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
Clock Rate	160MHz
Non-Volatile Memory	External
On-Chip RAM	112kB
Voltage - I/O	3.30V
Voltage - Core	2.50V
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	176-LQFP
Supplier Device Package	176-LQFP (24x24)
Purchase URL	https://www.e-xfl.com/product-detail/analog-devices/adsp-21991bstz

- EE-158: ADSP-2181 EZ-Kit Lite IDMA to PC Printer Port Interface
- EE-159: Initializing DSP System & Control Registers From C and C++
- EE-164: Advanced EPROM Boot and No-boot Scenarios with ADSP-219x DSPs
- EE-168: Using Third Overtone Crystals with the ADSP-218x DSP
- EE-17: ADSP-2187L Memory Organization
- EE-18: Choosing and Using FFTs for ADSP-21xx
- EE-188: Using C To Implement Interrupt-Driven Systems On ADSP-219x DSPs
- EE-2: Using ADSP-218x I/O Space
- EE-226: ADSP-2191 DSP Host Port Booting
- EE-227: CAN Configuration Procedure for ADSP-21992 DSPs
- EE-249: Implementing Software Overlays on ADSP-218x DSPs with VisualDSP++®
- EE-32: Language Extensions: Memory Storage Types, ASM & Inline Constructs
- EE-35: Troubleshooting your ADSP-218x EZ-ICE
- EE-356: Emulator and Evaluation Hardware Troubleshooting Guide for CCES Users
- EE-38: ADSP-2181 IDMA Port - Cycle Steal Timing
- EE-39: Interfacing 5V Flash Memory to an ADSP-218x (Byte Programming Algorithm)
- EE-5: ADSP-218x Full Memory Mode vs. Host Memory Mode
- EE-60: Simulating an RS-232 UART Using the Synchronous Serial Ports on the ADSP-21xx Family DSPs
- EE-64: Setting Mode Pins on Reset
- EE-68: Analog Devices JTAG Emulation Technical Reference
- EE-71: Minimum Rise Time Specs for Critical Interrupt and Clock Signals on the ADSP-21x1/21x5
- EE-74: Analog Devices Serial Port Development and Troubleshooting Guide
- EE-78: BDMA Usage on 100 pin ADSP-218x DSPs Configured for IDMA Use
- EE-79: EPROM Booting In Host Mode with 100 Pin 218x Processors
- EE-82: Using an ADSP-2181 DSP's IO Space to IDMA Boot Another ADSP-2181
- EE-89: Implementing A Software UART on the ADSP-2181 EZ-Kit-Lite
- EE-96: Interfacing Two AD73311 Codecs to the ADSP-218x

Data Sheet

- ADSP-21991: Mixed Signal DSP Controller Data Sheet

Integrated Circuit Anomalies

- ADSP-2199x Anomaly List for Revisions up to 1.1

Processor Manuals

- ADSP 21xx Processors: Manuals
- ADSP-2199x Mixed Signal DSP Controller Hardware Reference
- ADSP-219x DSP Instruction Set Reference
- Using the ADSP-2100 Family Volume 1

Product Highlight

- The ADSP-2199x Family: Mixed-Signal DSPs For Embedded Control & Signal Processing Applications Product Highlight

Software Manuals

- VisualDSP++ 3.5 Assembler and Preprocessor Manual for ADSP-218x and ADSP-219x DSPs
- VisualDSP++ 3.5 C Compiler and Library Manual for ADSP-218x DSPs
- VisualDSP++ 3.5 C/C++ Compiler and Library Manual for ADSP-219x Processors
- VisualDSP++ 3.5 Component Software Engineering User's Guide for 16-Bit Processors
- VisualDSP++ 3.5 Getting Started Guide for 16-Bit Processors
- VisualDSP++ 3.5 Kernel VDK User's Guide for 16-Bit Processors
- VisualDSP++ 3.5 Linker and Utilities Manual for 16-Bit Processors
- VisualDSP++ 3.5 Loader Manual for 16-Bit Processors
- VisualDSP++ 3.5 Quick Installation Reference Card
- VisualDSP++ 3.5 User's Guide for 16-Bit Processors

SOFTWARE AND SYSTEMS REQUIREMENTS

- Software and Tools Anomalies Search

TOOLS AND SIMULATIONS

- ADSP-21xx Processors: Software and Tools

REFERENCE MATERIALS

Technical Articles

- DSP Motor Control in Domestic Appliance Applications

KEY FEATURES (continued)
Integrated Power-On-Reset (POR) Generator
Flexible Power Management with Selectable Power-Down and Idle Modes
2.5 V Internal Operation with 3.3 V I/O
Operating Temperature Range of –40°C to +85°C
196-Ball Mini-BGA Package
176-Lead LQFP Package

TARGET APPLICATIONS
Industrial Motor Drives
Uninterruptible Power Supplies
Optical Networking Control
Data Acquisition Systems
Test and Measurement Systems
Portable Instrumentation

TABLE OF CONTENTS

GENERAL DESCRIPTION

The ADSP-21991 is a mixed signal DSP controller based on the ADSP-219x DSP Core, suitable for a variety of high performance industrial motor control and signal processing applications that require the combination of a high performance DSP and the mixed signal integration of embedded control peripherals such as analog-to-digital conversion.

The ADSP-21991 integrates the fixed point ADSP-219x family base architecture with a serial port, an SPI compatible port, a DMA controller, three programmable timers, general-purpose Programmable Flag pins, extensive interrupt capabilities, on-chip program and data memory spaces, and a complete set of embedded control peripherals that permits fast motor control and signal processing in a highly integrated environment.

The ADSP-21991 architecture is code compatible with previous ADSP-217x based ADMCxxx products. Although the architectures are compatible, the ADSP-21991, with ADSP-219x architecture, has a number of enhancements over earlier architectures. The enhancements to computational units, data address generators, and program sequencer make the ADSP-21991 more flexible and easier to program than the previous ADSP-21xx embedded DSPs.

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

The ADSP-21991 integrates 40K words of on-chip memory configured as 32K words (24-bit) of program RAM, and 8K words (16-bit) of data RAM.

Fabricated in a high speed, low power, CMOS process, the ADSP-21991 operates with a 6.25 ns instruction cycle time for a 160 MHz CCLK and with a 6.67 ns instruction cycle time for a 150 MHz CCLK. All instructions, except two multiword instructions, execute in a single DSP cycle.

The flexible architecture and comprehensive instruction set of the ADSP-21991 support multiple operations in parallel. For example, in one processor cycle, the ADSP-21991 can:

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

These operations take place while the processor continues to:

- Receive and transmit data through the serial port
- Receive or transmit data over the SPI port
- Access external memory through the external memory interface
- Decrement the timers
- Operate the embedded control peripherals (ADC, PWM, EIU, etc.)

DSP Core Architecture

- 6.25 ns instruction cycle time (internal), for up to 160 MIPS sustained performance (6.67 ns instruction cycle time for 150 MIPS sustained performance)
- ADSP-218x family code compatible with the same easy to use algebraic syntax
- Single cycle instruction execution
- Up to 1M words of addressable memory space with twenty four bits of addressing width
- Dual purpose program memory for both instruction and data storage
- Fully transparent instruction cache allows dual operand fetches in every instruction cycle
- Unified memory space permits flexible address generation, using two independent DAG units
- Independent ALU, multiplier/accumulator, and barrel shifter computational units with dual 40-bit accumulators
- Single cycle context switch between two sets of computational and DAG registers
- Parallel execution of computation and memory instructions
- Pipelined architecture supports efficient code execution at speeds up to 160 MIPS
- Register file computations with all nonconditional, nonparallel computational instructions
- Powerful program sequencer provides zero overhead looping and conditional instruction execution
- Architectural enhancements for compiled C code efficiency
- Architecture enhancements beyond ADSP-218x family are supported with instruction set extensions for added registers, ports, and peripherals.

The clock generator module of the ADSP-21991 includes clock control logic that allows the user to select and change the main clock frequency. The module generates two output clocks: the DSP core clock, CCLK; and the peripheral clock, HCLK. CCLK can sustain clock values of up to 160 MHz, while HCLK can be equal to CCLK or CCLK/2 for values up to a maximum 80 MHz peripheral clock at the 160 MHz CCLK rate.

The ADSP-21991 instruction set provides flexible data moves and multifunction (one or two data moves with a computation) instructions. Every single word instruction can be executed in a single processor cycle. The ADSP-21991 assembly language uses an algebraic syntax for ease of coding and readability. A comprehensive set of development tools supports program development.

