



Clarification to the Serial I/O Control Register Description for the DSP1620/27/28/29 Devices

Active Clock Frequency

The purpose of this advisory is to clarify the function of the serial I/O control registers in the DSP1620/27/28/29 devices. Specifically, it clarifies the function of the control register field that specifies the active clock frequency. The device data sheets state that the active clock frequency is a ratio of the **input** clock frequency on the CKI pin (DSP1627/28/29 devices) or the output clock frequency on the CKO pin (DSP1620 device). For all four devices, the actual active clock frequency is a ratio of the **internal** clock frequency, which can be programmed as either the input clock frequency on the CKI pin or the output of an internal clock synthesizer (PLL).

Table 1 summarizes information for each of the four devices. It lists the document number for each device data sheet. For example, the data sheet for the DSP1620, entitled *DSP1620 Digital Signal Processor*, has the document number DS97-321WDSP. Table 1 also lists the name of each serial I/O unit on each device, the corresponding control register, the data sheet page number that describes the register, and the corresponding field within the register that specifies the active clock frequency. For example, the DSP1620 contains two serial I/O units named SIO and SSIO. The control register for SIO is **sioc** described on page 94 of the data sheet. Bits 8—7 within **sioc** (CLK1 field) specify the active clock frequency of the SIO.

Table 1. Data Sheet and Serial I/O Information for the DSP1620/27/28/29 Devices

| Device | Data Sheet Document Number | Serial I/O Units | | | | |
|---------|----------------------------|------------------|------------------|---------------------|--------------------------------------|------|
| | | Name | Control Register | Data Sheet Page No. | Active Clock Frequency Control Field | |
| | | | | | Bits | Name |
| DSP1620 | DS97-321WDSP | SIO | sioc | 94 | 8—7 | CLK1 |
| | | SSIO | SSIOC | 96 | 8—7 | CLK2 |
| DSP1627 | DS96-188WDSP | SIO | sioc | 45 | 8—7 | CLK |
| | | SIO2 | | | | |
| DSP1628 | DS97-040WDSP | SIO | sioc | 55 | 8—7 | CLK |
| | | SIO2 | | | | |
| DSP1629 | DS96-039WDSP | SIO | sioc | 46 | 8—7 | CLK |
| | | SIO2 | | | | |

Table 2 shows a corrected description of the CLK/CLK1/CLK2 field of the serial I/O control register. The specific correction is shown in bold type—the active clock frequency is a ratio of **f_{internal clock}**, not of CKI or CKO.

Table 2. Corrected Description of CLK/CLK1/CLK2 Field

| Field | Value | Description |
|-------|-------|---|
| CLK | 00 | Active clock frequency = f_{internal clock} ÷ 2 |
| CLK1 | 01 | Active clock frequency = f_{internal clock} ÷ 6 |
| CLK2 | 10 | Active clock frequency = f_{internal clock} ÷ 8 |
| | 11 | Active clock frequency = f_{internal clock} ÷ 10 |

For additional information, contact your Microelectronics Group Account Manager or the following:

INTERNET: <http://www.lucent.com/micro>

E-MAIL: docmaster@micro.lucent.com

N. AMERICA: Microelectronics Group, Lucent Technologies Inc., 555 Union Boulevard, Room 30L-15P-BA, Allentown, PA 18103

1-800-372-2447, FAX 610-712-4106 (In CANADA: **1-800-553-2448**, FAX 610-712-4106)

ASIA PACIFIC: Microelectronics Group, Lucent Technologies Singapore Pte. Ltd., 77 Science Park Drive, #03-18 Cintech III, Singapore 118256

Tel. (65) 778 8833, FAX (65) 777 7495

CHINA: Microelectronics Group, Lucent Technologies (China) Co., Ltd., A-F2, 23/F, Zao Fong Universe Building, 1800 Zhong Shan Xi Road, Shanghai 200233 P. R. China **Tel. (86) 21 6440 0468, ext. 316**, FAX (86) 21 6440 0652

JAPAN: Microelectronics Group, Lucent Technologies Japan Ltd., 7-18, Higashi-Gotanda 2-chome, Shinagawa-ku, Tokyo 141, Japan

Tel. (81) 3 5421 1600, FAX (81) 3 5421 1700

EUROPE: Data Requests: MICROELECTRONICS GROUP DATALINE: **Tel. (44) 1189 324 299**, FAX (44) 1189 328 148

Technical Inquiries: GERMANY: **(49) 89 95086 0** (Munich), UNITED KINGDOM: **(44) 1344 865 900** (Ascot),

FRANCE: **(33) 1 40 83 68 00** (Paris), SWEDEN: **(46) 8 594 607 00** (Stockholm), FINLAND: **(358) 9 4354 2800** (Helsinki),

ITALY: **(39) 02 6608131** (Milan), SPAIN: **(34) 1 807 1441** (Madrid)

Lucent Technologies Inc. reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed as a result of their use or application. No rights under any patent accompany the sale of any such product(s) or information.

Copyright © 1999 Lucent Technologies Inc.
All Rights Reserved

May 1999
AY99-001WDSP
(must accompany DS97-321WDSP, DS96-188WDSP, DS97-040WDSP, and DS96-039WDSP)

microelectronics group

Lucent Technologies
Bell Labs Innovations





DSP1628 Digital Signal Processor

1 Features

- Optimized for digital cellular applications with a bit manipulation unit for higher coding efficiency and an error correction coprocessor for equalization and channel coding support.
- On-chip, programmable, PLL clock synthesizer.
- 19.2 ns and 12.5 ns instruction cycle times at 2.7 V.
- Mask-programmable memory map option: The DSP1628x16 features 16 Kwords on-chip dual-port RAM. The DSP1628x08 features 8 Kwords on-chip dual-port RAM. Both feature 48 Kwords on-chip ROM with a secure option.
- Low power consumption:
 - <1.9 mW/MIPS typical at 2.7 V.
- Flexible power management modes:
 - Standard sleep: 0.2 mW/MIPS at 2.7 V.
 - Sleep with slow internal clock: 0.7 mW at 2.7 V.
 - Hardware STOP (pin halts DSP): <20 μ A.
- Mask-programmable clock options: small signal, and CMOS.
- 144 PBGA package (13 mm x 13 mm) available.
- Sequenced accesses to X and Y external memory.
- Object code compatible with the DSP1618.
- Single-cycle squaring.
- 16 x 16-bit multiplication and 36-bit accumulation in one instruction cycle.
- Instruction cache for high-speed, program-efficient, zero-overhead looping.
- Dual 25 Mbit/s serial I/O ports with multiprocessor capability—16-bit data channel, 8-bit protocol channel.
- 8-bit parallel host interface
 - Supports 8- or 16-bit transfers.
 - *Motorola*^{*} or *Intel*[†] compatible.
- 8-bit control I/O interface.
- 256 memory-mapped I/O ports.
- *IEEE*[‡] P1149.1 test port (JTAG boundary scan).
- Full-speed in-circuit emulation hardware development system on-chip.
- Supported by DSP1628 software and hardware development tools.

2 Description

The DSP1628 digital signal processor offers 80 MIPS and 52 MIPS operation at 2.7 V. Designed specifically for applications requiring low power dissipation in digital cellular systems, the DSP1628 is a signal-coding device that can be programmed to perform a wide variety of fixed-point signal processing functions. The device is based on the DSP1600 core with a bit manipulation unit for enhanced signal coding efficiency, an external memory sequencer, an error correction coprocessor (ECCP) for more efficient Viterbi decoding, and an 8-bit parallel host interface for hardware flexibility. The DSP1628 includes a mix of peripherals specifically intended to support processing-intensive but cost-sensitive applications in the area of digital wireless communications.

The DSP1628x16 contains 16 Kwords of internal dual-port RAM (DPRAM), which allows simultaneous access to two RAM locations in a single instruction cycle. The DSP1628x08 supports the use of 8 Kwords of DPRAM. Both devices contain 48 Kwords of internal ROM (IROM).

The DSP1628 is object code compatible with the DSP1618, while providing more memory. The DSP1628 is pin compatible with the DSP1627. Note that TRST (JTAG test reset), replaces a VDD pin.

The DSP1628 supports 2.7 V operation with flexible power management modes required for portable cellular terminals. Several control mechanisms achieve low-power operation, including a STOP pin for placing the DSP into a fully static, halted state and a programmable power control register used to power down unused on-chip I/O units. These power management modes allow for trade-offs between power reduction and wake-up latency requirements. During system standby, power consumption is reduced to less than 20 μ A.

The on-chip clock synthesizer can be driven by an external clock whose frequency is a fraction of the instruction rate.

The device is packaged in a 144-pin PBGA, a 100-pin BQFP, or a 100-pin TQFP and is available with 19.2 ns and 12.5 ns instruction cycle times at 2.7 V.

* *Motorola* is a registered trademark of Motorola, Inc.

