender® Manual
- Blackfin®/SHARC® USB EZ-Extender® Manual

Integrated Circuit Anomalies

- ADSP-BF561 Blackfin Anomaly List for Revisions 0.3, 0.5

Processor Manuals

- ADSP-BF561 Blackfin ® Processor Hardware Reference
- ADSP-BF5xx/ADSP-BF60x Blackfin® Processor Programming Reference
- Blackfin Processors: Manuals

Product Highlight

- ADSP-BF561 Blackfin Dual-Core Embedded Processor
- Blackfin Processor Family Product Highlight
- EZ-KIT Lite for Analog Devices ADSP-BF561 Blackfin Processor

Software Manuals

- CrossCore® Embedded Studio 2.5.0 Assembler and Preprocessor Manual
- CrossCore® Embedded Studio 2.5.0 C/C++ Compiler and Library Manual for Blackfin Processors
- CrossCore® Embedded Studio 2.5.0 Linker and Utilities Manual
- CrossCore® Embedded Studio 2.5.0 Loader and Utilities Manual
- CrossCore® Software Licensing Guide
- lwIP for CrossCore® Embedded Studio 1.0.0 User's Guide
- VisualDSP++® 5.0 Assembler and Preprocessor Manual
- VisualDSP++® 5.0 C/C++ Compiler and Library Manual for Blackfin Processors
- VisualDSP++® 5.0 Device Drivers and System Services Manual for Blackfin Processors
- VisualDSP++® 5.0 Kernel (VDK) Users Guide
- VisualDSP++® 5.0 Licensing Guide
- VisualDSP++® 5.0 Linker and Utilities Manual
- VisualDSP++® 5.0 Loader and Utilities Manual
- VisualDSP++® 5.0 Product Release Bulletin
- VisualDSP++® 5.0 Quick Installation Reference Card
- VisualDSP++® 5.0 Users Guide

SOFTWARE AND SYSTEMS REQUIREMENTS

- Software and Tools Anomalies Search

TOOLS AND SIMULATIONS

- ADSP-BF561 Blackfin Processor BSDL File 256-Ball CSP BGA Package
- ADSP-BF561 Blackfin Processor BSDL File 256-Ball Sparse CSP_BGA Package
- ADSP-BF561 Blackfin Processor BSDL File 297-Ball PBGA Package
- ADSP-BF561 Blackfin Processor Core B BSDL File All Packages
- Designing with BGA
- Blackfin Processors Software and Tools
- ADSP-BF561 Blackfin Processor IBIS Datafile for 12x12 CSP BGA Package (02/2008)
- ADSP-BF561 Blackfin Processor IBIS Datafile for 17x17 CSP BGA Package (11/2008)
- ADSP-BF561 Blackfin Processor IBIS Datafile for 27x27 PBGA Package (02/2008)

TABLE OF CONTENTS

REVISION HISTORY

9/09—Rev. D to Rev. E

Correct all outstanding document errata.

Revised Figure 5 13

Added 533 MHz operation Table 10 20

Removed reference to 1.8 V operation Table 12 21

Added Table 17 and Figure 9 Power-Up Reset Timing 23

Removed references to T_J from t_{SCLK} parameter
Table 20 26

Added new SPORT timing parameters and diagram
Table 23 32
Figure 21 33

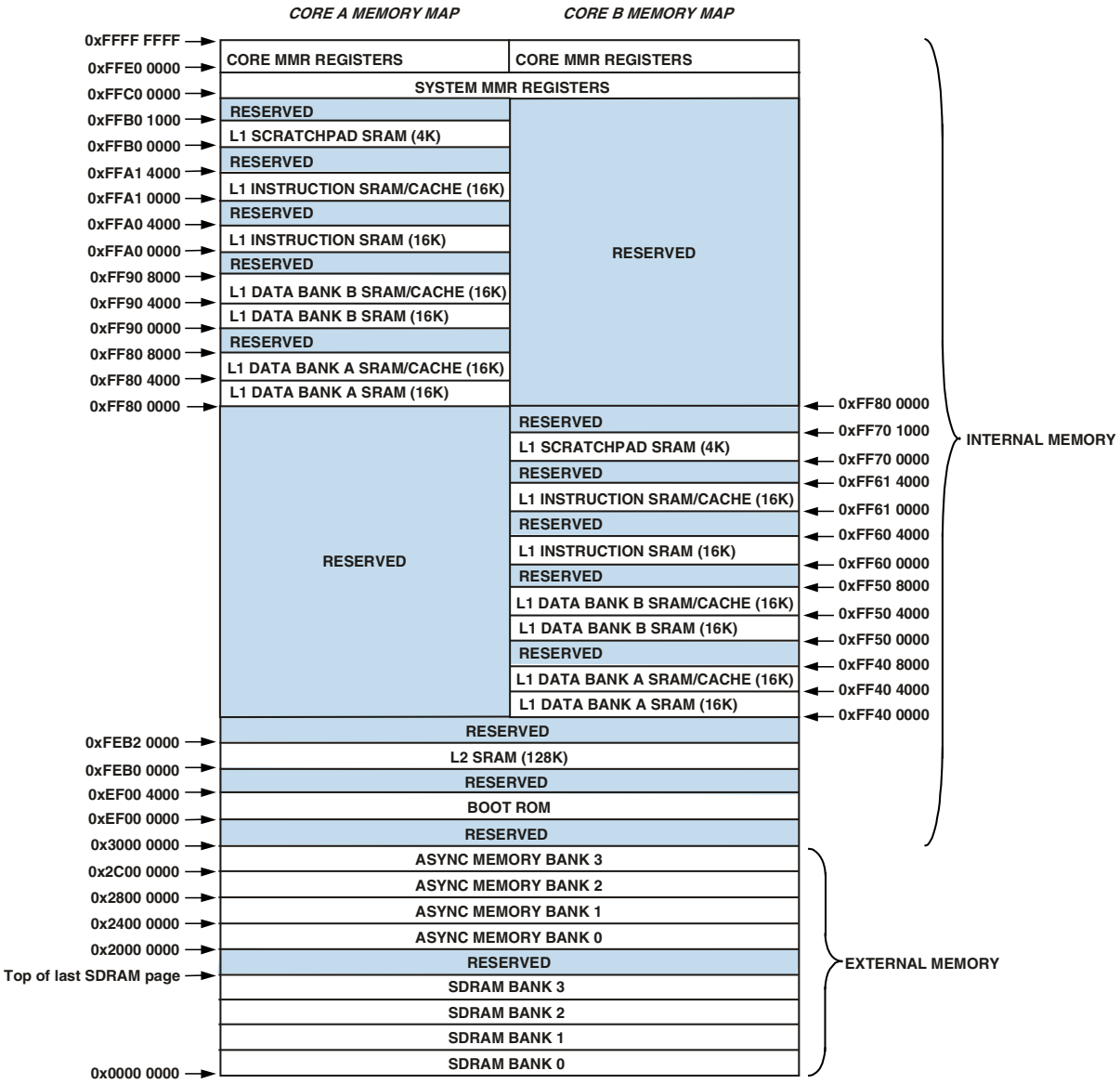

Figure 3. Memory Map

The fourth on-chip memory system is the L2 SRAM memory array which provides 128K bytes of high speed SRAM operating at one half the frequency of the core, and slightly longer latency than the L1 memory banks. The L2 memory is a unified instruction and data memory and can hold any mixture of code and data required by the system design. The Blackfin cores share a dedicated low latency 64-bit wide data path port into the L2 SRAM memory.

Each Blackfin core processor has its own set of core Memory Mapped Registers (MMRs) but share the same system MMR registers and 128K bytes L2 SRAM memory.