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

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

A powerful program sequencer controls the flow of instruction execution. The sequencer supports conditional jumps, subroutine calls, and low interrupt overhead. With internal loop counters and loop stacks, the ADSP-21991 executes looped code with zero overhead; no explicit jump instructions are required to maintain loops.

Two data address generators (DAGs) provide addresses for simultaneous dual operand fetches (from data memory and program memory). Each DAG maintains and updates four 16-bit address pointers. Whenever the pointer is used to access data (indirect addressing), it is pre- or post-modified by the value of one of four possible modify registers. A length value and base address may be associated with each pointer to implement automatic modulo addressing for circular buffers. Page registers in the DAGs allow circular addressing within 64K word boundaries of each of the 256 memory pages, but these buffers may not cross page boundaries. Secondary registers duplicate all the primary registers in the DAGs; switching between primary and secondary registers provides a fast context switch.

255) are available for external peripheral devices. External I/O pages have their own select pin ($\overline{\text{IOMS}}$). The DSP instruction set provides instructions for accessing I/O space.

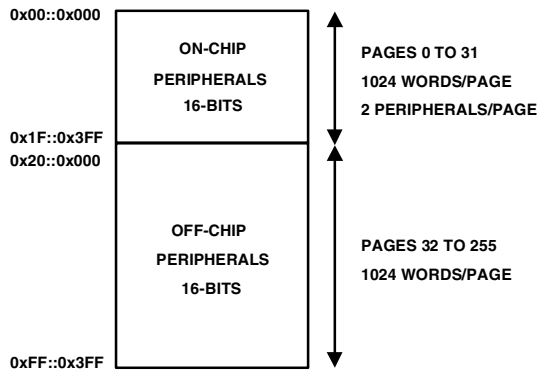

Figure 3. I/O Memory Map

Boot Memory Space

Boot memory space consists of one off-chip bank with 254 pages. The $\overline{\text{BMS}}$ memory bank pin selects boot memory space. Both the ADSP-219x core and DMA capable peripherals can access the off-chip boot memory space of the DSP. After reset, the DSP always starts executing instructions from the on-chip boot ROM.

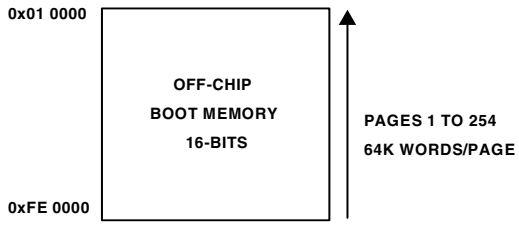

Figure 4. Boot Memory Map

Bus Request and Bus Grant

The ADSP-21991 can relinquish control of the data and address buses to an external device. When the external device requires access to the bus, it asserts the bus request ($\overline{\text{BR}}$) signal. The ($\overline{\text{BR}}$) signal is arbitrated with core and peripheral requests. External Bus requests have the lowest priority. If no other internal request is pending, the external bus request will be granted. Due to synchronizer and arbitration delays, bus grants will be provided with a minimum of three peripheral clock delays. The ADSP-21991 will respond to the bus grant by:

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

The ADSP-21991 will halt program execution if the bus is granted to an external device and an instruction fetch or data read/write request is made to external general-purpose or peripheral memory spaces. If an instruction requires two external memory read accesses, the bus will not be granted between the two accesses. If an instruction requires an external memory read and an external memory write access, the bus may be granted between the two accesses. The external memory interface can be configured so that the core will have exclusive use of the interface. DMA and Bus Requests will be granted. When the external device releases $\overline{\text{BR}}$, the DSP releases $\overline{\text{BG}}$ and continues program execution from the point at which it stopped.

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

The ADSP-21991 asserts the $\overline{\text{BGH}}$ pin when it is ready to start another external port access, but is held off because the bus was previously granted. This mechanism can be extended to define more complex arbitration protocols for implementing more elaborate multimaster systems.

DMA Controller

The ADSP-21991 has a DMA controller that supports automated data transfers with minimal overhead for the DSP core. Cycle stealing DMA transfers can occur between the ADSP-21991 internal memory and any of its DMA capable peripherals. Additionally, DMA transfers can be accomplished between any of the DMA capable peripherals and external devices connected to the external memory interface. DMA capable peripherals include the SPORT and SPI ports, and ADC Control module. Each individual DMA capable peripheral has a dedicated DMA channel. To describe each DMA sequence, the DMA controller uses a set of parameters—called a DMA descriptor. When successive DMA sequences are needed, these DMA descriptors can be linked or chained together, so the completion of one DMA sequence auto initiates and starts the next sequence. DMA sequences do not contend for bus access with the DSP core, instead DMAs “steal” cycles to access memory.

All DMA transfers use the DMA bus shown in Figure 1 on Page 4. Because all of the peripherals use the same bus, arbitration for DMA bus access is needed. The arbitration for DMA bus access appears in Table 1.

Table 1. I/O Bus Arbitration Priority

DMA Bus Master	Arbitration Priority
SPORT Receive DMA	0—Highest
SPORT Transmit DMA	1
ADC Control DMA	2
SPI Receive/Transmit DMA	3
Memory DMA	4—Lowest

DSP Peripherals Architecture

The ADSP-21991 contains a number of special purpose, embedded control peripherals, which can be seen in the Functional Block Diagram on Page 1. The ADSP-21991 contains a high performance, 8-channel, 14-bit ADC system with dual channel simultaneous sampling ability across four pairs of inputs. An internal precision voltage reference is also available as part of the ADC system. In addition, a 3-phase, 16-bit, center based PWM generation unit can be used to produce high accuracy PWM signals with minimal processor overhead. The ADSP-21991 also contains a flexible incremental encoder interface unit

Table 2. Interrupt Priorities/Addresses

Interrupt	IMASK/ IRPTL	Vector Address
User Assigned Interrupt (USR9)	13	0x00 01A0
User Assigned Interrupt (USR10)	14	0x00 01C0
User Assigned Interrupt (USR11) —Lowest Priority	15	0x00 01E0

There is no assigned priority for the peripheral interrupts after reset. To assign the peripheral interrupts a different priority, applications write the new priority to their corresponding control bits (determined by their ID) in the Interrupt Priority Control register.

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

The Interrupt Control (ICNTL) register controls interrupt nesting and enables or disables interrupts globally.

The IRPTL register is used to force and clear interrupts. On-chip stacks preserve the processor status and are automatically maintained during interrupt handling. To support interrupt, loop, and subroutine nesting, the PC stack is 33 levels deep, the loop stack is 8 levels deep, and the status stack is 16 levels deep. To prevent stack overflow, the PC stack can generate a stack level interrupt if the PC stack falls below 3 locations full or rises above 28 locations full.

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

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

At reset, interrupt servicing is disabled.

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

Peripheral Interrupt Controller

The Peripheral Interrupt Controller is a dedicated peripheral unit of the ADSP-21991 (accessed via IO mapped registers). The peripheral interrupt controller manages the connection of up to 32 peripheral interrupt requests to the DSP core.

For each peripheral interrupt source, there is a unique 4-bit code that allows the user to assign the particular peripheral interrupt to any one of the 12 user assignable interrupts of the embedded ADSP-219x core. Therefore, the peripheral interrupt controller of the ADSP-21991 contains eight, 16-bit Interrupt Priority Registers (Interrupt Priority Register 0 (IPR0) to Interrupt Priority Register 7 (IPR7)).

Each Interrupt Priority Register contains a four 4-bit codes; one specifically assigned to each peripheral interrupt. The user may write a value between 0x0 and 0xB to each 4-bit location in order to effectively connect the particular interrupt source to the corresponding user assignable interrupt of the ADSP-219x core.

Writing a value of 0x0 connects the peripheral interrupt to the USR0 user assignable interrupt of the ADSP-219x core while writing a value of 0xB connects the peripheral interrupt to the USR11 user assignable interrupt. The core interrupt USR0 is the highest priority user interrupt, while USR11 is the lowest priority. Writing a value between 0xC and 0xF effectively disables the peripheral interrupt by not connecting it to any ADSP-219x core interrupt input. The user may assign more than one peripheral interrupt to any given ADSP-219x core interrupt. In that case, the onus is on the user software in the interrupt vector table to determine the exact interrupt source through reading status bits.

This scheme permits the user to assign the number of specific interrupts that are unique to their application to the interrupt scheme of the ADSP-219x core. The user can then use the existing interrupt priority control scheme to dynamically control the priorities of the 12 core interrupts.