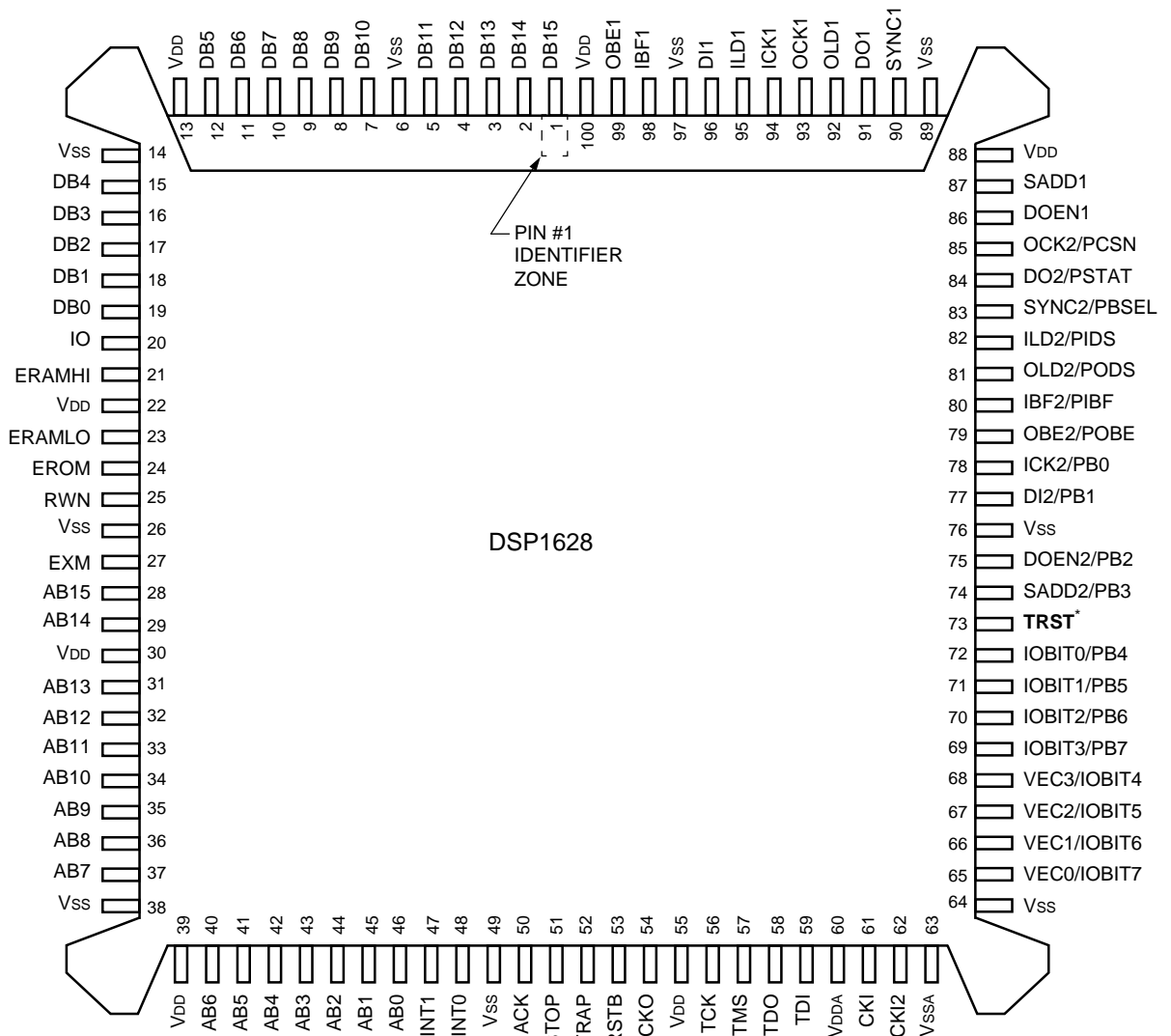
† *Intel* is a registered trademark of Intel Corporation.

‡ *IEEE* is a registered trademark of The Institute of Electrical and Electronics Engineers, Inc.

Table of Contents

| Contents | Page | Contents | Page |
|--|-------------|--|-------------|
| 1 Features | 1 | 10 Timing Characteristics for 2.7 V Operation..... | 86 |
| 2 Description | 1 | 10.1 DSP Clock Generation | 87 |
| 3 Pin Information | 3 | 10.2 Reset Circuit..... | 88 |
| 4 Hardware Architecture..... | 8 | 10.3 Reset Synchronization | 89 |
| 4.1 DSP1628 Architectural Overview | 8 | 10.4 JTAG I/O Specifications | 90 |
| 4.2 DSP1600 Core Architectural Overview | 12 | 10.5 Interrupt..... | 91 |
| 4.3 Interrupts and Trap..... | 13 | 10.6 Bit Input/Output (BIO)..... | 92 |
| 4.4 Memory Maps and Wait-States | 18 | 10.7 External Memory Interface | 93 |
| 4.5 External Memory Interface (EMI) | 21 | 10.8 PHIF Specifications | 97 |
| 4.6 Bit Manipulation Unit (BMU)..... | 22 | 10.9 Serial I/O Specifications | 103 |
| 4.7 Serial I/O Units (SIOs)..... | 22 | 10.10 Multiprocessor Communication | 108 |
| 4.8 Parallel Host Interface (PHIF) | 24 | 11 Outline Diagrams..... | 109 |
| 4.9 Bit Input/Output Unit (BIO) | 25 | 11.1 100-Pin BQFP (Bumpered Quad | |
| 4.10 Timer | 26 | Flat Pack) | 109 |
| 4.11 Error Correction Coprocessor (ECCP)..... | 26 | 11.2 100-Pin TQFP (Thin Quad Flat Pack) | 110 |
| 4.12 JTAG Test Port | 34 | 11.3 144-Pin PBGA (Plastic Ball Grid Array)..... | 111 |
| 4.13 Clock Synthesis..... | 36 | | |
| 4.14 Power Management | 39 | | |
| 5 Software Architecture | 46 | | |
| 5.1 Instruction Set | 46 | | |
| 5.2 Register Settings..... | 55 | | |
| 5.3 Instruction Set Formats | 66 | | |
| 6 Signal Descriptions..... | 72 | | |
| 6.1 System Interface | 72 | | |
| 6.2 External Memory Interface | 74 | | |
| 6.3 Serial Interface #1 | 75 | | |
| 6.4 Parallel Host Interface or Serial | | | |
| Interface #2 and Control I/O Interface..... | 76 | | |
| 6.5 Control I/O Interface..... | 76 | | |
| 6.6 JTAG Test Interface | 77 | | |
| 7 Mask-Programmable Options..... | 78 | | |
| 7.1 Input Clock Options..... | 78 | | |
| 7.2 Memory Map Options..... | 78 | | |
| 7.3 ROM Security Options | 78 | | |
| 8 Device Characteristics..... | 79 | | |
| 8.1 Absolute Maximum Ratings | 79 | | |
| 8.2 Handling Precautions | 79 | | |
| 8.3 Recommended Operating Conditions | 79 | | |
| 8.4 Package Thermal Considerations | 80 | | |
| 9 Electrical Characteristics and Requirements..... | 81 | | |
| 9.1 Power Dissipation | 84 | | |

3 Pin Information

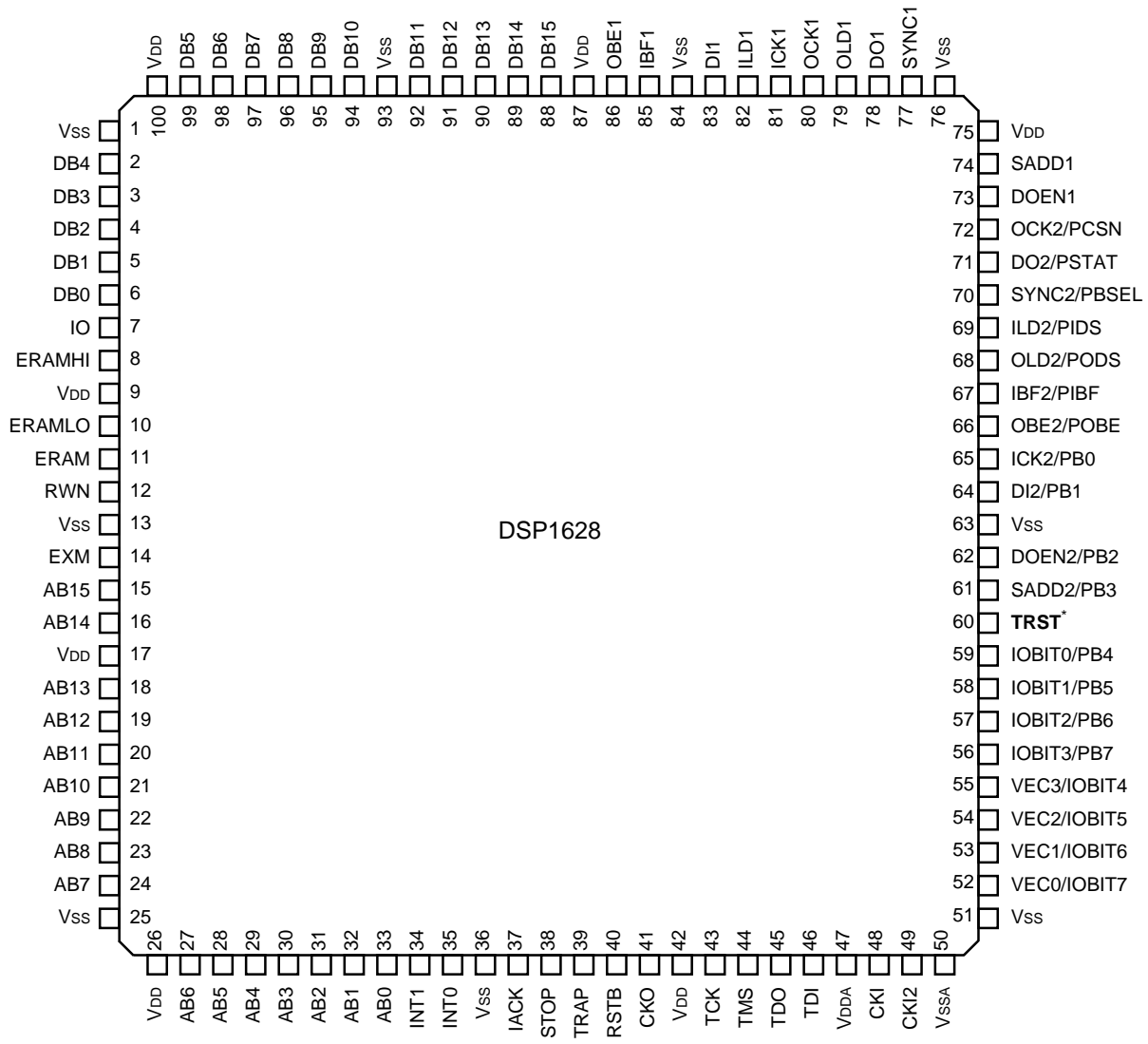


5-4218 (F).c

* Note the difference from the DSP1627 pinout.