External (Off-Chip) Memory

The ADSP-BF561 external memory is accessed via the External Bus Interface Unit (EBIU). This interface provides a glueless connection to up to four banks of synchronous DRAM (SDRAM) as well as up to four banks of asynchronous memory devices, including flash, EPROM, ROM, SRAM, and memory mapped I/O devices.

The PC133-compliant SDRAM controller can be programmed to interface to up to four banks of SDRAM, with each bank containing between 16M bytes and 128M bytes providing access to up to 512M bytes of SDRAM. Each bank is independently programmable and is contiguous with adjacent banks regardless of the sizes of the different banks or their placement. This allows

writing the appropriate values into the Interrupt Assignment Registers (SIC_IAR7–0). Table 2 describes the inputs into the SIC and the default mappings into the CEC.

Table 2. System Interrupt Controller (SIC)

Peripheral Interrupt Event	Default Mapping
PLL Wakeup	IVG7
DMA1 Error (Generic)	IVG7
DMA2 Error (Generic)	IVG7
IMDMA Error	IVG7
PPI0 Error	IVG7
PPI1 Error	IVG7
SPORT0 Error	IVG7
SPORT1 Error	IVG7
SPI Error	IVG7
UART Error	IVG7
Reserved	IVG7
DMA1 Channel 0 Interrupt (PPI0)	IVG8
DMA1 Channel 1 Interrupt (PPI1)	IVG8
DMA1 Channel 2 Interrupt	IVG8
DMA1 Channel 3 Interrupt	IVG8
DMA1 Channel 4 Interrupt	IVG8
DMA1 Channel 5 Interrupt	IVG8
DMA1 Channel 6 Interrupt	IVG8
DMA1 Channel 7 Interrupt	IVG8
DMA1 Channel 8 Interrupt	IVG8
DMA1 Channel 9 Interrupt	IVG8
DMA1 Channel 10 Interrupt	IVG8
DMA1 Channel 11 Interrupt	IVG8
DMA2 Channel 0 Interrupt (SPORT0 Rx)	IVG9
DMA2 Channel 1 Interrupt (SPORT0 Tx)	IVG9
DMA2 Channel 2 Interrupt (SPORT1 Rx)	IVG9
DMA2 Channel 3 Interrupt (SPORT1 Tx)	IVG9
DMA2 Channel 4 Interrupt (SPI)	IVG9
DMA2 Channel 5 Interrupt (UART Rx)	IVG9
DMA2 Channel 6 Interrupt (UART Tx)	IVG9
DMA2 Channel 7 Interrupt	IVG9
DMA2 Channel 8 Interrupt	IVG9
DMA2 Channel 9 Interrupt	IVG9
DMA2 Channel 10 Interrupt	IVG9
DMA2 Channel 11 Interrupt	IVG9
Timer0 Interrupt	IVG10
Timer1 Interrupt	IVG10
Timer2 Interrupt	IVG10
Timer3 Interrupt	IVG10
Timer4 Interrupt	IVG10
Timer5 Interrupt	IVG10
Timer6 Interrupt	IVG10
Timer7 Interrupt	IVG10
Timer8 Interrupt	IVG10
Timer9 Interrupt	IVG10
Timer10 Interrupt	IVG10
Timer11 Interrupt	IVG10
Programmable Flags 15–0 Interrupt A	IVG11
Programmable Flags 15–0 Interrupt B	IVG11
Programmable Flags 31–16 Interrupt A	IVG11
Programmable Flags 31–16 Interrupt B	IVG11
Programmable Flags 47–32 Interrupt A	IVG11
Programmable Flags 47–32 Interrupt B	IVG11
DMA1 Channel 12/13 Interrupt (Memory DMA/Stream 0)	IVG8
DMA1 Channel 14/15 Interrupt (Memory DMA/Stream 1)	IVG8
DMA2 Channel 12/13 Interrupt (Memory DMA/Stream 0)	IVG9
DMA2 Channel 14/15 Interrupt (Memory DMA/Stream 1)	IVG9
IMDMA Stream 0 Interrupt	IVG12
IMDMA Stream 1 Interrupt	IVG12
Watchdog Timer Interrupt	IVG13
Reserved	IVG7
Reserved	IVG7
Supplemental Interrupt 0	IVG7
Supplemental Interrupt 1	IVG7

Event Control

The ADSP-BF561 provides the user with a very flexible mechanism to control the processing of events. In the CEC, three registers are used to coordinate and control events. Each of the registers is 16 bits wide, while each bit represents a particular event class.

- CEC Interrupt Latch Register (ILAT) – The ILAT register indicates when events have been latched. The appropriate bit is set when the processor has latched the event and cleared when the event has been accepted into the system. This register is updated automatically by the controller, but may also be written to clear (cancel) latched events. This register may be read while in supervisor mode and may only be written while in supervisor mode when the corresponding IMASK bit is cleared.
- CEC Interrupt Mask Register (IMASK) – The IMASK register controls the masking and unmasking of individual events. When a bit is set in the IMASK register, that event is unmasked and will be processed by the CEC when asserted. A cleared bit in the IMASK register masks the event, thereby preventing the processor from servicing the event

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.

In addition to the software and hardware development tools available from Analog Devices, third parties provide a wide range of tools supporting the Blackfin processor family. Third party software tools include DSP libraries, real-time operating systems, and block diagram design tools.

EZ-KIT Lite Evaluation Board

For evaluation of ADSP-BF561 processors, use the ADSP-BF561 EZ-KIT Lite® board available from Analog Devices. Order part number ADDS-BF561-EZLITE. The board comes with on-chip emulation capabilities and is equipped to enable software development. Multiple daughter cards are available.

DESIGNING AN EMULATOR-COMPATIBLE PROCESSOR BOARD

The Analog Devices family of emulators are tools that every system developer needs to test and debug hardware and software systems. Analog Devices has supplied an IEEE 1149.1 JTAG Test Access Port (TAP) on the ADSP-BF561. The emulator uses the TAP to access the internal features of the processor, allowing the developer to load code, set breakpoints, observe variables, observe memory, and examine registers. The processor must be halted to send data and commands, but once an operation has been completed by the emulator, the processor is set running at full speed with no impact on system timing.

To use these emulators, the target board must include a header that connects the processor's JTAG port to the emulator.

For details on target board design issues, including mechanical layout, single processor connections, multiprocessor scan chains, signal buffering, signal termination, and emulator pod logic, see *Analog Devices JTAG Emulation Technical Reference (EE-68)* on the Analog Devices website (www.analog.com)—use site search on "EE-68." This document is updated regularly to keep pace with improvements to emulator support.

RELATED DOCUMENTS

The following publications that describe the ADSP-BF561 processors (and related processors) can be ordered from any Analog Devices sales office or accessed electronically on our website:

- *Getting Started With Blackfin Processors*
- *ADSP-BF561 Blackfin Processor Hardware Reference*
- *ADSP-BF53x/BF56x Blackfin Processor Programming Reference*
- ADSP-BF561 *Blackfin Processor Anomaly List*

Asynchronous Memory Write Cycle Timing

Table 19. Asynchronous Memory Write Cycle Timing

Parameter		Min	Max	Unit
Timing Requirements				
t_{SARDY}	ARDY Setup Before CLKOUT	4.0		ns
t_{HARDY}	ARDY Hold After CLKOUT	0.0		ns
Switching Characteristics				
t_{DDAT}	DATA31–0 Disable After CLKOUT		6.0	ns
t_{ENDAT}	DATA31–0 Enable After CLKOUT	1.0		ns
t_{DO}	Output Delay After CLKOUT[1]		6.0	ns
t_{HO}	Output Hold After CLKOUT [1]	0.8		ns

[1] Output pins include $\overline{AMS3-0}$, $\overline{ABE3-0}$, ADDR25–2, DATA31–0, $\overline{AOE}$, $\overline{AWE}$.