Low Power Operation

The ADSP-21991 has four low power options that significantly reduce the power dissipation when the device operates under standby conditions. To enter any of these modes, the DSP executes an IDLE instruction. The ADSP-21991 uses the configuration of the PD, STCK, and STALL bits in the PLLCTL register to select between the low power modes as the DSP executes the IDLE instruction. Depending on the mode, an IDLE shuts off clocks to different parts of the DSP in the different modes. The low power modes are:

- Idle
- Power-Down Core
- Power-Down Core/Peripherals
- Power-Down All

Idle Mode

When the ADSP-21991 is in Idle mode, the DSP core stops executing instructions, retains the contents of the instruction pipeline, and waits for an interrupt. The core clock and peripheral clock continue running.

To enter Idle mode, the DSP can execute the IDLE instruction anywhere in code. To exit Idle mode, the DSP responds to an interrupt and (after two cycles of latency) resumes executing instructions.

Power-Down Core Mode

When the ADSP-21991 is in Power-Down Core mode, the DSP core clock is off, but the DSP retains the contents of the pipeline and keeps the PLL running. The peripheral bus keeps running, letting the peripherals receive data.

Booting Modes

The ADSP-21991 supports a number of different boot modes that are controlled by the three dedicated hardware boot mode control pins (BMODE2, BMODE1, and BMODE0). The use of three boot mode control pins means that up to eight different boot modes are possible. Of these only five modes are valid on the ADSP-21991. The ADSP-21991 exposes the boot mechanism to software control by providing a nonmaskable boot interrupt that vectors to the start of the on-chip ROM memory block (at address 0xFF0000). A boot interrupt is automatically initiated following either a hardware initiated reset, via the $\overline{\text{RESET}}$ pin, or a software initiated reset, via writing to the Software Reset register. Following either a hardware or a software reset, execution always starts from the boot ROM at address 0xFF0000, irrespective of the settings of the BMODE2, BMODE1, and BMODE0 pins. The dedicated BMODE2, BMODE1, and BMODE0 pins are sampled at hardware reset.

The particular boot mode for the ADSP-21991 associated with the settings of the BMODE2, BMODE1, BMODE0 pins is defined in Table 3.

Table 3. Summary of Boot Modes

Boot Mode	BMODE2	BMODE1	BMODE0	Function
0	0	0	0	Illegal – Reserved
1	0	0	1	Boot from External 8-bit Memory over EMI
2	0	1	0	Execute from External 8-bit Memory
3	0	1	1	Execute from External 16-bit Memory
4	1	0	0	Boot from SPI ≤ 4K bits
5	1	0	1	Boot from SPI > 4K bits
6	1	1	0	Illegal – Reserved
7	1	1	1	Illegal – Reserved

Instruction Set Description

The ADSP-21991 assembly language instruction set has an algebraic syntax that was designed for ease of coding and readability. The assembly language, which takes full advantage of the unique architecture of the processor, offers the following benefits:

- ADSP-219x assembly language syntax is a superset of and source code compatible (except for two data registers and DAG base address registers) with ADSP-21xx family syntax. It may be necessary to restructure ADSP-21xx programs to accommodate the unified memory space of the ADSP-21991 and to conform to its interrupt vector map.
- The algebraic syntax eliminates the need to remember cryptic assembler mnemonics. For example, a typical arithmetic add instruction, such as AR = AX0 + AY0, resembles a simple equation.
- Every instruction, but two, assembles into a single, 24-bit word that can execute in a single instruction cycle. The exceptions are two dual word instructions. One writes 16-bit or 24-bit immediate data to memory, and the other is an absolute jump/call with the 24-bit address specified in the instruction.
- Multifunction instructions allow parallel execution of an arithmetic, MAC, or shift instruction with up to two fetches or one write to processor memory space during a single instruction cycle.
- Program flow instructions support a wider variety of conditional and unconditional jumps/calls and a larger set of conditions on which to base execution of conditional instructions.

Development Tools

The ADSP-21991 is supported with a complete set of CROSSCORE™ software and hardware development tools, including Analog Devices emulators and VisualDSP++™ development environment. The emulator hardware that supports other ADSP-219x DSPs also fully emulates the ADSP-21991.

The VisualDSP++ project management environment lets programmers develop and debug an application. This environment includes an easy to use assembler (which 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 DSP assembly. The DSP 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 influence on the design 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++ IDDE lets programmers define and manage DSP software development. Its dialog boxes and property pages let programmers configure and manage all of the ADSP-219x development tools, including the color syntax highlighting in the VisualDSP++ editor. This capability permits programmers to:

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

The VisualDSP++ Kernel (VDK) incorporates scheduling and resource management tailored specifically to address the memory and timing constraints of DSP 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, Preemptive, 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 via 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.

VCSE is Analog Devices technology for creating, using, and reusing software components (independent modules of substantial functionality) to quickly and reliably assemble software applications. Download components from the Web and drop them into the application. Publish component archives from within VisualDSP++. VCSE supports component implementation in C/C++ or assembly language.

Use the Expert Linker to visually manipulate the placement of code and data on the embedded system. View memory utilization in a color-coded graphical form, easily move code and data to different areas of the DSP or external memory with the drag of the mouse, examine run time stack and heap usage. 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 DSP emulators use the IEEE 1149.1 JTAG Test Access Port of the ADSP-21991 processor to monitor and control the target board processor during emulation. The emulator provides full speed emulation, allowing inspection and modification of memory, registers, and processor stacks. Non intrusive in-circuit emulation is assured by the use of the processor JTAG interface—target system loading and timing are not affected by the emulator.

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

Designing an Emulator-Compatible DSP Board

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

To use these emulators, the target board must include a header that connects the JTAG port of the DSP 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 the *EE-68: Analog Devices JTAG Emulation Technical Reference* 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.

Additional Information

This data sheet provides a general overview of the ADSP-21991 architecture and functionality. For detailed information on the ADSP-21991 embedded DSP core architecture, instruction set, communications ports and embedded control peripherals, refer to the *ADSP-2199x Mixed Signal DSP Controller Hardware Reference*.

PIN FUNCTION DESCRIPTIONS

ADSP-21991 pin definitions are listed in Table 4. All ADSP-21991 inputs are asynchronous and can be asserted asynchronously to CLKIN (or to TCK for $\overline{\text{TRST}}$).

Unused inputs should be tied or pulled to V_{DDEXT} or GND, except for ADDR21–0, DATA15–0, PF7-0, and inputs that have internal pull-up or pull-down resistors ($\overline{\text{TRST}}$, BMODE0, BMODE1, BMODE2, BYPASS, TCK, TMS, TDI, PWMPOL, $\overline{\text{PWMSR}}$, and $\overline{\text{RESET}}$)—these pins can be left floating. These

Clock In and Clock Out Cycle Timing

Table 5 and Figure 6 describe clock and reset operations. Combinations of CLKIN and clock multipliers must not select core/peripheral clocks in excess of 160 MHz/80 MHz for the ADSP-21991BST and 150 MHz/75 MHz for the ADSP-21991BBC, when the peripheral clock rate is one-half the core clock rate. If the peripheral clock rate is equal to the core clock rate, the maximum peripheral clock rate is 80 MHz for the ADSP-21991BST and 75 MHz for the ADSP-21991BBC. The peripheral clock is supplied to the CLKOUT pins.

When changing from bypass mode to PLL mode, allow 512 HCLK cycles for the PLL to stabilize.

Table 5. Clock In and Clock Out Cycle Timing

Parameter		Min	Max	Unit
Timing Requirements				
t_{CK}	CLKIN Period[1, 2]	10	200	ns
t_{CKL}	CLKIN Low Pulse	4.5		ns
t_{CKH}	CLKIN High Pulse	4.5		ns
t_{WRST}	$\overline{\text{RESET}}$ Asserted Pulsewidth Low	$200t_{CLKOUT}$		ns
t_{MSS}	MSELx/BYPASS Stable Before $\overline{\text{RESET}}$ Deasserted Setup	40		µs
t_{MSH}	MSELx/BYPASS Stable After $\overline{\text{RESET}}$ Deasserted Hold	1000		ns
t_{MSD}	MSELx/BYPASS Stable After $\overline{\text{RESET}}$ Asserted		200	ns
t_{PFD}	Flag Output Disable Time After $\overline{\text{RESET}}$ Asserted		10	ns
Switching Characteristics				
t_{CKOD}	CLKOUT Delay from CLKIN	0	5.8	ns
t_{CKO}	CLKOUT Period[3]	12.5		ns

[1] In clock multiplier mode and MSEL6–0 set for 1:1 (or CLKIN = CCLK), $t_{CK} = t_{CCLK}$.
[2] In bypass mode, $t_{CK} = t_{CCLK}$.
[3] CLKOUT jitter can be as great as 8 ns when CLKOUT frequency is less than 20 MHz. For frequencies greater than 20 MHz, jitter is less than 1 ns.