Figure 1. DSP1628 BQFP Pin Diagram

3 Pin Information (continued)

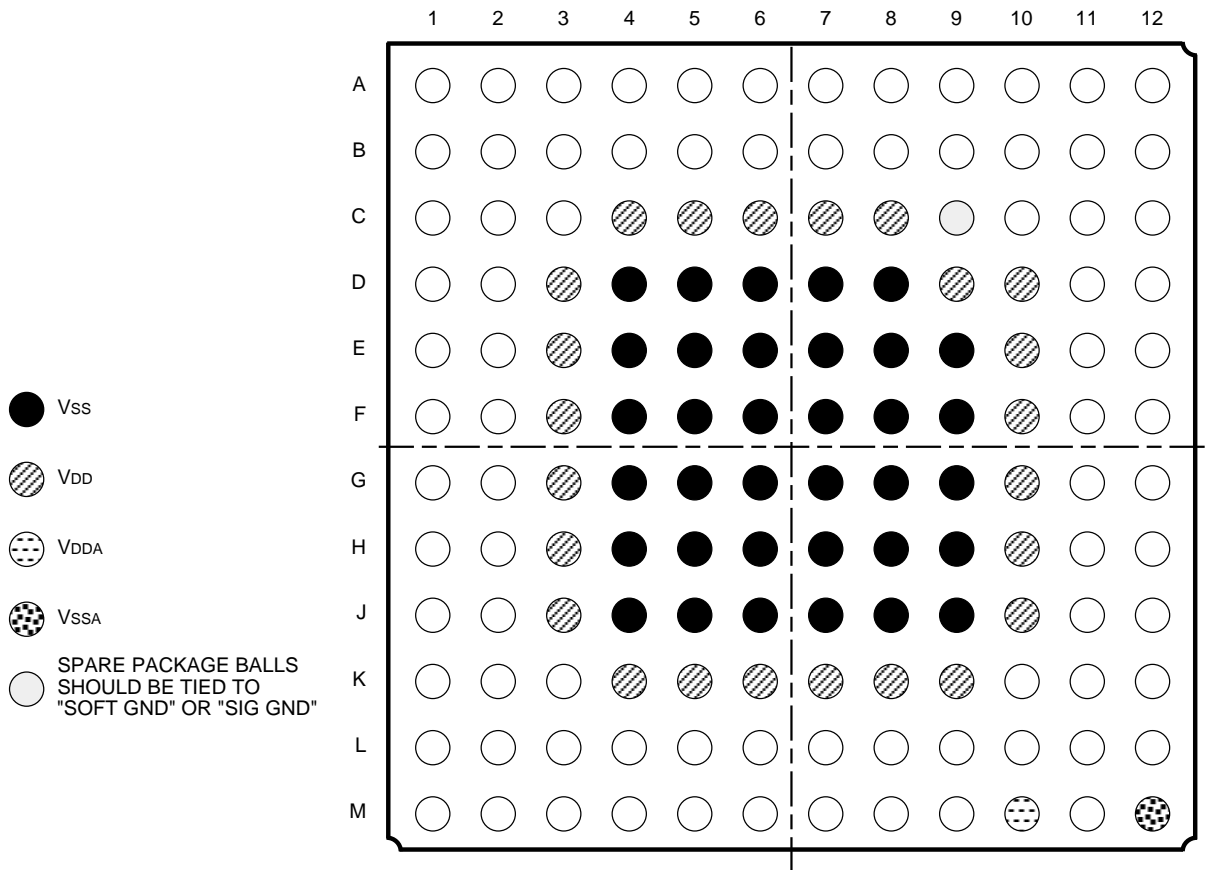


5-4219 (F).c

* Note the difference from the DSP1627 pinout.

Figure 2. DSP1628 TQFP Pin Diagram

3 Pin Information (continued)



5-5224 (C)

Note: Solder balls viewed thru package.

Figure 3. 144-Pin Plastic Ball Grid Array (Top View)

3 Pin Information (continued)

Functional descriptions of pins 1—100 are found in Section 6, Signal Descriptions. The functionality of CKI and CKI2 pins are mask-programmable (see Section 7, Mask-Programmable Options). Input levels on all I and I/O type pins are designed to remain at full CMOS levels when not driven by the DSP.

Table 1. Pin Descriptions

| PBGA Pin | BQFP Pin | TQFP Pin | Symbol | Type | Name/Function | |
|--|--|--|-------------|------|---|---------------------|
| B6, A6, B5, A5, B4, A4, B3, A3, B2, A2, A1, B1, C2, C1, C3, D1 | 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 15, 16, 17, 18, 19 | 88, 89, 90, 91, 92, 94, 95, 96, 97, 98, 99, 2, 3, 4, 5, 6 | DB[15:0] | I/O* | External Memory Data Bus 15—0. | |
| D2 | 20 | 7 | IO | O† | Data Address 0x4000 to 0x40FF I/O Enable. | |
| E1 | 21 | 8 | ERAMHI | O† | Data Address 0x8000 to 0xFFFF External RAM Enable. | |
| E2 | 23 | 10 | ERAMLO | O† | Data Address 0x4100 to 0x7FFF External RAM Enable. | |
| F1 | 24 | 11 | EROM | O† | Program Address External ROM Enable. | |
| F2 | 25 | 12 | RWN | O† | Read/Write Not. | |
| G1 | 27 | 14 | EXM | I | External ROM Enable. | |
| G2, H1, H2, J1, J2, K1, K2, L1, L2, M1, K3, M2, L3, M3, L4, M4 | 28, 29, 31, 32, 33, 34, 35, 36, 37, 40, 41, 42, 43, 44, 45, 46 | 15, 16, 18, 19, 20, 21, 22, 23, 24, 27, 28, 29, 30, 31, 32, 33 | AB[15:0] | O* | External Memory Address Bus 15—0. | |
| L5 | 47 | 34 | INT1 | I | Vectored Interrupt 1. | |
| M5 | 48 | 35 | INT0 | I | Vectored Interrupt 0. | |
| L6 | 50 | 37 | IACK | O* | Interrupt Acknowledge. | |
| M6 | 51 | 38 | STOP | I | STOP Input Clock. | |
| L7 | 52 | 39 | TRAP | I/O* | Nonmaskable Program Trap/Breakpoint Indication. | |
| M7 | 53 | 40 | RSTB | I | Reset Bar. | |
| L8 | 54 | 41 | CKO | O† | Processor Clock Output. | |
| M8 | 56 | 43 | TCK | I | JTAG Test Clock. | |
| L9 | 57 | 44 | TMS | I‡ | JTAG Test Mode Select. | |
| M9 | 58 | 45 | TDO | O§ | JTAG Test Data Output. | |
| L10 | 59 | 46 | TDI | I‡ | JTAG Test Data Input. | |
| | | | | | Mask-Programmable Input Clock Option | |
| | | | | | CMOS | Small Signal |
| L11 | 61 | 48 | CKI** | I | CKI | VAC |
| M11 | 62 | 49 | CKI2** | I | VSSA | VCM |
| K10 | 65 | 52 | VEC0/IOBIT7 | I/O* | Vectored Interrupt Indication 0/Status/Control Bit 7. | |
| L12 | 66 | 53 | VEC1/IOBIT6 | I/O* | Vectored Interrupt Indication 1/Status/Control Bit 6. | |
| K11 | 67 | 54 | VEC2/IOBIT5 | I/O* | Vectored Interrupt Indication 2/Status/Control Bit 5. | |
| K12 | 68 | 55 | VEC3/IOBIT4 | I/O* | Vectored Interrupt Indication 3/Status/Control Bit 4. | |
| J11 | 69 | 56 | IOBIT3/PB7 | I/O* | Status/Control Bit 3/PHIF Data Bus Bit 7. | |
| J12 | 70 | 57 | IOBIT2/PB6 | I/O* | Status/Control Bit 2/PHIF Data Bus Bit 6. | |

* 3-states when RSTB = 0, or by JTAG control.

† 3-states when RSTB = 0 and INT0 = 1. Output = 1 when RSTB = 0 and INT0 = 0, except CKO which is free-running.

‡ Pull-up devices on input.

§ 3-states by JTAG control.

** See Section 7, Mask-Programmable Options.

†† For SIO multiprocessor applications, add 5 kΩ external pull-up resistors to SADD1 and/or SADD2 for proper initialization.

3 Pin Information (continued)

Functional descriptions of pins 1—100 are found in Section 6, Signal Descriptions.