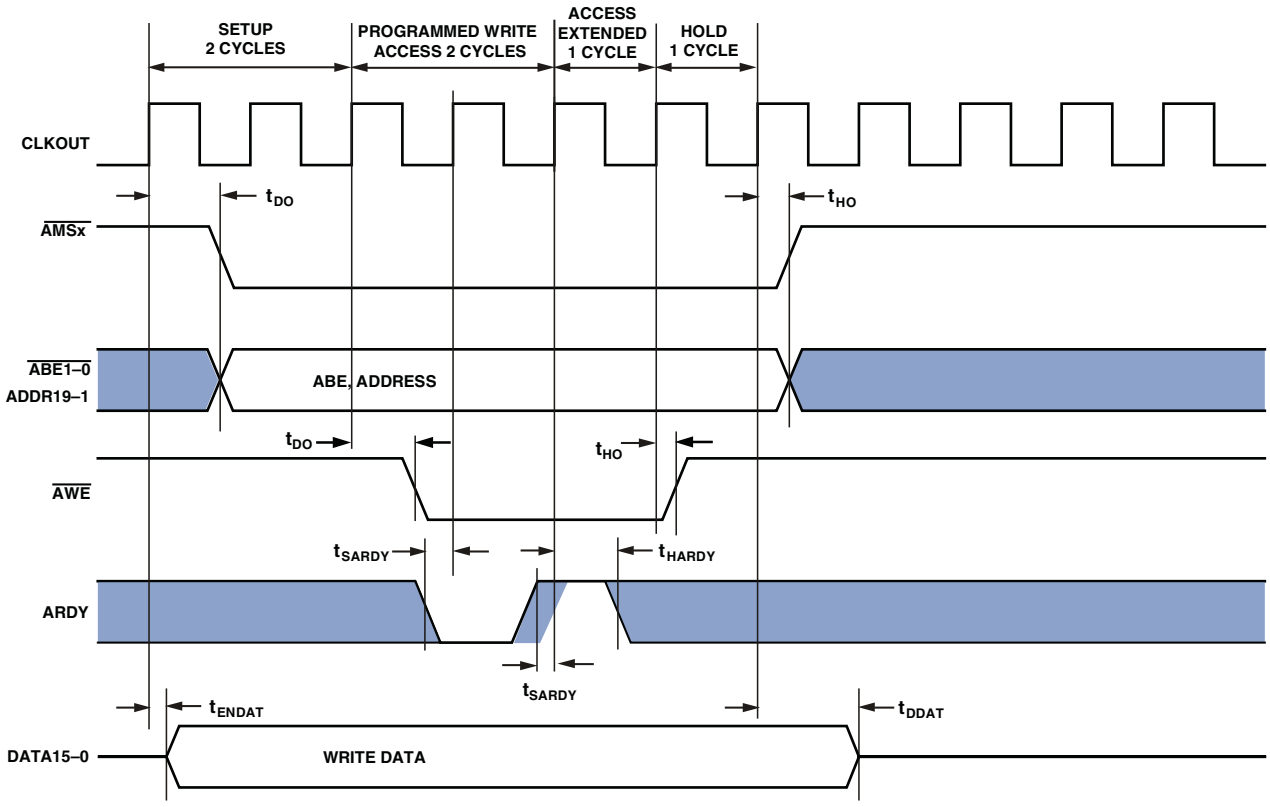

Figure 11. Asynchronous Memory Write Cycle Timing

SDRAM Interface Timing

Table 20. SDRAM Interface Timing

Parameter		Min	Max	Unit
Timing Requirements				
t_{SSDAT}	DATA Setup Before CLKOUT	1.5		ns
t_{HSDAT}	DATA Hold After CLKOUT	0.8		ns
Switching Characteristics				
t_{DCAD}	Command, ADDR, Data Delay After CLKOUT[1]		4.0	ns
t_{HCAD}	Command, ADDR, Data Hold After CLKOUT[1]	0.8		ns
t_{DSDAT}	Data Disable After CLKOUT		4.0	ns
t_{ENSDAT}	Data Enable After CLKOUT	1.0		ns
t_{SCLK}	CLKOUT Period	7.5		ns
t_{SCLKH}	CLKOUT Width High	2.5		ns
t_{SCLKL}	CLKOUT Width Low	2.5		ns

[1] Command pins include: $\overline{SRAS}$, $\overline{SCAS}$, $\overline{SWE}$, SDQM, $\overline{SMS3–0}$, SA10, SCKE.