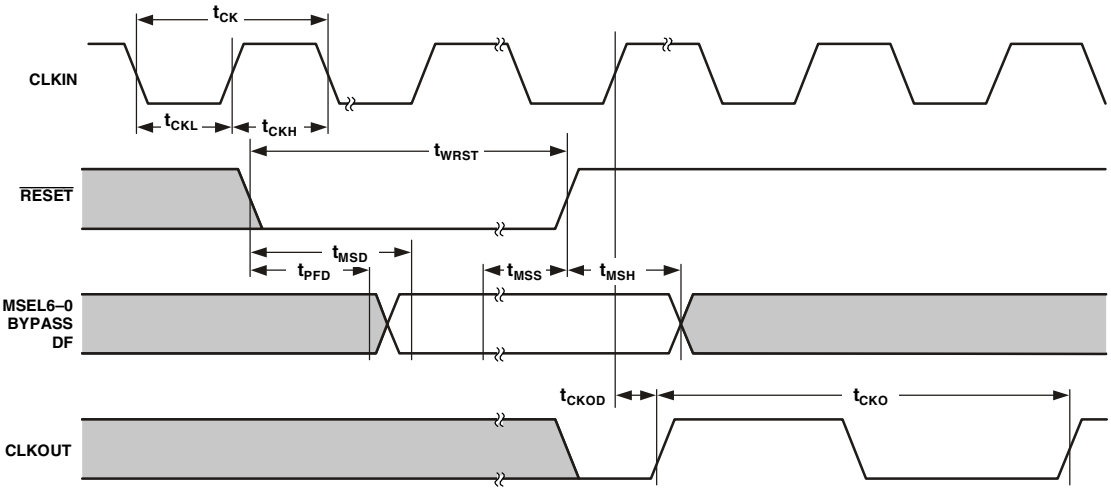

Figure 6. Clock In and Clock Out Cycle Timing

Programmable Flags Cycle Timing

Table 6 and Figure 7 describe Programmable Flag operations.

Table 6. Programmable Flags Cycle Timing

Parameter		Min	Max	Unit
Timing Requirement				
t_{HFI}	Flag Input Hold is Asynchronous	3		ns
Switching Characteristics				
t_{DFO}	Flag Output Delay with Respect to CLKOUT		7	ns
t_{HFO}	Flag Output Hold After CLKOUT High		6	ns

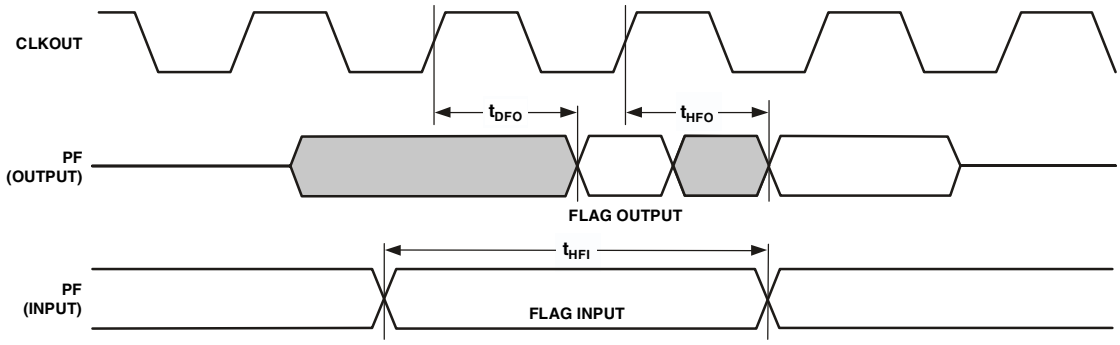

Figure 7. Programmable Flags Cycle Timing

Timer PWM_OUT Cycle Timing

Table 7 and Figure 8 describe timer expired operations. The input signal is asynchronous in "width capture mode" and has an absolute maximum input frequency of 40 MHz.

Table 7. Timer PWM_OUT Cycle Timing

Parameter		Min	Max	Unit
Switching Characteristic				
t_{HTO}	Timer Pulsewidth Output[1]	12.5	$(2^{32}-1)$ cycles	ns

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

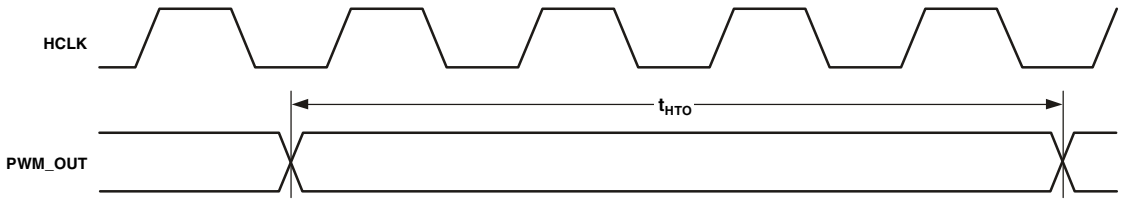

Figure 8. Timer PWM_OUT Cycle Timing

External Port Write Cycle Timing

Table 8 and Figure 9 describe external port write operations.

The external port lets systems extend read/write accesses in three ways: wait states, ACK input, and combined wait states and ACK. To add waits with ACK, the DSP must see ACK low at the rising edge of EMI clock. ACK low causes the DSP to wait, and the DSP requires two EMI clock cycles after ACK goes high to finish the access. For more information, see the External Port chapter in the *ADSP-2199x DSP Hardware Reference*.

Table 8. External Port Write Cycle Timing

Parameter[1, 2]		Min	Max	Unit
Timing Requirements				
t_{AKW}	ACK Strobe Pulsewidth	12.5		ns
t_{DWSAK}	ACK Delay from $\overline{XMS}$ Low		$0.5t_{EMICLK}-1$	ns
Switching Characteristics				
t_{CSWS}	Chip Select Asserted to $\overline{WR}$ Asserted Delay	$0.5t_{EMICLK}-4$		ns
t_{AWS}	Address Valid to $\overline{WR}$ Setup and Delay	$0.5t_{EMICLK}-3$		ns
t_{WSCS}	$\overline{WR}$ Deasserted to Chip Select Deasserted	$0.5t_{EMICLK}-4$		ns
t_{WSA}	$\overline{WR}$ Deasserted to Address Invalid	$0.5t_{EMICLK}-3$		ns
t_{WW}	$\overline{WR}$ Strobe Pulsewidth	$t_{EMICLK}-2+W^3$		ns
t_{CDA}	$\overline{WR}$ to Data Enable Access Delay		0	ns
t_{CDD}	$\overline{WR}$ to Data Disable Access Delay	$0.5t_{EMICLK}-3$	$0.5t_{EMICLK}+4$	ns
t_{DSW}	Data Valid to $\overline{WR}$ Deasserted Setup	$t_{EMICLK}+1+W^3$	$t_{EMICLK}+7+W^3$	ns
t_{DHW}	$\overline{WR}$ Deasserted to Data Invalid Hold Time; E_WHC[4, 5]	3.4		ns
t_{DHW}	$\overline{WR}$ Deasserted to Data Invalid Hold Time; E_WHC[4, 6]	$t_{EMICLK}+3.4$		ns
t_{WWR}	$\overline{WR}$ Deasserted to $\overline{WR}$, $\overline{RD}$ Asserted	t_{HCLK}		ns

[1] t_{EMICLK} is the External Memory Interface clock period. t_{HCLK} is the peripheral clock period.
[2] These are timing parameters that are based on worst-case operating conditions.
[3] W = (number of wait states specified in wait register) × t_{EMICLK}.
[4] Write hold cycle–memory select control registers (MS × CTL).
[5] Write wait state count (E_WWC) = 0
[6] Write wait state count (E_WWC) = 1