Table 1. Pin Descriptions (continued)

| PBGA Pin | BQFP Pin | TQFP Pin | Symbol | Type | Name/Function |
|--|-----------------------------------|-----------------------------------|-------------|------|---|
| H11 | 71 | 58 | IOBIT1/PB5 | I/O* | Status/Control Bit 1/PHIF Data Bus Bit 5. |
| H12 | 72 | 59 | IOBIT0/PB4 | I/O* | Status/Control Bit 0/PHIF Data Bus Bit 4. |
| G11 | 73 | 60 | TRST | ‡ | JTAG Test Reset. |
| G12 | 74 | 61 | SADD2/PB3†† | I/O* | SIO2 Multiprocessor Address/PHIF Data Bus Bit 3. |
| F11 | 75 | 62 | DOEN2/PB2 | I/O* | SIO2 Data Output Enable/PHIF Data Bus Bit 2. |
| F12 | 77 | 64 | DI2/PB1 | I/O* | SIO2 Data Input/PHIF Data Bus Bit 1. |
| E11 | 78 | 65 | ICK2/PB0 | I/O* | SIO2 Input Clock/PHIF Data Bus Bit 0. |
| E12 | 79 | 66 | OBE2/POBE | O* | SIO2 Output Buffer Empty/PHIF Output Buffer Empty. |
| D11 | 80 | 67 | IBF2/PIBF | O* | SIO2 Input Buffer Full/PHIF Input Buffer Full. |
| D12 | 81 | 68 | OLD2/PODS | I/O* | SIO2 Output Load/PHIF Output Data Strobe. |
| C11 | 82 | 69 | ILD2/PIDS | I/O* | SIO2 Input Load/PHIF Input Data Strobe. |
| C12 | 83 | 70 | SYNC2/PBSEL | I/O* | SIO2 Multiprocessor Synchronization/PHIF Byte Select. |
| C10 | 84 | 71 | DO2/PSTAT | I/O* | SIO2 Data Output/PHIF Status Register Select. |
| B12 | 85 | 72 | OCK2/PCSN | I/O* | SIO2 Output Clock/PHIF Chip Select Not. |
| B11 | 86 | 73 | DOEN1 | I/O* | SIO1 Data Output Enable. |
| A12 | 87 | 74 | SADD1†† | I/O* | SIO1 Multiprocessor Address. |
| A11 | 90 | 77 | SYNC1 | I/O* | SIO1 Multiprocessor Synchronization. |
| B10 | 91 | 78 | DO1 | O* | SIO1 Data Output. |
| A10 | 92 | 79 | OLD1 | I/O* | SIO1 Output Load. |
| B9 | 93 | 80 | OCK1 | I/O* | SIO1 Output Clock. |
| A9 | 94 | 81 | ICK1 | I/O* | SIO1 Input Clock. |
| B8 | 95 | 82 | ILD1 | I/O* | SIO1 Input Load. |
| A8 | 96 | 83 | DI1 | I | SIO1 Data Input. |
| B7 | 98 | 85 | IBF1 | O* | SIO1 Input Buffer Full. |
| A7 | 99 | 86 | OBE1 | O* | SIO1 Output Buffer Empty. |
| D4, D5, D6, D7, D8, E4, E5, E6, E7, E8, E9, F4, F5, F6, F7, F8, F9, G4, G5, G6, G7, G8, G9, H4, H5, H6, H7, H8, H9, J4, J5, J6, J7, J8, J9 | 6, 14, 26, 38, 49, 64, 76, 89, 97 | 93, 1, 13, 25, 36, 51, 63, 76, 84 | Vss | P | Ground. |
| C4, C5, C6, C7, C8, D3, D9, D10, E3, E10, F3, F10, G3, G10, H3, H10, J3, J10, K4, K5, K6, K7, K8, K9, | 13, 22, 30, 39, 55, 88, 100 | 100, 9, 17, 26, 42, 75, 87 | VDD | P | Power Supply. |
| M10 | 60 | 47 | VDDA | P | Analog Power Supply. |
| M12 | 63 | 50 | VSSA | P | Analog Ground. |
| C9 | — | — | — | — | No Die Connect—unused. |

* 3-states when RSTB = 0, or by JTAG control.

† 3-states when RSTB = 0 and INTO = 1. Output = 1 when RSTB = 0 and INTO = 0, except CKO which is free-running.

‡ Pull-up devices on input.

§ 3-states by JTAG control.

** See Section 7, Mask-Programmable Options.

†† For SIO multiprocessor applications, add 5 kΩ external pull-up resistors to SADD1 and/or SADD2 for proper initialization.

4 Hardware Architecture

The DSP1628 device is a 16-bit, fixed-point programmable digital signal processor (DSP). The DSP1628 consists of a DSP1600 core together with on-chip memory and peripherals. Added architectural features give the DSP1628 high program efficiency for signal coding applications.

4.1 DSP1628 Architectural Overview

Figure 4 shows a block diagram of the DSP1628. The following modules make up the DSP1628.

DSP1600 Core

The DSP1600 core is the heart of the DSP1628 chip. The core contains data and address arithmetic units, and control for on-chip memory and peripherals. The core provides support for external memory wait-states and on-chip dual-port RAM and features vectored interrupts and a trap mechanism.

Dual-Port RAM (DPRAM)

The DSP1628x16 contains 16 banks of zero wait-state memory and the DSP1628x08 contains 8 banks of zero wait-state memory. Each bank consists of 1K 16-bit words and has separate address and data ports to the instruction/coefficient and data memory spaces. A program can reference memory from either space. The DSP1600 core automatically performs the required multiplexing. If references to both ports of a single bank are made simultaneously, the DSP1600 core automatically inserts a wait-state and performs the data port access first, followed by the instruction/coefficient port access.

A program can be downloaded from slow, off-chip memory into DPRAM, and then executed without wait-states. DPRAM is also useful for improving convolution performance in cases where the coefficients are adaptive. Since DPRAM can be downloaded through the JTAG port, full-speed remote in-circuit emulation is possible. DPRAM can also be used for downloading self-test code via the JTAG port.

Read-Only Memory (ROM)

The DSP1628 contains 48K 16-bit words of zero wait-state mask-programmable ROM for program and fixed coefficients.

External Memory Multiplexer (EMUX)

The EMUX is used to connect the DSP1628 to external memory and I/O devices. It supports read/write operations from/to instruction/coefficient memory (X memory space) and data memory (Y memory space). The DSP1600 core automatically controls the EMUX. Instructions can transparently reference external memory from either set of internal buses. A sequencer allows a single instruction to access both the X and the Y external memory spaces.

Clock Synthesis

The DSP powers up with a 1X input clock (CKI/CKI2) as the source for the processor clock. An on-chip clock synthesizer (PLL) can also be used to generate the system clock for the DSP, which will run at a frequency multiple of the input clock. The clock synthesizer is deselected and powered down on reset. For low-power operation, an internally generated slow clock can be used to drive the DSP. If both the clock synthesizer and the internally generated slow clock are selected, the slow clock will drive the DSP; however, the synthesizer will continue to run.

The clock synthesizer and other programmable clock sources are discussed in Section 4.13. The use of these programmable clock sources for power management is discussed in Section 4.14.

4 Hardware Architecture (continued)

Bit Manipulation Unit (BMU)

The BMU extends the DSP1600 core instruction set to provide more efficient bit operations on accumulators. The BMU contains logic for barrel shifting, normalization, and bit field insertion/extraction. The unit also contains a set of 36-bit alternate accumulators. The data in the alternate accumulators can be shuffled with the data in the main accumulators. Flags returned by the BMU mesh seamlessly with the DSP1600 conditional instructions.

Error Correction Coprocessor (ECCP)

The ECCP performs full Viterbi decoding with instructions for MLSE equalization and convolutional decoding. It is designed for 2-tap to 6-tap MLSE equalization with Euclidean branch metrics and rate 1/1 to 1/6 convolutional decoding using constraint lengths from 2 to 7 with Euclidean or Manhattan branch metrics. Two variants of soft-decoded symbols, as well as hard-decoded symbols may be programmed. The ECCP operates in parallel with the DSP1600 core, increasing the throughput rate. Single instruction Viterbi decoding provides significant code compression required for single DSP solutions in modern digital cellular applications. The ECCP is the source of two interrupts and one flag to the DSP1600 core.

Bit Input/Output (BIO)