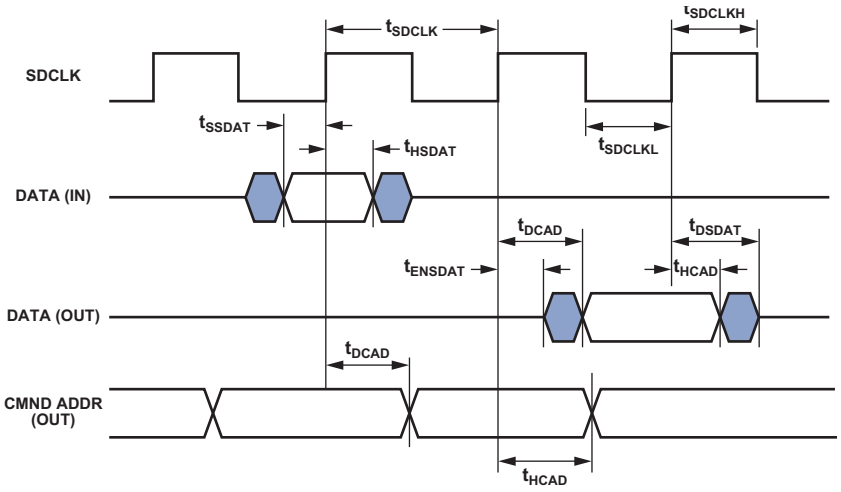

Figure 12. SDRAM Interface Timing

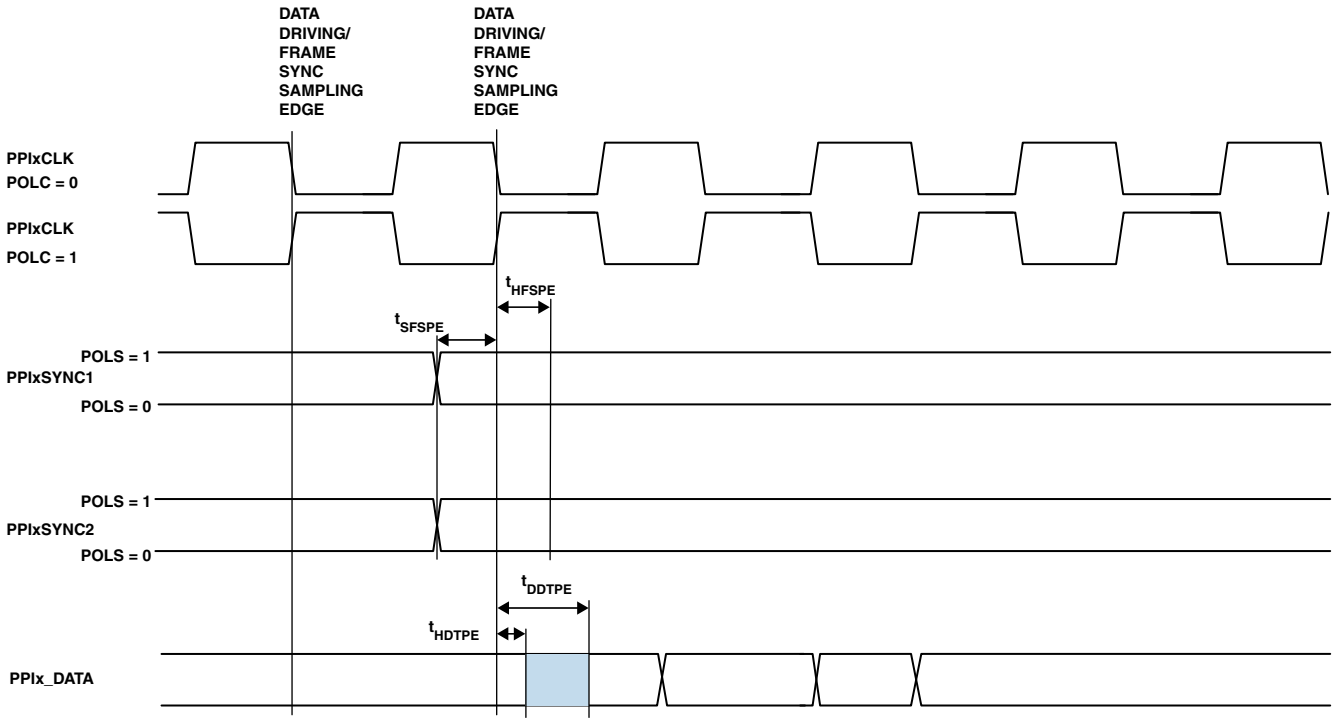

Figure 19. PPI GP Tx Mode with External Frame Sync Timing (Bit 4 of PLL_CTL Set)

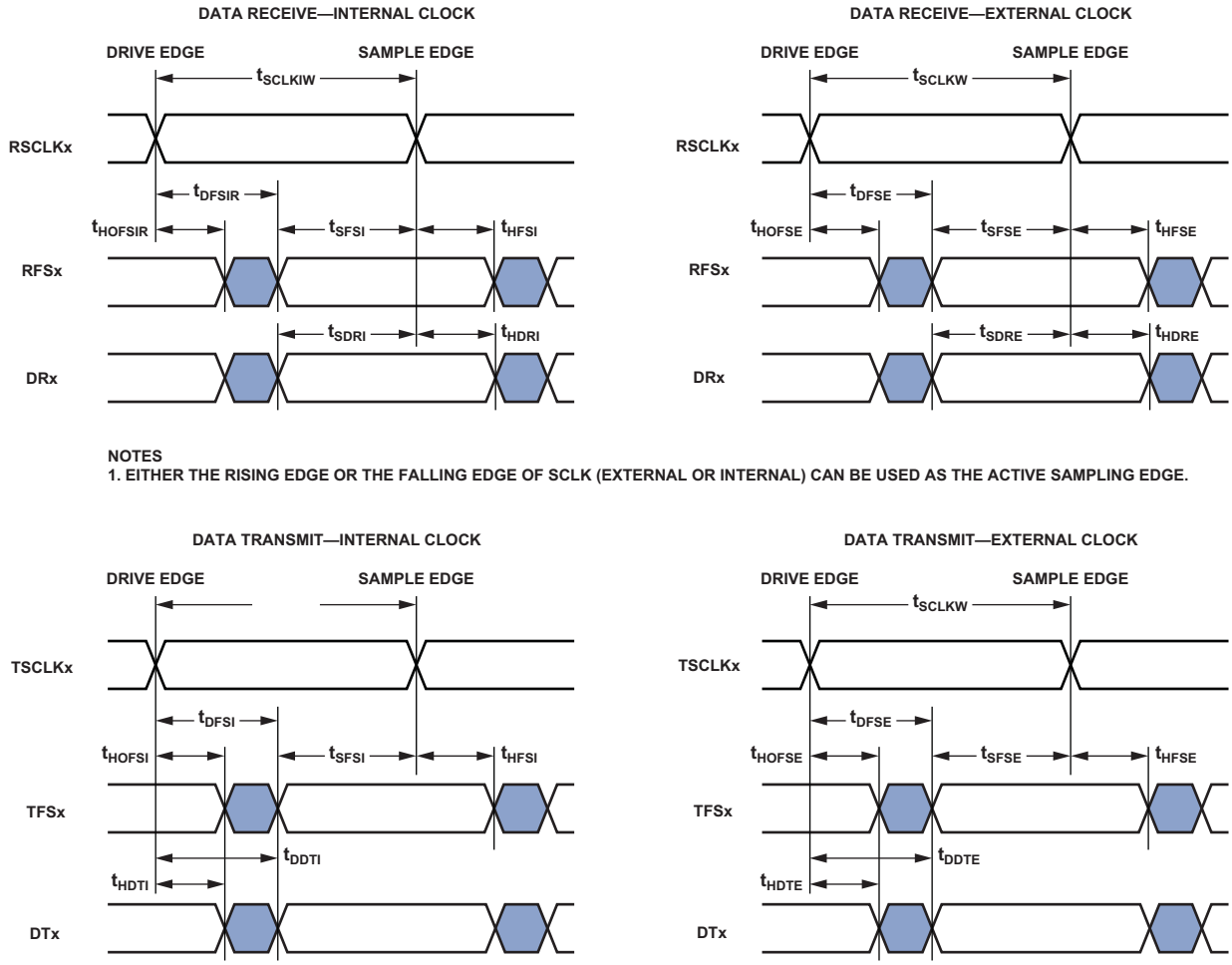

Figure 20. Serial Ports

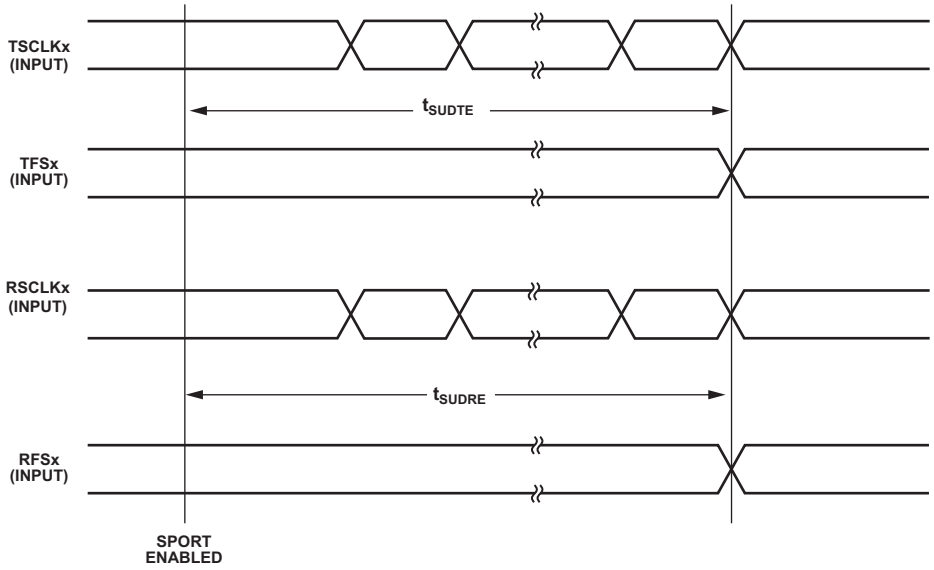

Figure 21. Serial Port Start Up with External Clock and Frame Sync

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.