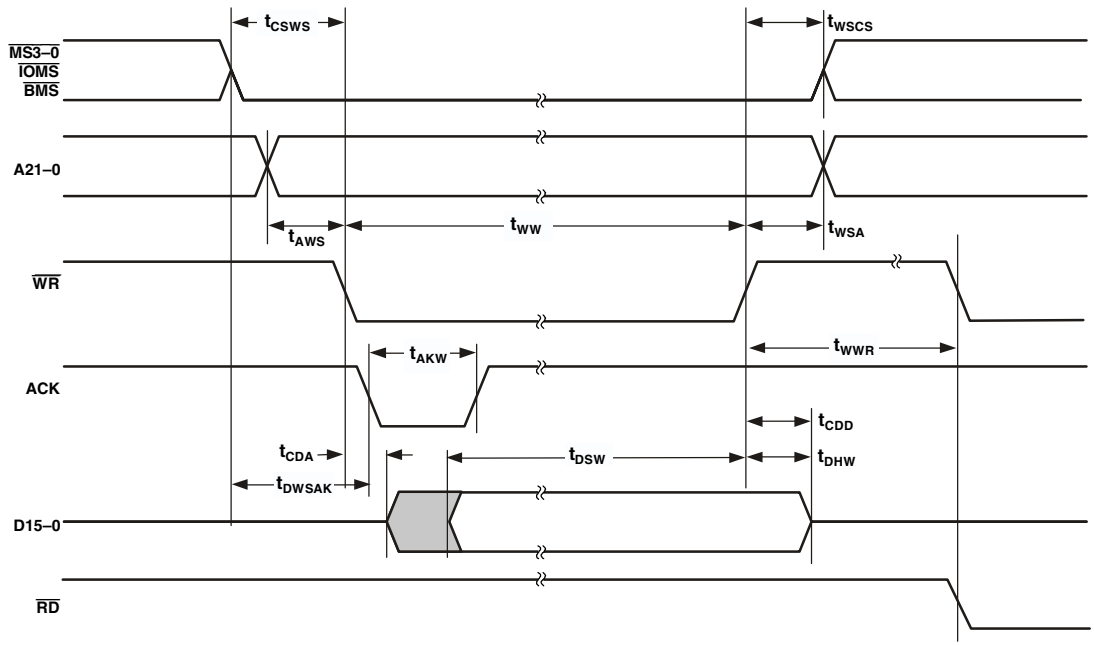

Figure 9. External Port Write Cycle Timing

External Port Read Cycle Timing

Table 9 and Figure 10 describe external port read operations. For additional information on the ACK signal, see the discussion on Page 26.

Table 9. External Port Read Cycle Timing

Parameter[1, 2]		Min	Max	Unit
Timing Requirements				
t_{AKW}	ACK Strobe Pulsewidth	t_{HCLK}		ns
t_{RDA}	$\overline{RD}$ Asserted to Data Access Setup		$t_{EMICLK}-5+W^3$	ns
t_{ADA}	Address Valid to Data Access Setup		$t_{EMICLK}+W^3$	ns
t_{SDA}	Chip Select Asserted to Data Access Setup		$t_{EMICLK}+W^3$	ns
t_{SD}	Data Valid to $\overline{RD}$ Deasserted Setup	5		ns
t_{HRD}	$\overline{RD}$ Deasserted to Data Invalid Hold	0		ns
t_{DRSAK}	ACK Delay from $\overline{XMS}$ Low		$0.5t_{EMICLK}-1$	ns
Switching Characteristics				
t_{CSRS}	Chip Select Asserted to $\overline{RD}$ Asserted Delay	$0.5t_{EMICLK}-3$		ns
t_{ARS}	Address Valid to $\overline{RD}$ Setup and Delay	$0.5t_{EMICLK}-3$		ns
t_{RSCS}	$\overline{RD}$ Deasserted to Chip Select Deasserted Setup	$0.5t_{EMICLK}-2$		ns
t_{RW}	$\overline{RD}$ Strobe Pulsewidth	$t_{EMICLK}-2+W^3$		ns
t_{RSA}	$\overline{RD}$ Deasserted to Address Invalid Setup	$0.5t_{HCLK}-2$		ns
t_{RWR}	$\overline{RD}$ Deasserted to $\overline{WR}$, $\overline{RD}$ Asserted	t_{HCLK}		ns

[1] t_{EMICLK} is the External Memory Interface clock period. t_{HCLK} is the peripheral clock period.
[2] These are timing parameters that are based on worst-case operating conditions.
[3] W = (number of wait states specified in wait register) × t_{EMICLK}.

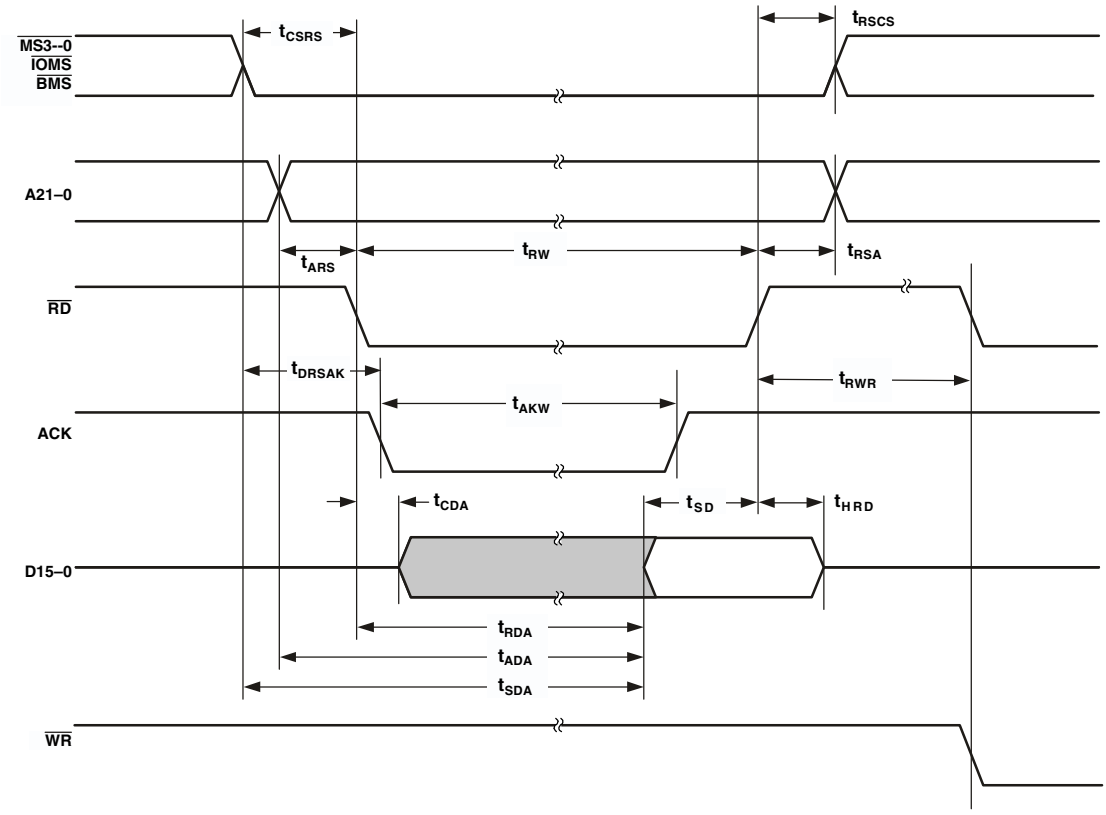

Figure 10. External Port Read Cycle Timing

Serial Port Timing

Table 11 and Figure 12 describe SPORT transmit and receive operations, while Figure 13 and Figure 14 describe SPORT Frame Sync operations.

Table 11. Serial Port[1, 2]

Parameter		Min	Max	Unit
External Clock Timing Requirements				
t_{SFSE}	TFS/RFS Setup Before TCLK/RCLK[3]	4		ns
t_{HFSE}	TFS/RFS Hold After TCLK/RCLK[3]	4		ns
t_{SDRE}	Receive Data Setup Before RCLK[3]	1.5		ns
t_{HDRE}	Receive Data Hold After RCLK[3]	4		ns
t_{SCLKW}	TCLK/RCLK Width	$0.5t_{HCLK}-1$		ns
t_{SCLK}	TCLK/RCLK Period	$2t_{HCLK}$		ns
Internal Clock Timing Requirements				
t_{SFSI}	TFS Setup Before TCLK[4]; RFS Setup Before RCLK[3]	4		ns
t_{HFSI}	TFS/RFS Hold After TCLK/RCLK[3]	3		ns
t_{SDRI}	Receive Data Setup Before RCLK[3]	2		ns
t_{HDRI}	Receive Data Hold After RCLK[3]	5		ns
External or Internal Clock Switching Characteristics				
t_{DFSE}	TFS/RFS Delay After TCLK/RCLK (Internally Generated FS)[4]		14	ns
t_{HOFSE}	TFS/RFS Hold After TCLK/RCLK (Internally Generated FS)[4]	3		ns
External Clock Switching Characteristics				
t_{DDTE}	Transmit Data Delay After TCLK[4]		13.4	ns
t_{HDTE}	Transmit Data Hold After TCLK[4]	4		ns
Internal Clock Switching Characteristics				
t_{DDTI}	Transmit Data Delay After TCLK[4]		13.4	ns
t_{HDTI}	Transmit Data Hold After TCLK[4]	4		ns
t_{SCLKIW}	TCLK/RCLK Width	$0.5t_{HCLK}-3.5$	$0.5t_{HCLK}+2.5$	ns
Enable and Three-State[5] Switching Characteristics				
t_{DTENE}	Data Enable from External TCLK[4]	0	12.1	ns
t_{DDTTE}	Data Disable from External TCLK[4]		13	ns
t_{DTENI}	Data Enable from Internal TCLK[4]	0	13	ns
t_{DDTTI}	Data Disable from External TCLK[4]		12	ns
External Late Frame Sync Switching Characteristics				
$t_{DDTLFSE}$	Data Delay from Late External TFS with MCE = 1, MFD = 0[6, 7]		10.5	ns
$t_{DTENLFSE}$	Data Enable from Late FS or MCE = 1, MFD = 0[6, 7]	3.5		ns