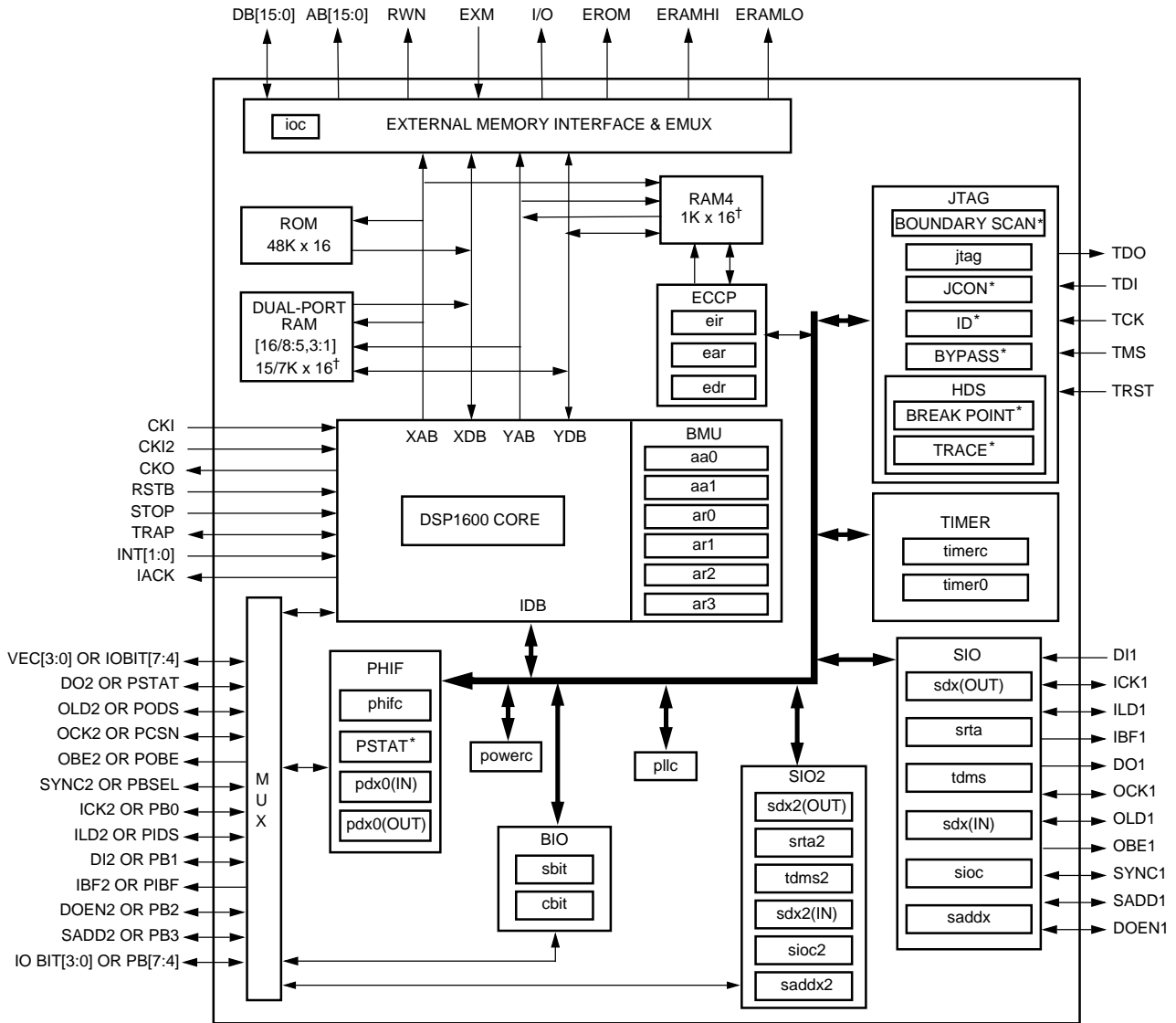
The BIO provides convenient and efficient monitoring and control of eight individually configurable pins. When configured as outputs, the pins can be individually set, cleared, or toggled. When configured as inputs, individual pins or combinations of pins can be tested for patterns. Flags returned by the BIO mesh seamlessly with conditional instructions.

Serial Input/Output Units (SIO and SIO2)

SIO and SIO2 offer asynchronous, full-duplex, double-buffered channels that operate at up to 25 Mbits/s (for 20 ns instruction cycle in a nonmultiprocessor configuration), and easily interface with other Lucent Technologies fixed-point DSPs in a multiple-processor environment. Commercially available codecs and time-division multiplex (TDM) channels can be interfaced to the serial I/O ports with few, if any, additional components. SIO2 is identical to SIO.

An 8-bit serial protocol channel may be transmitted in addition to the address of the called processor in multiprocessor mode. This feature is useful for transmitting high-level framing information or for error detection and correction. SIO2 and BIO are pin-multiplexed with the PHIF.

4 Hardware Architecture (continued)



5-4142 (F).f

* These registers are accessible through the pins only.
† 16K x 16 for the DSP1628x16, 8K x 16 for the DSP1628x08.

Figure 4. DSP1628 Block Diagram

4 Hardware Architecture (continued)

Table 2. DSP1628 Block Diagram Legend

| Symbol | Name |
|---------------|--|
| aa<0—1> | Alternate Accumulators. |
| ar<0—3> | Auxiliary BMU Registers. |
| BIO | Bit Input/Output Unit. |
| BMU | Bit Manipulation Unit. |
| BREAKPOINT | Four Instruction Breakpoint Registers. |
| BYPASS | JTAG Bypass Register. |
| cbit | Control Register for BIO. |
| Dual-Port RAM | Internal RAM (16 Kwords for DSP1628x16, 8 Kwords for DSP1628x08). |
| ECCP | Error Correction Coprocessor. |
| ear | Error Correction Coprocessor Address Register. |
| edr | Error Correction Coprocessor Data Register. |
| eir | Error Correction Coprocessor Instruction Register. |
| EMUX | External Memory Multiplexer. |
| HDS | Hardware Development System. |
| ID | JTAG Device Identification Register. |
| IDB | Internal Data Bus. |
| ioc | I/O Configuration Register. |
| JCON | JTAG Configuration Registers. |
| jtag | 16-bit Serial/Parallel Register. |
| pdx0(in) | Parallel Data Transmit Input Register 0. |
| pdx0(out) | Parallel Data Transmit Output Register 0. |
| PHIF | Parallel Host Interface. |
| phifc | Parallel Host Interface Control Register. |
| pll | Phase-Locked Loop Control Register. |
| powerc | Power Control Register. |
| PSTAT | Parallel Host Interface Status Register. |
| saddx | Multiprocessor Protocol Register. |
| saddx2 | Multiprocessor Protocol Register for SIO2. |
| sbit | Status Register for BIO. |
| sdx(in) | Serial Data Transmit Input Register. |
| sdx2(in) | Serial Data Transmit Input Register for SIO2. |
| sdx(out) | Serial Data Transmit Output Register. |
| sdx2(out) | Serial Data Transmit Output Register for SIO2. |
| SIO | Serial Input/Output Unit. |
| SIO2 | Serial Input/Output Unit #2. |
| sioc | Serial I/O Control Register. |
| sioc2 | Serial I/O Control Register for SIO2. |
| srt | Serial Receive/Transmit Address Register. |
| srt2 | Serial Receive/Transmit Address Register for SIO2. |
| tdms | Serial I/O Time-division Multiplex Signal Control Register. |
| tdms2 | Serial I/O Time-division Multiplex Signal Control Register for SIO2. |
| TIMER | Programmable Timer. |
| timer0 | Timer Running Count Register. |
| timerc | Timer Control Register. |
| TRACE | Program Discontinuity Trace Buffer. |
| XAB | Program Memory Address Bus. |
| XDB | Program Memory Data Bus. |
| YAB | Data Memory Address Bus. |
| YDB | Data Memory Data Bus. |

4 Hardware Architecture (continued)

Parallel Host Interface (PHIF)

The PHIF is a passive, 8-bit parallel port which can interface to an 8-bit bus containing other Lucent Technologies DSPs (e.g., DSP1620, DSP1627, DSP1628, DSP1629, DSP1611, DSP1616, DSP1617, DSP1618), microprocessors, or peripheral I/O devices. The PHIF port supports either *Motorola* or *Intel* protocols, as well as 8-bit or 16-bit transfers, configured in software. The port data rate depends upon the instruction cycle rate. A 25 ns instruction cycle allows the PHIF to support data rates up to 11.85 Mbytes/s, assuming the external host device can transfer 1 byte of data in 25 ns.

The PHIF is accessed in two basic modes, 8-bit or 16-bit mode. In 16-bit mode, the host determines access of the high or low byte. In 8-bit mode, only the low byte is accessed. Software-programmable features allow for a glueless host interface to microprocessors (see Section 4.8, Parallel Host Interface).

Timer

The timer can be used to provide an interrupt at the expiration of a programmed interval. The interrupt may be single or repetitive. More than nine orders of magnitude of interval selection are provided. The timer may be stopped and restarted at any time.

Hardware Development System (HDS) Module

The on-chip HDS performs instruction breakpointing and branch tracing at full speed without additional off-chip hardware. Using the JTAG port, the breakpointing is set up, and the trace history is read back. The port works in conjunction with the HDS code in the on-chip ROM and the hardware and software in a remote computer. The HDS code must be linked to the user's application code and reside in the first 4 Kwords of ROM. The on-chip HDS cannot be used with the secure ROM masking option (see Section 7.2, ROM Security Options).

Four hardware breakpoints can be set on instruction addresses. A counter can be preset with the number of breakpoints to receive before trapping the core. Breakpoints can be set in interrupt service routines. Alternatively, the counter can be preset with the number of cache instructions to execute before trapping the core.

Every time the program branches instead of executing the next sequential instruction, the addresses of the instructions executed before and after the branch are caught in circular memory. The memory contains the last four pairs of program discontinuities for hardware tracing.

In systems with multiple processors, the processors may be configured such that any processor reaching a breakpoint will cause all the other processors to be trapped (see Section 4.3, Interrupts and Trap).

Pin Multiplexing

In order to allow flexible device interfacing while maintaining a low package pin count, the DSP1628 multiplexes 16 package pins between BIO, PHIF, VEC[3:0], and SIO2.

Upon reset, the vectored interrupt indication signals, VEC[3:0], are connected to the package pins while IOBIT[4:7] are disconnected. Setting bit 12, EBIOH, of the **ioc** register connects IOBIT[4:7] to the package pins and disconnects VEC[3:0].

Upon reset, the parallel host interface (PHIF) is connected to the package pins while the second serial port (SIO2) and IOBIT[3:0] are disconnected. Setting bit 10, ESIO2, of the **ioc** register connects the SIO2 and IOBIT[3:0] and disconnects the PHIF.

Power Management

Many applications, such as portable cellular terminals, require programmable sleep modes for power management. There are three different control mechanisms for achieving low-power operation: the **powerc** control register, the STOP pin, and the AWAIT bit in the **alf** register. The AWAIT bit in the **alf** register allows the processor to go into a power-saving standby mode until an interrupt occurs. The **powerc** register configures various power-saving modes by controlling internal clocks and peripheral I/O units. The STOP pin controls the internal processor clock. The various power management options may be chosen based on power consumption and/or wake-up latency requirements.