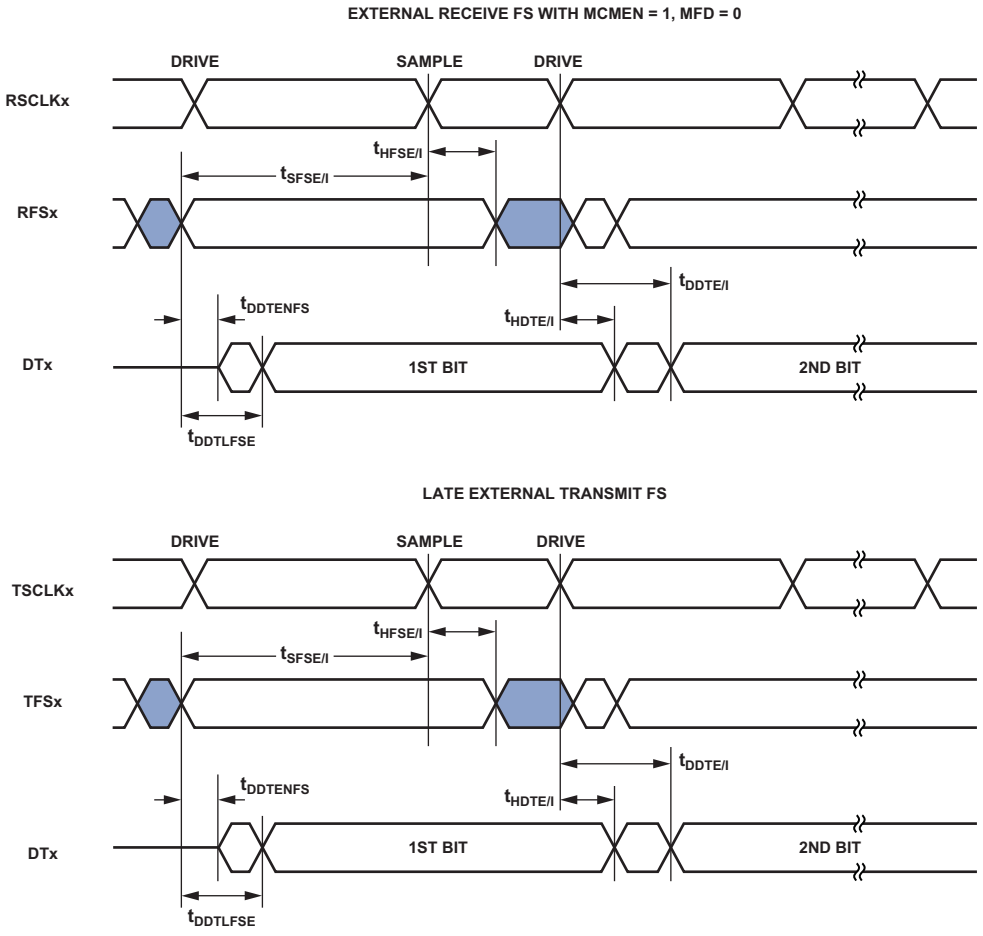

Figure 22. External Late Frame Sync

Timer Cycle Timing

Table 30 and Figure 27 describe timer expired operations. The input signal is asynchronous in width capture mode and external clock mode and has an absolute maximum input frequency of $f_{SCLK}/2$ MHz.

Table 30. Timer Cycle Timing

Parameter		Min	Max	Unit
Timing Characteristics				
t_{WL}	Timer Pulse Width Input Low[1] (Measured in SCLK Cycles)	1		SCLK
t_{WH}	Timer Pulse Width Input High[1] (Measured in SCLK Cycles)	1		SCLK
Switching Characteristic				
t_{HTO}	Timer Pulse Width Output[2] (Measured in SCLK Cycles)	1	$(2^{32}-1)$	SCLK

[1] The minimum pulse widths apply for TMRx input pins in width capture and external clock modes. They also apply to the PF1 or PPIxCLK input pins in PWM output mode.
[2] The minimum time for t_{HTO} is one cycle, and the maximum time for t_{HTO} equals $(2^{32}-1)$ cycles.

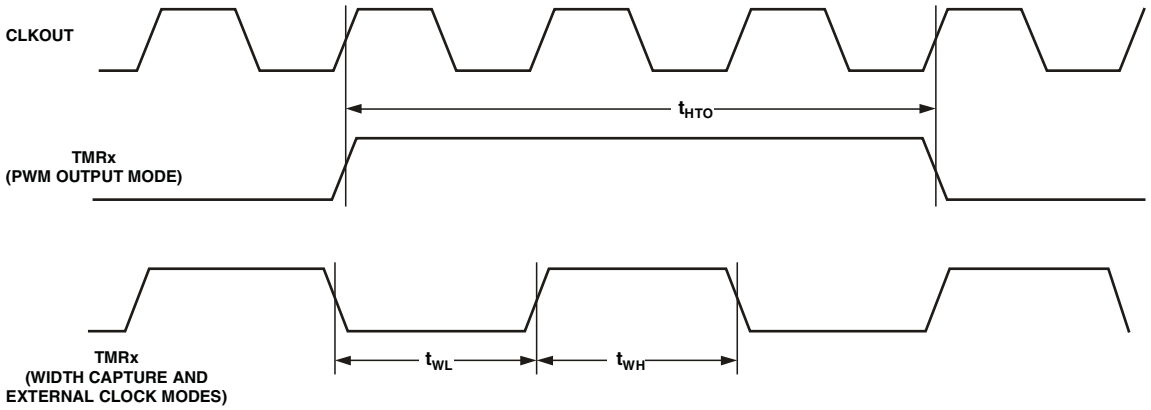

Figure 27. Timer PWM_OUT Cycle Timing

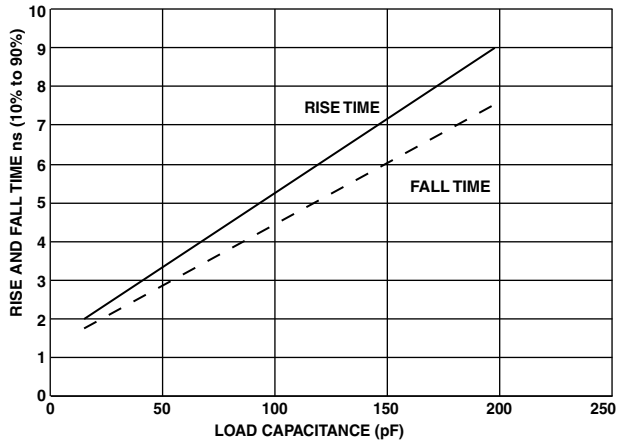

Figure 43. Typical Rise and Fall Times (10% to 90%) versus Load Capacitance for Driver B at V_{DDEXT} (max)

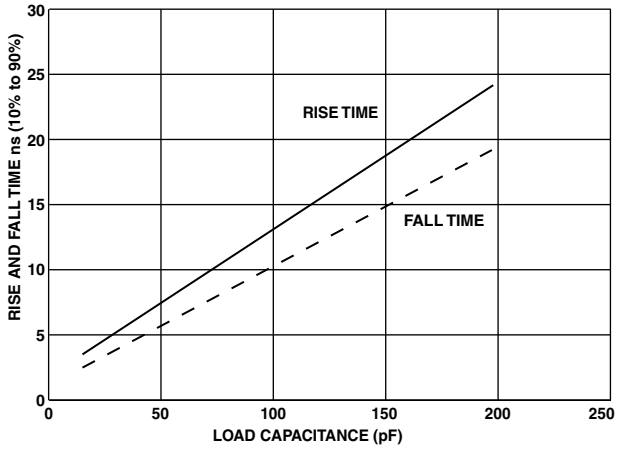

Figure 44. Typical Rise and Fall Times (10% to 90%) versus Load Capacitance for Driver C at V_{DDEXT} (min)