[1] To determine whether communication is possible between two devices at clock speed n, the following specifications must be confirmed: 1) frame sync delay and frame sync setup-and-hold, 2) data delay and data setup-and-hold, and 3) SCLK width.
[2] Word selected timing for I^2S mode is the same as TFS/RFS timing (normal framing only).
[3] Referenced to sample edge.
[4] Referenced to drive edge.
[5] Only applies to SPORT.
[6] MCE = 1, TFS enable, and TFS valid follow $t_{DDTENFS}$ and $t_{DDTLFSE}$.
[7] If external RFSD/TFS setup to RCLK/TCLK > $0.5t_{LSCK}$, $t_{DDTLSCK}$ and $t_{DTENLSCK}$ apply; otherwise, $t_{DDTLFSE}$ and $t_{DTENLFS}$ apply.

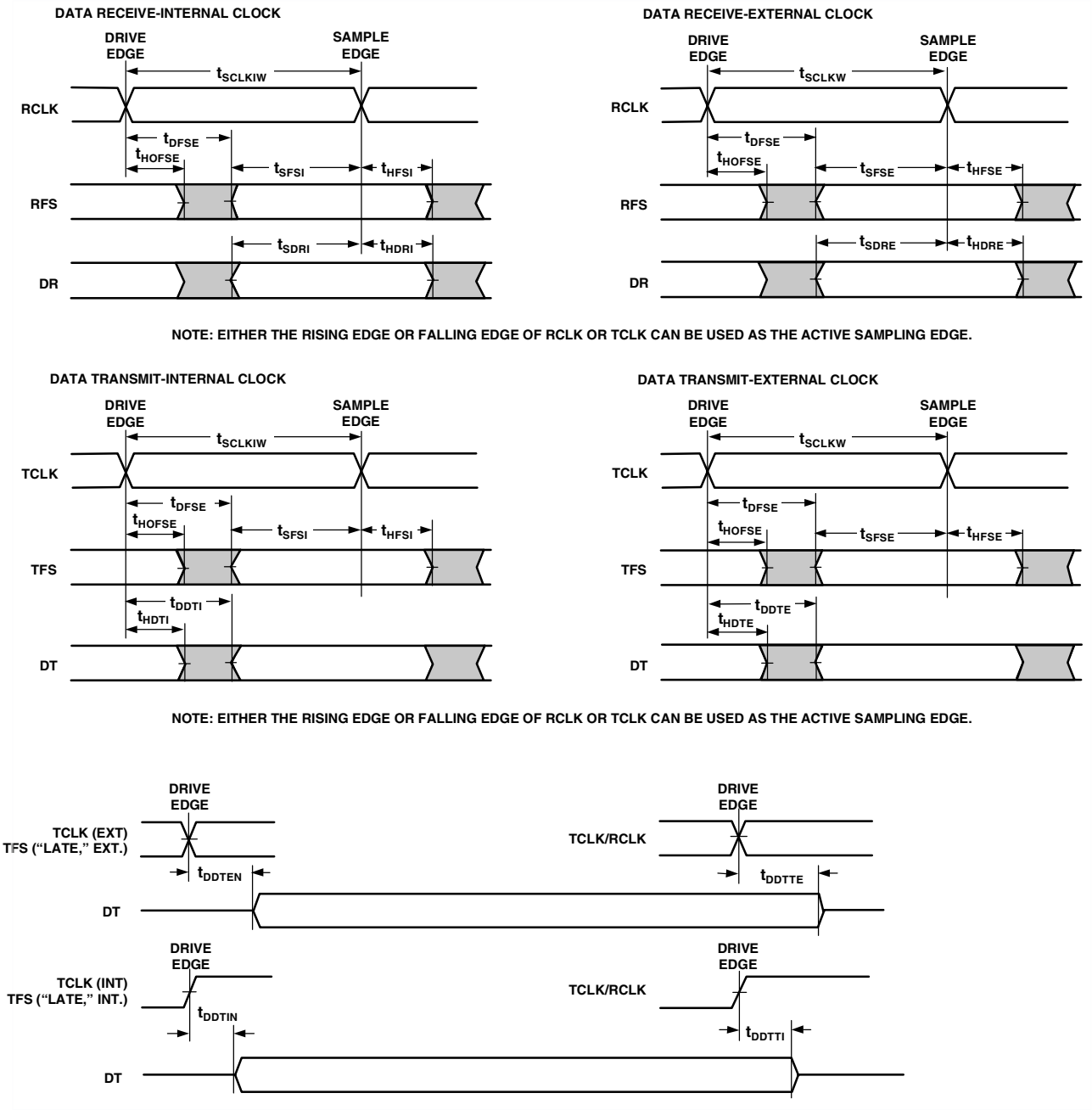

Figure 12. Serial Port

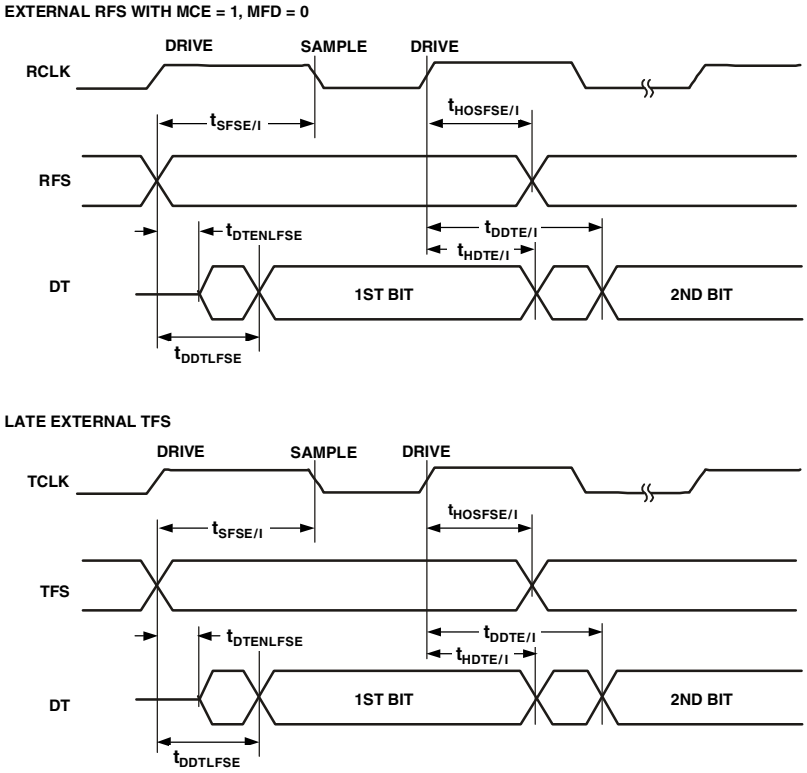

Figure 13. Serial Port—External Late Frame Sync (Frame Sync Setup > 0.5t_{SCLK})

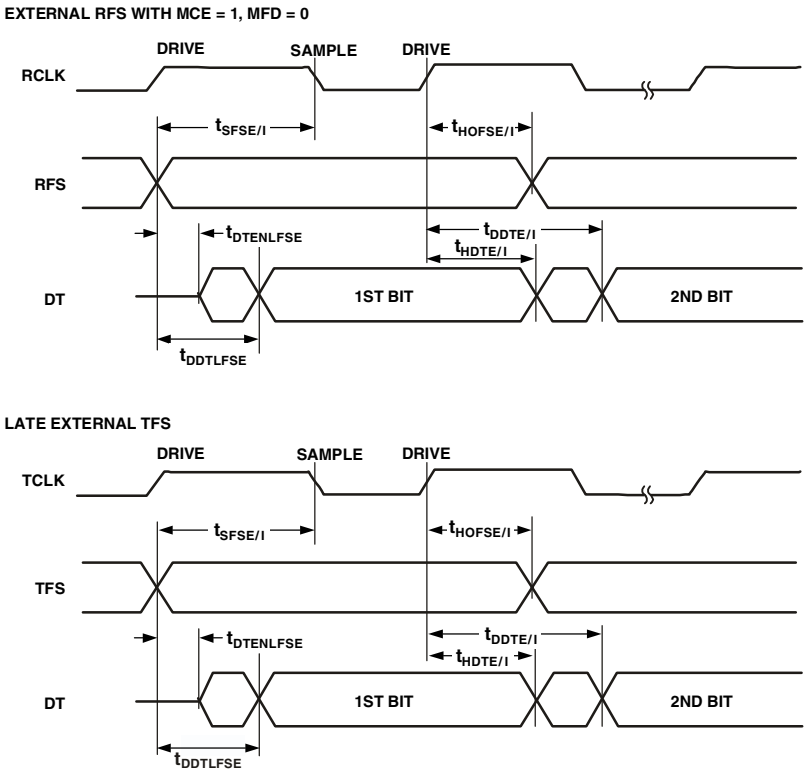

Figure 14. Serial Port—External Late Frame Sync (Frame Sync Setup < 0.5t_{HCLK})

Serial Peripheral Interface Port—Master Timing

Table 12 and Figure 15 describe SPI port master operations.