4.2 DSP1600 Core Architectural Overview

Figure 5 shows a block diagram of the DSP1600 core.

System Cache and Control Section (SYS)

This section of the core contains a 15-word cache memory and controls the instruction sequencing. It handles vectored interrupts and traps, and also provides decoding for registers outside of the DSP1600 core. SYS stretches the processor cycle if wait-states are required (wait-states are programmable for external memory accesses). SYS sequences downloading via JTAG of self-test programs to on-chip, dual-port RAM.

The cache loop iteration count can be specified at run time under program control as well as at assembly time.

4 Hardware Architecture (continued)

Data Arithmetic Unit (DAU)

The data arithmetic unit (DAU) contains a 16 x 16-bit parallel multiplier that generates a full 32-bit product in one instruction cycle. The product can be accumulated with one of two 36-bit accumulators. The accumulator data can be directly loaded from, or stored to, memory in two 16-bit words with optional saturation on overflow. The arithmetic logic unit (ALU) supports a full set of arithmetic and logical operations on either 16- or 32-bit data. A standard set of flags can be tested for conditional ALU operations, branches, and subroutine calls. This procedure allows the processor to perform as a powerful 16- or 32-bit microprocessor for logical and control applications. The available instruction set is compatible with the DSP1618 instruction set. See Section 5.1 for more information on the instruction set.

The user also has access to two additional DAU registers. The **psw** register contains status information from the DAU (see Table 30, Processor Status Word Register). The arithmetic control register, **auc**, is used to configure some of the features of the DAU (see Table 31) including single-cycle squaring. The **auc** register alignment field supports an arithmetic shift left by one and left or right by two. The **auc** register is cleared by reset.

The counters **c0** to **c2** are signed, 8 bits wide, and may be used to count events such as the number of times the program has executed a sequence of code. They are controlled by the conditional instructions and provide a convenient method of program looping.

Y Space Address Arithmetic Unit (YAAU)

The YAAU supports high-speed, register-indirect, compound, and direct addressing of data (Y) memory. Four general-purpose, 16-bit registers, **r0** to **r3**, are available in the YAAU. These registers can be used to supply the read or write addresses for Y space data. The YAAU also decodes the 16-bit data memory address and outputs individual memory enables for the data access. The YAAU can address the six 1 Kword banks of on-chip DPRAM or three external data memory segments. Up to 48 Kwords of off-chip RAM are addressable, with 16K addresses reserved for internal RAM.

Two 16-bit registers, **rb** and **re**, allow zero-overhead modulo addressing of data for efficient filter implementations. Two 16-bit signed registers, **j** and **k**, are used to hold user-defined postmodification increments. Fixed increments of +1, -1, and +2 are also available. Four compound-addressing modes are provided to make read/write operations more efficient.

The YAAU allows direct (or indexed) addressing of data memory. In direct addressing, the 16-bit base register (**ybase**) supplies the 11 most significant bits of the address. The direct data instruction supplies the remaining 5 bits to form an address to Y memory space and also specifies one of 16 registers for the source or destination.

X Space Address Arithmetic Unit (XAAU)

The XAAU supports high-speed, register-indirect, instruction/coefficient memory addressing with postmodification of the register. The 16-bit **pt** register is used for addressing coefficients. The signed register **i** holds a user-defined postincrement. A fixed postincrement of +1 is also available. Register **PC** is the program counter. Registers **pr** and **pi** hold the return address for subroutine calls and interrupts, respectively.

The XAAU decodes the 16-bit instruction/coefficient address and produces enable signals for the appropriate X memory segment. The addressable X segments are 48 Kwords of internal ROM, up to 16 Kwords of DPRAM for the DSP1628x16 or up to 8 Kwords of DPRAM for the DSP1628x08, and external ROM.

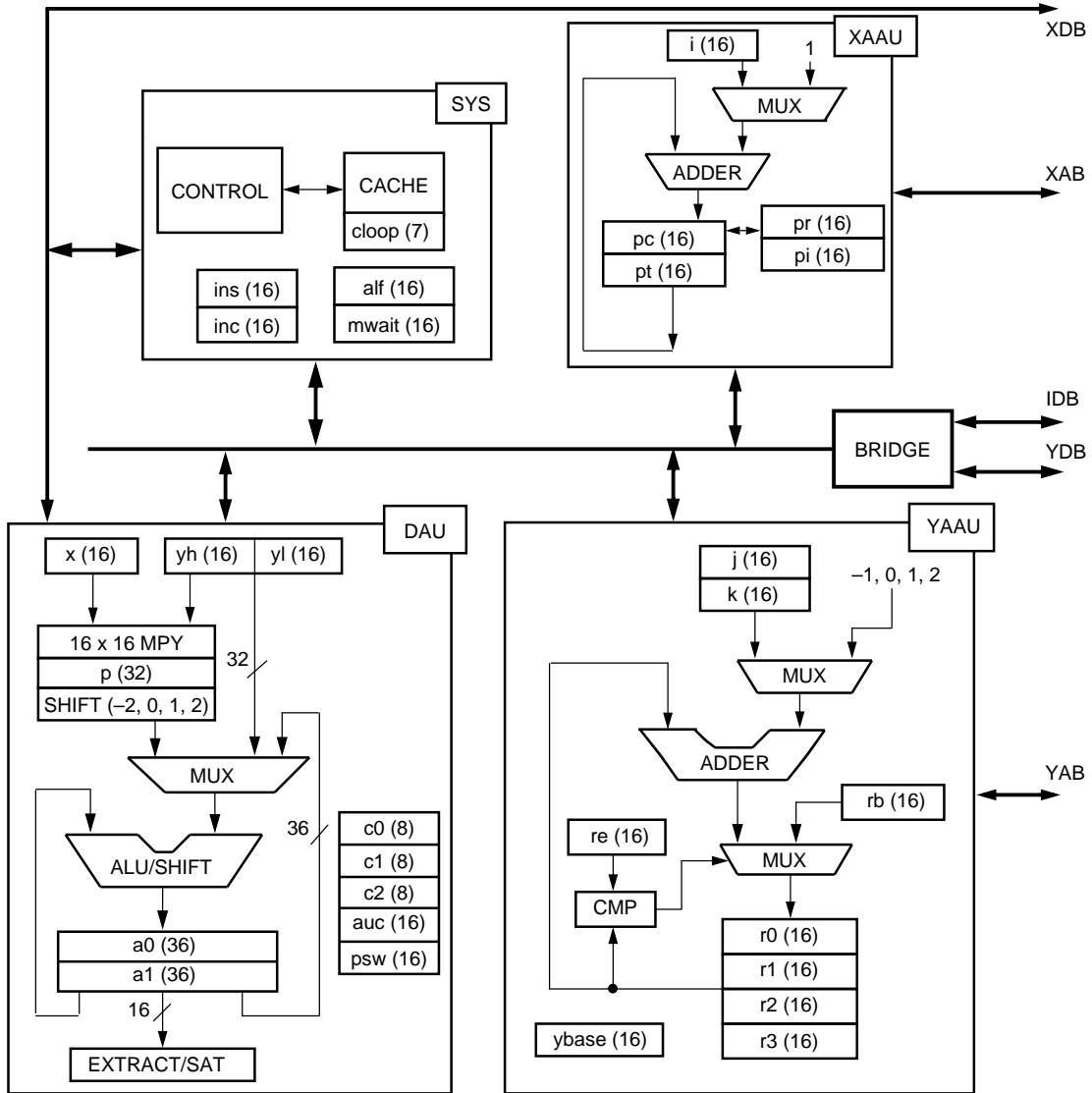
The locations of these memory segments depend upon the memory map selected (see Table 5). A security mode can be selected by mask option. This prevents unauthorized access to the contents of on-chip ROM (see Section 7, Mask-Programmable Options).

4.3 Interrupts and Trap

The DSP1628 supports prioritized, vectored interrupts and a trap. The device has eight internal hardware sources of program interrupt and two external interrupt pins. Additionally, there is a trap pin and a trap signal from the hardware development system (HDS). A software interrupt is available through the **icall** instruction. The **icall** instruction is reserved for use by the HDS. Each of these sources of interrupt and trap has a unique vector address and priority assigned to it. DSP16A interrupt compatibility is not maintained.

The software interrupt and the traps are always enabled and do not have a corresponding bit in the **ins** register. Other vectored interrupts are enabled in the **inc** register (see Table 33, Interrupt Control (**inc**) Register) and monitored in the **ins** register (see Table 34, Interrupt Status (**ins**) Register). When the DSP1628 goes into an interrupt or trap service routine, the IACK pin is asserted. In addition, pins VEC[3:0] encode which interrupt/trap is being serviced. Table 4 details the encoding used for VEC[3:0].