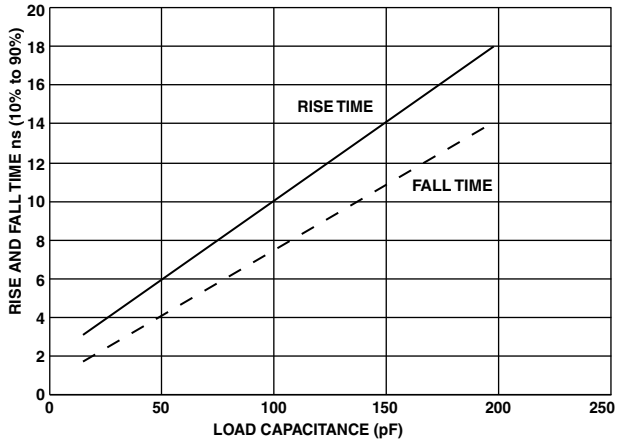

Figure 45. Typical Rise and Fall Times (10% to 90%) versus Load Capacitance for Driver C at V_{DDEXT} (max)

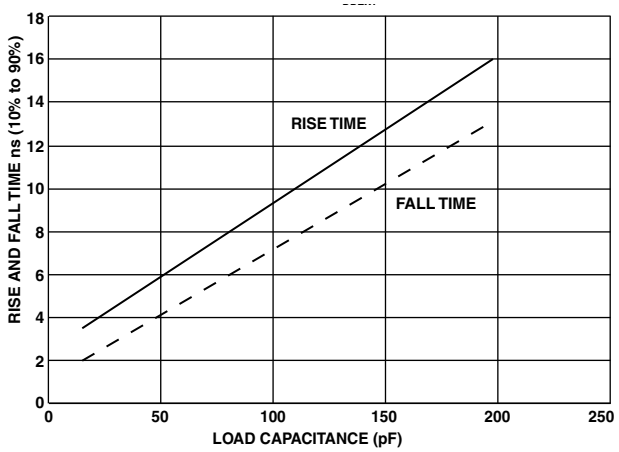

Figure 46. Typical Rise and Fall Times (10% to 90%) versus Load Capacitance for Driver D at V_{DDEXT} (min)

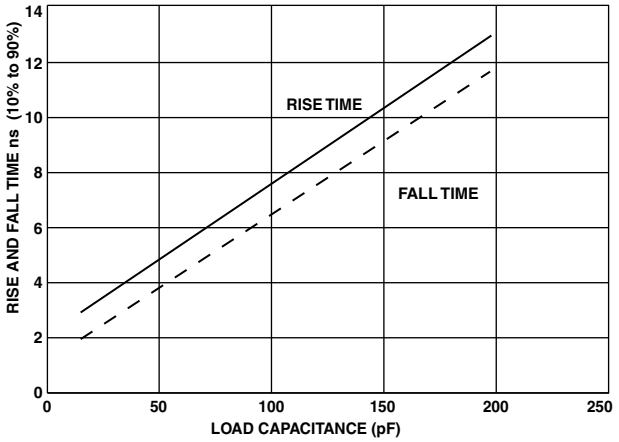

Figure 47. Typical Rise and Fall Times (10% to 90%) versus Load Capacitance for Driver D at V_{DDEXT} (max)

ENVIRONMENTAL CONDITIONS

To determine the junction temperature on the application printed circuit board use:

$$T_J = T_{CASE} + (\Psi_{JT} \times P_D)$$

where:
T_J = junction temperature (°C).

T_{CASE} = case temperature (°C) measured by customer at top center of package.

Ψ_{JT} = from Table 32 on Page 45 through Table 34 on Page 45.

P_D = power dissipation (see Power Dissipation on Page 42 for the method to calculate P_D).

Values of θ_{JA} are provided for package comparison and printed circuit board design considerations. θ_{JA} can be used for a first order approximation of T_J by the equation:

$$T_J = T_A + (\theta_{JA} \times P_D)$$

where:
T_A = ambient temperature (°C).

Table 36. 256-Ball CSP_BGA (17 mm × 17 mm) Ball Assignment (Alphabetically by Signal)

Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.
$\overline{ABE0}$	C12	$\overline{BR}$	B11	DT0SEC	P16	GND	M9	PPI0D13	E2
$\overline{ABE1}$	D12	BYPASS	F4	DT1PRI	T15	GND	M11	PPI0D14	E4
$\overline{ABE2}$	A12	CLKIN	F1	DT1SEC	N14	GND	N9	PPI0D15	D1
$\overline{ABE3}$	A13	DATA0	C15	$\overline{EMU}$	T11	MISO	T12	PPI0SYNC1	C1
ADDR02	B15	DATA01	D14	GND	E7	MOSI	R12	PPI0SYNC2	D3
ADDR03	A15	DATA02	D13	GND	E9	NC	L13	PPI0SYNC3	D2
ADDR04	C14	DATA03	D15	GND	E11	NC	R15	PPI1CLK	E5
ADDR05	B14	DATA04	B16	GND	F8	NC	T2	PPI1D0	P4
ADDR06	A14	DATA05	C16	GND	F9	NMI0	P11	PPI1D1	R4
ADDR07	C13	DATA06	E13	GND	G4	NMI1	P9	PPI1D2	T4
ADDR08	B13	DATA07	D16	GND	G5	PF0	N5	PPI1D3	T3
ADDR09	D6	DATA08	F14	GND	G7	PF01	P5	PPI1D4	R3
ADDR10	B6	DATA09	E15	GND	G8	PF02	R5	PPI1D5	N4
ADDR11	A5	DATA10	F15	GND	G9	PF03	T5	PPI1D6	P3
ADDR12	B5	DATA11	F13	GND	G10	PF04	N6	PPI1D7	N3
ADDR13	C6	DATA12	E16	GND	H2	PF05	T6	PPI1D8	P2
ADDR14	A4	DATA13	E14	GND	H3	PF06	P6	PPI1D9	M4
ADDR15	D5	DATA14	G14	GND	H6	PF07	R6	PPI1D10	N2
ADDR16	C5	DATA15	G15	GND	H7	PF08	P7	PPI1D11	R2
ADDR17	B4	DATA16	F16	GND	H8	PF09	N7	PPI1D12	R1
ADDR18	A3	DATA17	G13	GND	H9	PF10	T7	PPI1D13	P1
ADDR19	B3	DATA18	G16	GND	H10	PF11	R7	PPI1D14	M3
ADDR20	C4	DATA19	H15	GND	H11	PF12	N8	PPI1D15	M2
ADDR21	D4	DATA20	J14	GND	H12	PF13	T8	PPI1SYNC1	N1
ADDR22	A2	DATA21	H14	GND	H13	PF14	R8	PPI1SYNC2	M1
ADDR23	B2	DATA22	J15	GND	J5	PF15	P8	PPI1SYNC3	K4
ADDR24	B1	DATA23	H16	GND	J6	PPI0CLK	C3	$\overline{RESET}$	F3
ADDR25	C2	DATA24	J16	GND	J7	PPI0D0	L4	RFS0	N15
$\overline{AMS0}$	A7	DATA25	K15	GND	J8	PPI0D1	L3	RFS1	P13
$\overline{AMS1}$	B7	DATA26	K14	GND	J9	PPI0D2	L2	RSCLK0	M13
$\overline{AMS2}$	C7	DATA27	K16	GND	J10	PPI0D3	L1	RSCLK1	R13
$\overline{AMS3}$	A6	DATA28	L16	GND	J11	PPI0D4	K3	RX	N12
$\overline{AOE}$	B8	DATA29	M16	GND	K7	PPI0D5	K2	SA10	C11
ARDY	A8	DATA30	N16	GND	K8	PPI0D6	K1	$\overline{SCAS}$	D10
$\overline{ARE}$	C8	DATA31	L15	GND	K9	PPI0D7	J3	SCK	P12
$\overline{AWE}$	D7	DR0PRI	M14	GND	K10	PPI0D8	J2	SCKE	B10
$\overline{BG}$	B12	DR0SEC	P15	GND	K12	PPI0D9	J4	SCLK0	A10
$\overline{BGH}$	D11	DR1PRI	T14	GND	K13	PPI0D10	F2	SCLK1	A11
BMODE0	N11	DR1SEC	N13	GND	L9	PPI0D11	E1	SLEEP	R11
BMODE1	N10	DT0PRI	L14	GND	M7	PPI0D12	E3	$\overline{SMS0}$	D8

Figure 48 lists the top view of the 256-Ball CSP_BGA (17 mm × 17 mm) ball configuration. Figure 49 lists the bottom view.