Table 12. Serial Peripheral Interface (SPI) Port—Master Timing

Parameter		Min	Max	Unit
Timing Requirements				
t_{SSPID}	Data Input Valid to SCLK Edge (Data Input Setup)	8		ns
t_{HSPID}	SCLK Sampling Edge to Data Input Invalid (Data In Hold)	1		ns
Switching Characteristics				
t_{SDSCIM}	$\overline{SPISEL}$ Low to First SCLK Edge	$2t_{HCLK}-3$		ns
t_{SPICHM}	Serial Clock High Period	$2t_{HCLK}-3$		ns
t_{SPICLM}	Serial Clock Low Period	$2t_{HCLK}-3$		ns
t_{SPICLK}	Serial Clock Period	$4t_{HCLK}-1$		ns
t_{HDSM}	Last SCLK Edge to $\overline{SPISEL}$ High	$2t_{HCLK}-3$		ns
t_{SPITDM}	Sequential Transfer Delay	$2t_{HCLK}-2$		ns
t_{DDSPID}	SCLK Edge to Data Output Valid (Data Out Delay)	0	6	ns
t_{HDSPID}	SCLK Edge to Data Output Invalid (Data Out Hold)	0	5	ns

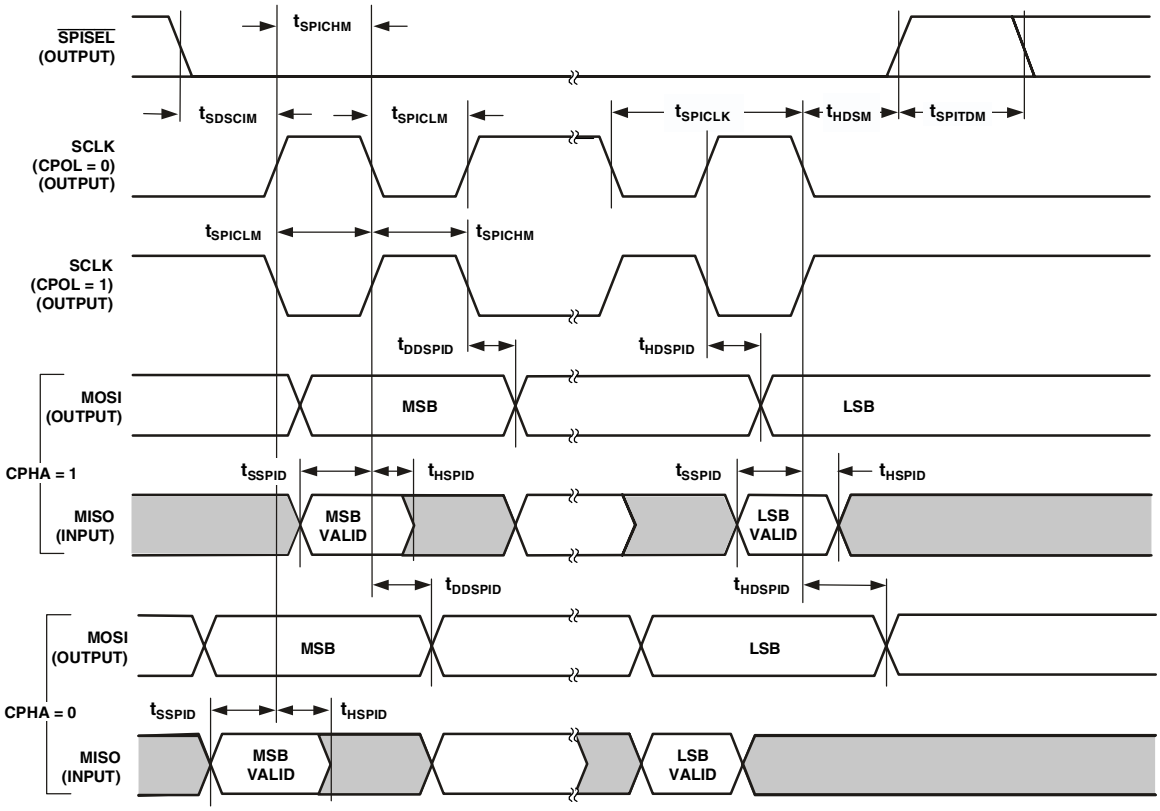

Figure 15. Serial Peripheral Interface (SPI) Port—Master Timing

Table 16. 196-Ball Mini-BGA Signal by Ball Number

Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal
A1	nc	D8	AVSS	H1	A10	L8	VDDINT
A2	DR	D9	PF3/SPISEL3	H2	A11	L9	VDDEXT
A3	DT	D10	$\overline{\text{AUXTRIP}}$	H3	$\overline{\text{MS3}}$	L10	VDDEXT
A4	RFS	D11	VDDEXT	H4	GND	L11	GND
A5	VIN4	D12	AUX1	H5	nc	L12	BMODE2
A6	BSHAN	D13	AUX0	H6	nc	L13	BMODE1
A7	VIN0	D14	PF15	H7	nc	L14	CLKIN
A8	VIN1	E1	A16	H8	nc	M1	A2
A9	VIN3	E2	A17	H9	nc	M2	A3
A10	PF0/$\overline{\text{SPISS}}$	E3	$\overline{\text{WR}}$	H10	nc	M3	$\overline{\text{MS2}}$
A11	PF4/SPISEL4	E4	GND	H11	VDDEXT	M4	GND
A12	PF6/SPISEL6	E5	VDDEXT	H12	TMR0	M5	VDDEXT
A13	PF7/SPISEL7	E6	nc	H13	$\overline{\text{POR}}$	M6	GND
A14	nc	E7	nc	H14	$\overline{\text{RESET}}$	M7	VDDEXT
B1	SCK	E8	nc	J1	A8	M8	nc
B2	RCLK	E9	nc	J2	A9	M9	CL
B3	TCLK	E10	nc	J3	$\overline{\text{BMS}}$	M10	AL
B4	TFS	E11	GND	J4	VDDEXT	M11	PWMPOL
B5	VIN6	E12	EIA	J5	nc	M12	$\overline{\text{PWMTRIP}}$
B6	ASHAN	E13	EIB	J6	nc	M13	BYPASS
B7	VIN2	E14	EIS	J7	nc	M14	BMODE0
B8	SENSE	F1	A14	J8	nc	N1	A0
B9	CAPB	F2	A15	J9	nc	N2	A1
B10	PF1/SPISEL1	F3	$\overline{\text{BG}}$	J10	nc	N3	D13
B11	PF5/SPISEL5	F4	GND	J11	GND	N4	D11
B12	PF8	F5	nc	J12	TMS	N5	D9
B13	PF9	F6	nc	J13	TCK	N6	D7
B14	PF13	F7	nc	J14	TDI	N7	D5
C1	$\overline{\text{BR}}$	F8	nc	K1	A6	N8	D3
C2	$\overline{\text{RD}}$	F9	nc	K2	A7	N9	D1
C3	MISO	F10	nc	K3	$\overline{\text{MS0}}$	N10	CH
C4	MOSI	F11	VDDINT	K4	GND	N11	AH
C5	VIN7	F12	EIZ	K5	GND	N12	nc
C6	VIN5	F13	TMR2	K6	GND	N13	PWMSYNC
C7	CAPT	F14	XTAL	K7	GND	N14	$\overline{\text{PWMSR}}$
C8	VREF	G1	A12	K8	GND	P1	nc
C9	CML	G2	A13	K9	GND	P2	D15
C10	PF2/SPISEL2	G3	$\overline{\text{BGH}}$	K10	GND	P3	D14
C11	PF10	G4	VDDINT	K11	VDDINT	P4	D12
C12	PF11	G5	nc	K12	$\overline{\text{EMU}}$	P5	D10
C13	PF12	G6	nc	K13	$\overline{\text{TRST}}$	P6	D8
C14	PF14	G7	nc	K14	TDO	P7	D6
D1	A18	G8	nc	L1	A4	P8	D4
D2	A19	G9	nc	L2	A5	P9	D2
D3	$\overline{\text{IOMS}}$	G10	nc	L3	$\overline{\text{MS1}}$	P10	D0
D4	ACK	G11	GND	L4	VDDEXT	P11	BL
D5	AVDD	G12	TMR1	L5	VDDINT	P12	BH
D6	AVDD	G13	CONVST	L6	VDDEXT	P13	nc
D7	AVSS	G14	CLKOUT	L7	VDDINT	P14	nc