4 Hardware Architecture (continued)



5-1741 (F),b

Figure 5. DSP1600 Core Block Diagram

4 Hardware Architecture (continued)

Table 3. DSP1600 Core Block Diagram Legend

| Symbol | Name |
|-------------|--|
| 16 x 16 MPY | 16-bit x 16-bit Multiplier. |
| a0—a1 | Accumulators 0 and 1 (16-bit halves specified as a0 , a0l , a1 , and a1l)*. |
| alf | AWAIT, LOWPR, Flags. |
| ALU/SHIFT | Arithmetic Logic Unit/Shifter. |
| auc | Arithmetic Unit Control. |
| c0—c2 | Counters 0—2. |
| cloop | Cache Loop Count. |
| CMP | Comparator. |
| DAU | Digital Arithmetic Unit. |
| i | Increment Register for the X Address Space. |
| IDB | Internal Data Bus. |
| inc | Interrupt Control. |
| ins | Interrupt Status. |
| j | Increment Register for the Y Address Space. |
| k | Increment Register for the Y Address Space. |
| MUX | Multiplexer. |
| mwait | External Memory Wait-states Register. |
| p | Product Register (16-bit halves specified as p , pl). |
| PC | Program Counter. |
| pi | Program Interrupt Return Register. |
| pr | Program Return Register. |
| psw | Processor Status Word. |
| pt | X Address Space Pointer. |
| r0—r3 | Y Address Space Pointers. |
| rb | Modulo Addressing Register (begin address). |
| re | Modulo Addressing Register (end address). |
| SYS | System Cache and Control Section. |
| x | Multiplier Input Register. |
| XAAU | X Space Address Arithmetic Unit. |
| XAB | X Space Address Bus. |
| XDB | X Space Data Bus. |
| YAAU | Y Space Address Arithmetic Unit. |
| YAB | Y Space Address Bus. |
| YDB | Y Space Data Bus. |
| ybase | Direct Addressing Base Register. |
| y | DAU Register (16-bit halves specified as y , yl). |

* F3 ALU instructions with immediates require specifying the high half of the accumulators as **a0h** and **a1h**.

4 Hardware Architecture (continued)

Interruptibility

Vectored interrupts are serviced only after the execution of an interruptible instruction. If more than one vectored interrupt is asserted at the same time, the interrupts are serviced sequentially according to their assigned priorities. See Table 4 for the priorities assigned to the vectored interrupts. Interrupt service routines, branch and conditional branch instructions, cache loops, and instructions that only decrement one of the RAM pointers, **r0** to **r3** (e.g., ***r3--**), are not interruptible.

A trap is similar to an interrupt, but it gains control of the processor by branching to the trap service routine even when the current instruction is noninterruptible. It may not be possible to return to normal instruction execution from the trap service routine since the machine state cannot always be saved. In particular, program execution cannot be continued from a trapped cache loop or interrupt service routine. While in a trap service routine, another trap is ignored.

When set to 1, the status bits in the **ins** register indicate that an interrupt has occurred. The processor must reach an interruptible state (completion of an interruptible instruction) before an enabled vectored interrupt will be acted on. An interrupt will not be serviced if it is not enabled. Polled interrupt service can be implemented by disabling the interrupt in the **inc** register and then polling the **ins** register for the expected event.

Vectored Interrupts

Tables 33 and 34 show the **inc** and **ins** registers. A logic 1 written to any bit of **inc** enables (or unmask) the associated interrupt. If the bit is cleared to a logic 0, the interrupt is masked. Note that neither the software interrupt nor traps can be masked.

The occurrence of an interrupt that is not masked will cause the program execution to transfer to the memory location pointed to by that interrupt's vector address, assuming no other interrupt is being serviced (see Table 4, Interrupt Vector Table). The occurrence of an interrupt that is masked causes no automatic processor action, but will set the corresponding status bit in the **ins** register. If a masked interrupt occurs, it is latched in the **ins** register, but the interrupt is not taken. When unlatched, this latched interrupt will initiate automatic processor interrupt action. See the *DSP1611/17/18/27 Digital Signal Processor Information Manual* for a more detailed description of the interrupts.

Signaling Interrupt Service Status

Five pins of DSP1628 are devoted to signaling interrupt service status. The IACK pin goes high while any interrupt or user trap is being serviced, and goes low when the ireturn instruction from the service routine is issued. Four pins, VEC[3:0], carry a code indicating which of the interrupts or trap is being serviced. Table 4 contains the encodings used by each interrupt.

Traps due to HDS breakpoints have no effect on either the IACK or VEC[3:0] pins. Instead, they show the interrupt state or interrupt source of the DSP when the trap occurred.

Clearing Interrupts

The PHIF interrupts (PIBF and POBE) are cleared by reading or writing the parallel host interface data transmit registers **pdx0[in]** and **pdx0[out]**, respectively. The SIO and SIO2 interrupts (IBF, IBF2, OBE, and OBE2) are cleared one instruction cycle AFTER reading or writing the serial data registers, (**sdx[in]**, **sdx2[in]**, **sdx[out]**, or **sdx2[out]**). To account for this added latency, the user must ensure that a single instruction (NOP or any other valid DSP16XX instruction) follows the **sdx** register read or write instruction prior to exiting an interrupt service routine (via an ireturn or goto pi instruction) or before checking the **ins** register for the SIO flag status. Adding this instruction ensures that interrupts are not reported incorrectly following an ireturn or that stale flags are not read from the **ins** register. The JTAG interrupt (JINT) is cleared by reading the **jtag** register.

Five of the vectored interrupts are cleared by writing to the **ins** register. Writing a 1 to the INT0, INT1, EREADY, EOVF, or TIME bits in the **ins** will cause the corresponding interrupt status bit to be cleared to a logic 0. The status bit for these vectored interrupts is also cleared when the ireturn instruction is executed, leaving set any other vectored interrupts that are pending.

Traps

The TRAP pin of the DSP1628 is a bidirectional signal. At reset, it is configured as an input to the processor. Asserting the TRAP pin will force a user trap. The trap mechanism is used for two purposes. It can be used by an application to rapidly gain control of the processor for asynchronous time-critical event handling (typically for catastrophic error recovery). It is also used by the HDS for breakpointing and gaining control of the processor. Separate vectors are provided for the user trap (0x46) and the HDS trap (0x3). Traps are not maskable.

4 Hardware Architecture (continued)

Table 4. Interrupt Vector Table

| Source | Vector | Priority | VEC[3:0] | Issued by |
|--------------------|--------|--------------|----------|--------------------------|
| No Interrupt | — | — | 0x0 | — |
| Software Interrupt | 0x2 | 1 | 0x1 | icall |
| INT0 | 0x1 | 2 | 0x2 | pin |
| JINT | 0x42 | 3 | 0x8 | jtag in |
| INT1 | 0x4 | 4 | 0x9 | pin |
| TIME | 0x10 | 7 | 0xc | timer |
| IBF2 | 0x14 | 8 | 0xd | SIO2 in |
| OBE2 | 0x18 | 9 | 0xe | SIO2 out |
| Reserved | 0x1c | 10 | 0x0 | — |
| EREDY | 0x20 | 11 | 0x1 | ECCP ready |
| EOVF | 0x24 | 12 | 0x2 | ECCP overflow |
| IBF | 0x2c | 14 | 0x3 | SIO in |
| OBE | 0x30 | 15 | 0x4 | SIO out |
| PIBF | 0x34 | 16 | 0x5 | PHIF in |
| POBE | 0x38 | 17 | 0x6 | PHIF out |
| TRAP from HDS | 0x3 | 18 | * | breakpoint, jtag, or pin |
| TRAP from User | 0x46 | 19 = highest | 0x7 | pin |

* Traps due to HDS breakpoints have no effect on VEC[3:0] pins.

A trap has four cycles of latency. At most, two instructions will execute from the time the trap is received at the pin to when it gains control. An instruction that is executing when a trap occurs is allowed to complete before the trap service routine is entered. (Note that the instruction could be lengthened by wait-states.) During normal program execution, the **pi** register contains either the address of the next instruction (two-cycle instruction executing) or the address following the next instruction (one-cycle instruction executing). In an interrupt service routine, **pi** contains the interrupt return address. When a trap occurs during an interrupt service routine, the value of the **pi** register may be overwritten. Specifically, it is not possible to return to an interrupt service routine from a user trap (0x46) service routine. Continuing program execution when a trap occurs during a cache loop is also not possible.

The HDS trap causes circuitry to force the program memory map to MAP1 (with on-chip ROM starting at address 0x0) when the trap is taken. The previous memory map is restored when the trap service routine exits by issuing an **ireturn**. The map is forced to MAP1 because the HDS code, if present, resides in the on-chip ROM.

Using the Lucent Technologies development tools, the TRAP pin may be configured to be an output, or an input vectoring to address 0x3. In a multiprocessor environment, the TRAP pins of all the DSPs present can be tied together. During HDS operations, one DSP is selected by the host software to be the master. The master processor's TRAP pin is configured to be an output.

The TRAP pins of the slave processors are configured as inputs. When the master processor reaches a breakpoint, the master's TRAP pin is asserted. The slave processors will respond to their TRAP input by beginning to execute the HDS code.

AWAIT Interrupt (Standby or Sleep Mode)

Setting the AWAIT bit (bit 15) of the **alf** register (**alf** = 0x8000) causes the processor to go into a power-saving standby or sleep mode. Only the minimum circuitry on the chip required to process an incoming interrupt remains active. After the AWAIT bit is set, one additional instruction will be executed before the standby power-saving mode is entered. A PHIF or SIO word transfer will complete if already in progress. The AWAIT bit is reset when the first interrupt occurs. The chip then wakes up and continues executing.

Two **nop** instructions should be programmed after the AWAIT bit is set. The first **nop** (one cycle) will be executed before sleeping; the second will be executed after the interrupt signal awakens the DSP and before the interrupt service routine is executed.

4 Hardware Architecture (continued)

The WAIT bit should be set from within the cache if the code which is executing resides in external ROM where more than one wait-state has been programmed. This ensures that an interrupt will not disturb the device from completely entering the sleep state.

For additional power savings, set **ioc = 0x0180** and **timerc = 0x0040** in addition to setting **alf = 0x8000**. This will hold the CKO pin low and shut down the timer and prescaler (see Table 42 and Table 35).