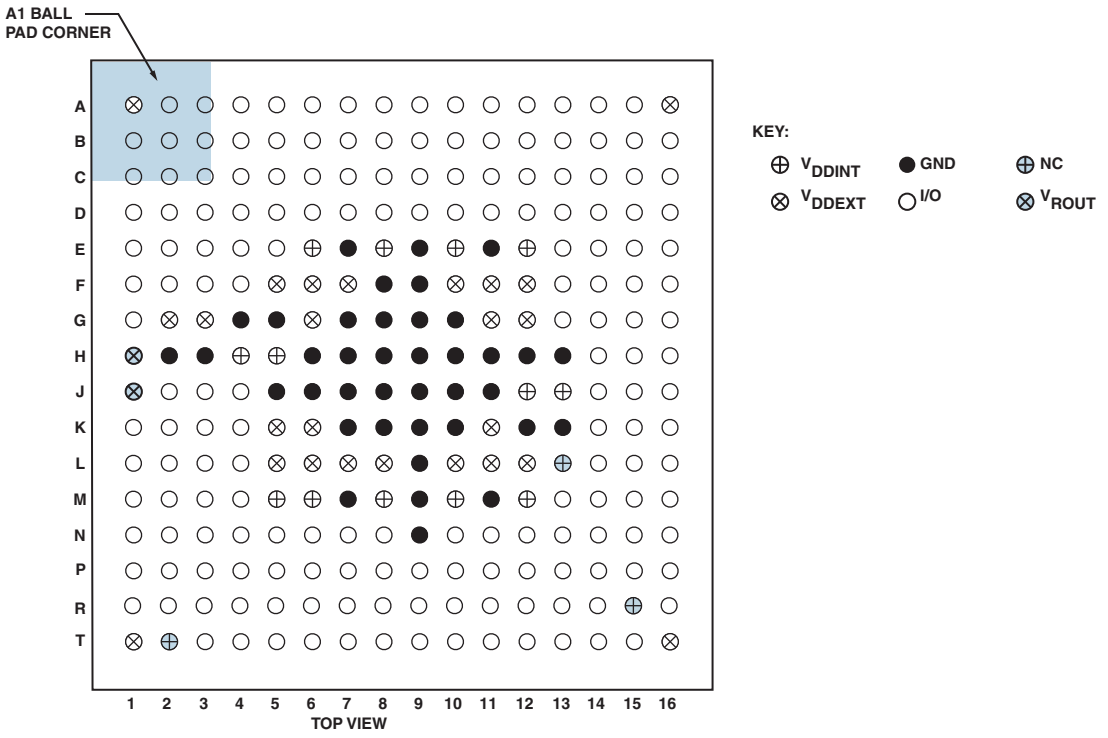

Figure 48. 256-Ball CSP_BGA Ball Configuration (Top View)

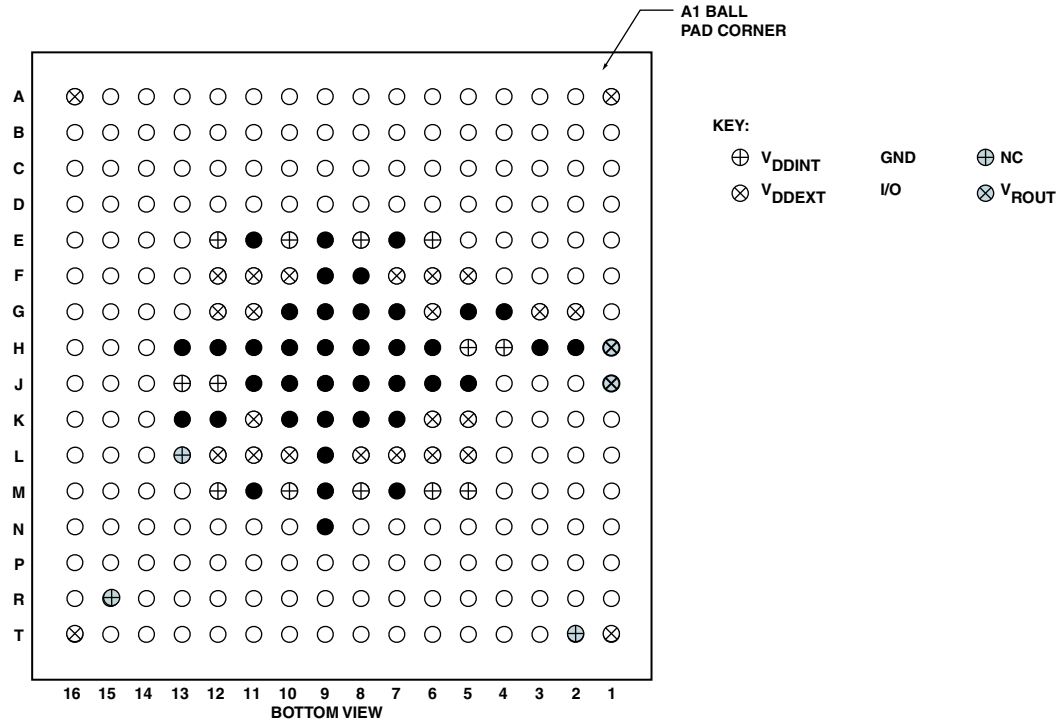

Figure 49. 256-Ball CSP_BGA Ball Configuration (Bottom View)

Table 38. 256-Ball CSP_BGA (12 mm × 12 mm) Ball Assignment (Alphabetically by Signal)

Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.
$\overline{ABE0}$	E11	$\overline{BR}$	B12	DT0SEC	N15	GND	N14
$\overline{ABE1}$	B13	BYPASS	G04	DT1PRI	R15	GND	P02
$\overline{ABE2}$	A14	CLKIN	F01	DT1SEC	T15	GND	P05
$\overline{ABE3}$	A15	DATA0	B16	$\overline{EMU}$	R11	GND	P09
ADDR02	D13	DATA1	C15	GND	C05	GND	P12
ADDR03	G11	DATA2	E12	GND	C11	MISO	R12
ADDR04	B15	DATA3	C16	GND	C13	MOSI	N11
ADDR05	G10	DATA4	E14	GND	D05	NC	M05
ADDR06	B14	DATA5	D15	GND	D06	NC	M13
ADDR07	C14	DATA6	D16	GND	D08	NMI0	P11
ADDR08	F11	DATA7	E15	GND	D14	NMI1	R09
ADDR09	D07	DATA8	F13	GND	E01	PF0	P04
ADDR10	A06	DATA9	F15	GND	E13	PF1	N05
ADDR11	C06	DATA10	F12	GND	F08	PF2	T04
ADDR12	B05	DATA11	F16	GND	F10	PF3	M06
ADDR13	E06	DATA12	F14	GND	G02	PF4	R05
ADDR14	A05	DATA13	G15	GND	G06	PF5	P06
ADDR15	E05	DATA14	G13	GND	G07	PF6	T05
ADDR16	B04	DATA15	G12	GND	G08	PF7	M07
ADDR17	F06	DATA16	H12	GND	G14	PF8	R06
ADDR18	B03	DATA17	H15	GND	H01	PF9	N06
ADDR19	C04	DATA18	H13	GND	H02	PF10	R07
ADDR20	A03	DATA19	H16	GND	H08	PF11	P07
ADDR21	F05	DATA20	H14	GND	H09	PF12	T07
ADDR22	B02	DATA21	J15	GND	H10	PF13	N08
ADDR23	D04	DATA22	J13	GND	J07	PF14	R08
ADDR24	A02	DATA23	J16	GND	J11	PF15	P08
ADDR25	C03	DATA24	K14	GND	J14	PPI0CLK	C02
$\overline{AMS0}$	C08	DATA25	K15	GND	K07	PPI0D0	L01
$\overline{AMS1}$	B07	DATA26	K13	GND	K09	PPI0D1	J05
$\overline{AMS2}$	E07	DATA27	L15	GND	K10	PPI0D2	J03
$\overline{AMS3}$	A07	DATA28	K12	GND	L03	PPI0D3	J04
$\overline{AOE}$	C07	DATA29	L16	GND	L07	PPI0D4	K02
ARDY	D09	DATA30	J12	GND	L09	PPI0D5	H05
$\overline{ARE}$	B08	DATA31	M15	GND	L11	PPI0D6	K01
$\overline{AWE}$	A08	DR0PRI	L12	GND	L14	PPI0D7	H04
$\overline{BG}$	A13	DR0SEC	P16	GND	M04	PPI0D8	K03
$\overline{BGH}$	C12	DR1PRI	M12	GND	M09	PPI0D9	H03
BMODE0	M10	DR1SEC	T14	GND	N07	PPI0D10	F04
BMODE1	N10	DT0PRI	M16	GND	N12	PPI0D11	E02