Table 17. 196-Ball Mini-BGA Ball Number by Signal

Signal	Ball No.	Signal	Ball No.	Signal	Ball No.	Signal	Ball No.
A0	N1	CONVST	G13	nc	E6	PF15	D14
A1	N2	D0	P10	nc	E7	$\overline{\text{POR}}$	H13
A2	M1	D1	N9	nc	E8	PWMPOL	M11
A3	M2	D2	P9	nc	E9	PWMSYNC	N13
A4	L1	D3	N8	nc	E10	$\overline{\text{PWMSR}}$	N14
A5	L2	D4	P8	nc	F5	$\overline{\text{PWMTRIP}}$	M12
A6	K1	D5	N7	nc	F6	RCLK	B2
A7	K2	D6	P7	nc	F7	$\overline{\text{RD}}$	C2
A8	J1	D7	N6	nc	F8	$\overline{\text{RESET}}$	H14
A9	J2	D8	P6	nc	F9	RFS	A4
A10	H1	D9	N5	nc	F10	SCK	B1
A11	H2	D10	P5	nc	G5	SENSE	B8
A12	G1	D11	N4	nc	G6	TCK	J13
A13	G2	D12	P4	nc	G7	TCLK	B3
A14	F1	D13	N3	nc	G8	TDI	J14
A15	F2	D14	P3	nc	G9	TDO	K14
A16	E1	D15	P2	nc	G10	TFS	B4
A17	E2	DR	A2	nc	H5	TMR0	H12
A18	D1	DT	A3	nc	H6	TMR1	G12
A19	D2	EIA	E12	nc	H7	TMR2	F13
ACK	D4	EIB	E13	nc	H8	TMS	J12
AH	N11	EIS	E14	nc	H9	$\overline{\text{TRST}}$	K13
AL	M10	EIZ	F12	nc	H10	VDDEXT	D11
ASHAN	B6	$\overline{\text{EMU}}$	K12	nc	J5	VDDEXT	E5
$\overline{\text{AUXTRIP}}$	D10	GND	E4	nc	J6	VDDEXT	H11
AUX1	D12	GND	E11	nc	J7	VDDEXT	J4
AUX0	D13	GND	F4	nc	J8	VDDEXT	L4
AVDD	D5	GND	G11	nc	J9	VDDEXT	L6
AVDD	D6	GND	H4	nc	J10	VDDEXT	L9
AVSS	D7	GND	J11	nc	M8	VDDEXT	L10
AVSS	D8	GND	K4	nc	N12	VDDEXT	M5
$\overline{\text{BG}}$	F3	GND	K5	nc	P1	VDDEXT	M7
$\overline{\text{BGH}}$	G3	GND	K6	nc	P13	VDDINT	G4
BL	P11	GND	K7	nc	P14	VDDINT	L5
BH	P12	GND	K8	PF0/$\overline{\text{SPISS}}$	A10	VDDINT	L7
BMODE0	M14	GND	K9	PF1/SPISEL1	B10	VDDINT	L8
BMODE1	L13	GND	K10	PF2/SPISEL2	C10	VDDINT	K11
BMODE2	L12	GND	L11	PF3/SPISEL3	D9	VDDINT	F11
$\overline{\text{BMS}}$	J3	GND	M4	PF4/SPISEL4	A11	VIN0	A7
$\overline{\text{BR}}$	C1	GND	M6	PF5/SPISEL5	B11	VIN1	A8
BSHAN	A6	$\overline{\text{IOMS}}$	D3	PF6/SPISEL6	A12	VIN2	B7
BYPASS	M13	MISO	C3	PF7/SPISEL7	A13	VIN3	A9
CAPB	B9	MOSI	C4	PF8	B12	VIN4	A5
CAPT	C7	$\overline{\text{MS0}}$	K3	PF9	B13	VIN5	C6
CH	N10	$\overline{\text{MS1}}$	L3	PF10	C11	VIN6	B5
CL	M9	$\overline{\text{MS2}}$	M3	PF11	C12	VIN7	C5
CLKIN	L14	$\overline{\text{MS3}}$	H3	PF12	C13	VREF	C8
CLKOUT	G14	nc	A1	PF13	B14	$\overline{\text{WR}}$	E3
CML	C9	nc	A14	PF14	C14	XTAL	F14

Table 18. 176-Lead LQFP Signal by Lead Number

Lead No.	Signal	Lead No.	Signal	Lead No.	Signal	Lead No.	Signal
1	nc	45	VDDEXT	89	nc	133	VDDEXT
2	nc	46	A4	90	nc	134	PF11
3	VDDEXT	47	A3	91	VDDEXT	135	PF10
4	RCLK	48	A2	92	BYPASS	136	PF9
5	SCK	49	A1	93	BMODE0	137	PF8
6	MISO	50	A0	94	BMODE1	138	PF7/SPISEL7
7	MOSI	51	D15	95	BMODE2	139	PF6/SPISEL6
8	$\overline{RD}$	52	D14	96	nc	140	PF5/SPISEL5
9	$\overline{WR}$	53	D13	97	GND	141	PF4/SPISEL4
10	ACK	54	D12	98	VDDINT	142	GND
11	$\overline{BR}$	55	D11	99	$\overline{EMU}$	143	VDDEXT
12	$\overline{BG}$	56	GND	100	$\overline{TRST}$	144	PF3/SPISEL3
13	$\overline{BGH}$	57	VDDEXT	101	TDO	145	PF2/SPISEL2
14	$\overline{IOMS}$	58	GND	102	TDI	146	PF1/SPISEL1
15	$\overline{BMS}$	59	VDDINT	103	TMS	147	PF0/$\overline{SPISS}$
16	$\overline{MS3}$	60	D10	104	TCK	148	GND
17	GND	61	D9	105	$\overline{POR}$	149	VDDINT
18	VDDEXT	62	D8	106	$\overline{RESET}$	150	AVSS
19	$\overline{MS2}$	63	D7	107	CLKIN	151	AVDD
20	$\overline{MS1}$	64	D6	108	XTAL	152	nc
21	$\overline{MS0}$	65	D5	109	CLKOUT	153	VREF
22	GND	66	GND	110	CONVST	154	CML
23	VDDINT	67	VDDINT	111	TMR0	155	CAPT
24	A19	68	D4	112	GND	156	CAPB
25	A18	69	D3	113	VDDEXT	157	SENSE
26	A17	70	D2	114	TMR1	158	VIN3
27	A16	71	D1	115	TMR2	159	VIN2
28	A15	72	D0	116	EIS	160	VIN1
29	A14	73	nc	117	GND	161	VIN0
30	A13	74	GND	118	VDDINT	162	ASHAN
31	GND	75	VDDEXT	119	EIZ	163	BSHAN
32	VDDEXT	76	CL	120	EIB	164	VIN4
33	A12	77	CH	121	EIA	165	VIN5
34	A11	78	BL	122	$\overline{AUXTRIP}$	166	VIN6
35	A10	79	BH	123	AUX1	167	VIN7
36	A9	80	AL	124	AUX0	168	AVSS
37	A8	81	AH	125	PF15	169	AVDD
38	A7	82	nc	126	PF14	170	DT
39	A6	83	nc	127	PF13	171	DR
40	A5	84	PWMSYNC	128	PF12	172	RFS
41	GND	85	PWMPOL	129	GND	173	TFS
42	nc	86	$\overline{PWMSR}$	130	nc	174	TCLK
43	nc	87	$\overline{PWMTRIP}$	131	nc	175	GND
44	nc	88	GND	132	nc	176	nc