Table 38. 256-Ball CSP_BGA (12 mm × 12 mm) Ball Assignment (Alphabetically by Signal) (Continued)

Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.
PPI0D12	E03	PPI1SYNC1	K04	TDO	N09	VDDEXT	M14
PPI0D13	D01	PPI1SYNC2	L02	TFS0	L13	VDDEXT	T01
PPI0D14	G05	PPI1SYNC3	L04	TFS1	P14	VDDEXT	T03
PPI0D15	D02	$\overline{\text{RESET}}$	F03	TMS	T10	VDDEXT	T06
PPI0SYNC1	E04	RFS0	R16	$\overline{\text{TRST}}$	P10	VDDEXT	T08
PPI0SYNC2	C01	RFS1	N13	TSCLK0	N16	VDDEXT	T12
PPI0SYNC3	D03	RSCLK0	P15	TSCLK1	R14	VDDEXT	T16
PPI1CLK	B01	RSCLK1	P13	TX/PF26	R13	VDDINT	E08
PPI1D0	R04	RX	T13	VDDEXT	A01	VDDINT	F07
PPI1D1	N04	SA10	D11	VDDEXT	A04	VDDINT	F09
PPI1D2	R03	$\overline{\text{SCAS}}$	D10	VDDEXT	A09	VDDINT	G09
PPI1D3	N03	SCK	M11	VDDEXT	A16	VDDINT	H06
PPI1D4	T02	SCKE	B10	VDDEXT	B06	VDDINT	H07
PPI1D5	P03	SCLK0	A11	VDDEXT	B11	VDDINT	H11
PPI1D6	R02	SCLK1	A12	VDDEXT	D12	VDDINT	J08
PPI1D7	R01	SLEEP	T11	VDDEXT	E16	VDDINT	J09
PPI1D8	P01	$\overline{\text{SMS0}}$	E09	VDDEXT	F02	VDDINT	J10
PPI1D9	M03	$\overline{\text{SMS1}}$	B09	VDDEXT	G03	VDDINT	K08
PPI1D10	N02	$\overline{\text{SMS2}}$	C09	VDDEXT	G16	VDDINT	K11
PPI1D11	L06	$\overline{\text{SMS3}}$	A10	VDDEXT	J06	VDDINT	L08
PPI1D12	N01	$\overline{\text{SRAS}}$	C10	VDDEXT	K06	VDDINT	M08
PPI1D13	M02	$\overline{\text{SWE}}$	E10	VDDEXT	K16	VROUT0	J01
PPI1D14	K05	TCK	T09	VDDEXT	L05	VROUT1	J02
PPI1D15	M01	TDI	R10	VDDEXT	L10	XTAL	G01

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 40. 297-Ball PBGA Ball Assignment (Alphabetically by Signal) (Continued)

Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.
GND	AF26	PPI0D7	P01	RSCLK0	AD26	VDDEXT	K13
MISO	AE19	PPI0D8	P02	RSCLK1	AF21	VDDEXT	K14
MOSI	AE20	PPI0D9	N02	RX	AE21	VDDEXT	K15
NC	K02	PPI0D10	G02	SA10	B19	VDDEXT	L10
NC	L01	PPI0D11	G01	$\overline{\text{SCAS}}$	A18	VDDEXT	M10
NC	L02	PPI0D12	F02	SCK	AF19	VDDEXT	N10
NC	AD25	PPI0D13	F01	SCKE	B17	VDDEXT	P10
NC	AE13	PPI0D14	E02	SCLK0	A19	VDDEXT	R10
NC	AE26	PPI0D15	E01	SCLK1	A20	VDDEXT	T10
NMI0	AF18	PPI0SYNC1	D01	SLEEP	AF17	VDDEXT	U10
NMI1	AF13	PPI0SYNC2	D02	$\overline{\text{SMS0}}$	A15	VDDEXT	U11
PF0	AE05	PPI0SYNC3	C01	$\overline{\text{SMS1}}$	B15	VDDEXT	U12
PF1	AF05	PPI1CLK	B01	$\overline{\text{SMS2}}$	A16	VDDEXT	U13
PF2	AE06	PPI1D0	AF04	$\overline{\text{SMS3}}$	B16	VDDINT	J16
PF3	AF06	PPI1D1	AE04	$\overline{\text{SRAS}}$	A17	VDDINT	J17
PF4	AE07	PPI1D2	AF03	$\overline{\text{SWE}}$	B18	VDDINT	J18
PF5	AF07	PPI1D3	AE03	TCK	AF14	VDDINT	K16
PF6	AE08	PPI1D4	AF02	TDI	AF15	VDDINT	K17
PF7	AF08	PPI1D5	AE01	TDO	AE14	VDDINT	K18
PF8	AE09	PPI1D6	AD02	TFS0	AB25	VDDINT	L18
PF9	AF09	PPI1D7	AD01	TFS1	AE24	VDDINT	M18
PF10	AE10	PPI1D8	AC02	TMS	AF16	VDDINT	N18
PF11	AF10	PPI1D9	AC01	$\overline{\text{TRST}}$	AE15	VDDINT	P18
PF12	AE11	PPI1D10	AB02	TSCLK0	AA26	VDDINT	R18
PF13	AF11	PPI1D11	AB01	TSCLK1	AF23	VDDINT	T18
PF14	AE12	PPI1D12	AA02	TX/PF26	AF20	VDDINT	U15
PF15	AF12	PPI1D13	AA01	VDDEXT	J10	VDDINT	U16
PPI0CLK	C02	PPI1D14	Y02	VDDEXT	J11	VDDINT	U17
PPI0D0	V02	PPI1D15	Y01	VDDEXT	J12	VDDINT	U18
PPI0D1	U01	PPI1SYNC1	W01	VDDEXT	J13	VROUT0	M01
PPI0D2	U02	PPI1SYNC2	W02	VDDEXT	J14	VROUT1	N01
PPI0D3	T01	PPI1SYNC3	V01	VDDEXT	J15	XTAL	K01
PPI0D4	T02	$\overline{\text{RESET}}$	H02	VDDEXT	K10		
PPI0D5	R01	RFS0	AC26	VDDEXT	K11		
PPI0D6	R02	RFS1	AE22	VDDEXT	K12		