For a description of the control mechanisms for putting the DSP into low-power modes, see Section 4.13, Power Management.

4.4 Memory Maps and Wait-States

The DSP1600 core implements a modified Harvard architecture that has separate on-chip 16-bit address and data buses for the instruction/coefficient (X) and data (Y) memory spaces. Table 5 shows the instruction/coefficient memory space maps for both the DSP1628x16 and DSP1628x08.

The DSP1628 provides a multiplexed external bus which accesses external RAM (ERAM) and ROM (EROM). Programmable wait-states are provided for external memory accesses. The instruction/coefficient memory map is configurable to provide application flexibility. Table 6 shows the data memory space, which has one map.

Instruction/Coefficient Memory Map Selection

In determining which memory map to use, the processor evaluates the state of two parameters. The first is the LOWPR bit (bit 14) of the **alf** register. The LOWPR bit of the **alf** register is initialized to 0 automatically at reset. LOWPR controls the starting address in memory assigned to 1K banks of dual-port RAM. If LOWPR is low, internal dual-port RAM begins at address 0xC000. If LOWPR is high, internal dual-port RAM begins at address 0x0. LOWPR also moves IROM from 0x0 in MAP1 to 0x4000 in MAP3, and EROM from 0x0 in MAP2 to 0x4000 in MAP4.

The second parameter is the value at reset of the EXM pin (pin 27 or pin 14, depending upon the package type). EXM determines whether the internal 48 Kwords ROM (IROM) will be addressable in the memory map.

The Lucent Technologies development system tools, together with the on-chip HDS circuitry and the JTAG port, can independently set the memory map. Specifically, during an HDS trap, the memory map is forced to MAP1. The user's map selection is restored when the trap service routine has completed execution.

MAP1

MAP1 has the IROM starting at 0x0 and 1 Kword banks of DPRAM starting at 0xC000. MAP1 is used if DSP1628 has EXM low at reset and the LOWPR parameter is programmed to zero. It is also used during an HDS trap.

MAP2

MAP2 differs from MAP1 in that the lowest 48 Kwords reference external ROM (EROM). MAP2 is used if EXM is high at reset, the LOWPR parameter is programmed to zero, and an HDS trap is not in progress.

MAP3

MAP3 has the 1 Kword banks of DPRAM starting at address 0x0. In MAP3, the 48 Kwords of IROM start at 0x4000. MAP3 is used if EXM is low at reset, the LOWPR bit is programmed to 1, and an HDS trap is not in progress. Note that this map is not available if the secure mask-programmable option has been ordered.

MAP4

MAP4 differs from MAP3 in that addresses above 0x4000 reference external ROM (EROM). This map is used if the LOWPR bit is programmed to 1, an HDS trap is not in progress, and, either EXM is high during reset, or the secure mask-programmable option has been ordered.

Whenever the chip is reset using the RSTB pin, the default memory map will be MAP1 or MAP2, depending upon the state of the EXM pin at reset. A reset through the HDS will not reinitialize the **alf** register, so the previous memory map is retained.

Boot from External ROM

After RSTB goes from low to high, the DSP1628 comes out of reset and fetches an instruction from address zero of the instruction/coefficient space. The physical location of address zero is determined by the memory map in effect. If EXM is high at the rising edge of RSTB, MAP2 is selected. MAP2 has EROM at location zero; thus, program execution begins from external memory. If EXM is high and INT1 is low when RSTB rises, the **mwait** register defaults to 15 wait-states for all external memory segments. If INT1 is high, the **mwait** register defaults to 0 wait-states.

4 Hardware Architecture (continued)

Table 5. Instruction/Coefficient Memory Maps
DSP1628x16

| X Address | AB[0:15] | MAP 1* EXM = 0 LOWPR = 0 [†] | MAP 2 EXM = 1 LOWPR = 0 | MAP 3 [‡] EXM = 0 LOWPR = 1 | MAP 4 EXM = 1 LOWPR = 1 | | |
|-----------|----------|---|-------------------------------|--|-------------------------------|---------------|---------------|
| 0 | 0x0000 | IROM (48K) | EROM (48K) | DPRAM (16K) | DPRAM (16K) | | |
| 4K | 0x1000 | | | | | | |
| 8K | 0x2000 | | | | | | |
| 12K | 0x3000 | | | | | | |
| 16K | 0x4000 | | | | | | |
| 20K | 0x5000 | | | | | | |
| 24K | 0x6000 | | | | | | |
| 28K | 0x7000 | | | | | | |
| 32K | 0x8000 | | | | | | |
| 36K | 0x9000 | | | | | | |
| 40K | 0xA000 | | | | | | |
| 44K | 0xB000 | | | | | | |
| 48K | 0xC000 | | | DPRAM (16K) | DPRAM (16K) | IROM (48K) | EROM (48K) |
| 52K | 0xD000 | | | | | | |
| 54K | 0xD800 | | | | | | |
| 56K | 0xE000 | | | | | | |
| 60K—64K | 0xFFFF | | | | | | |

DSP1628x08

| X Address | AB[0:15] | MAP 1* EXM = 0 LOWPR = 0 [†] | MAP 2 EXM = 1 LOWPR = 0 | MAP 3 [‡] EXM = 0 LOWPR = 1 | MAP 4 EXM = 1 LOWPR = 1 |
|-----------|----------|---|-------------------------------|--|-------------------------------|
| 0 | 0x0000 | IROM (48K) | EROM (48K) | DPRAM (8K) | DPRAM (8K) |
| 4K | 0x1000 | | | | |
| 6K | 0x1800 | | | | |
| 8K | 0x2000 | | | | |
| 12K | 0x3000 | | | | |
| 16K | 0x4000 | | | | |
| 20K | 0x5000 | | | | |
| 24K | 0x6000 | | | | |
| 28K | 0x7000 | | | | |
| 32K | 0x8000 | | | | |
| 36K | 0x9000 | | | | |
| 40K | 0xA000 | | | | |
| 44K | 0xB000 | | | | |
| 48K | 0xC000 | | | DPRAM (8K) | DPRAM (8K) |
| 52K | 0xD000 | | | | |
| 54K | 0xD800 | | | | |
| 56K | 0xE000 | | | | |
| 58K | 0xE800 | | | | |
| 60K—64K | 0xFFFF | | | | |

* MAP1 is set automatically during an HDS trap. The user-selected map is restored at the end of the HDS trap service routine.

[†] LOWPR is an **alf** register bit. The Lucent Technologies development system tools can independently set the memory map.

[‡] MAP3 is not available if the secure mask-programmable option is selected.

4 Hardware Architecture (continued)

Table 6. Data Memory Maps

1628x16 Data Memory Map (Not to Scale)

| Decimal Address | Address in r0, r1, r2, r3 | Segment |
|-----------------|---------------------------|-------------|
| 0 | 0x0000 | DPRAM[1:16] |
| 16K | 0x4000 | IO |
| 16,640 | 0x4100 | ERAMLO |
| 32K | 0x8000 | ERAMHI |
| 64K – 1 | 0xFFFF | |

1628x08 Data Memory Map (Not to Scale)

| Decimal Address | Address in r0, r1, r2, r3 | Segment |
|-----------------|---------------------------|------------|
| 0 | 0x0000 | DPRAM[1:8] |
| 8K | 0x2000 | Reserved |
| 16K | 0x4000 | IO |
| 16,640 | 0x4100 | ERAMLO |
| 32K | 0x8000 | ERAMHI |
| 64K – 1 | 0xFFFF | |

On the data memory side (see Table), the 1K banks of dual-port RAM are located starting at address 0. Addresses from 0x4000 to 0x40FF reference a 256-word memory-mapped I/O segment (IO). Addresses from 0x4100 to 0x7FFF reference the low external data RAM segment (ERAMLO). Addresses above 0x8000 reference high external data RAM (ERAMHI).

Wait-States

The number of wait-states (from 0 to 15) used when accessing each of the four external memory segments (ERAMLO, IO, ERAMHI, and EROM) is programmable in the **mwait** register (see Table 40). When the program references memory in one of the four external segments, the internal multiplexer is automatically switched to the appropriate set of internal buses, and the associated external enable of ERAMLO, IO, ERAMHI, or EROM is issued. The external memory cycle is automatically stretched by the number of wait-states configured in the appropriate field of the **mwait** register.

4 Hardware Architecture (continued)

4.5 External Memory Interface (EMI)

The external memory interface supports read/write operations from instruction/coefficient memory, data memory, and memory-mapped I/O devices. The DSP1628 provides a 16-bit external address bus, AB[15:0], and a 16-bit external data bus, DB[15:0]. These buses are multiplexed between the internal buses for the instruction/coefficient memory and the data memory. Four external memory segment enables, ERAMLO, IO, ERAMHI, and EROM, select the external memory segment to be addressed.

If a data memory location with an address between 0x4100 and 0x7FFF is addressed, ERAMLO is asserted low.

If one of the 256 external data memory locations, with an address greater than or equal to 0x4000, and less than or equal to 0x40FF, is addressed, IO is asserted low. IO is intended for memory-mapped I/O.

If a data memory location with an address greater than or equal to 0x8000 is addressed, ERAMHI is asserted low. When the external instruction/coefficient memory is addressed, EROM is asserted low.

The flexibility provided by the programmable options of the external memory interface (see Table 40, **mwait** Register and Table 42, **ioc** Register) allows the DSP1628 to interface gluelessly with a variety of commercial memory chips.

Each of the four external memory segments, ERAMLO, IO, ERAMHI, and EROM, has a number of wait-states that is programmable (from 0 to 15) by writing to the **mwait** register. When the program references memory in one of the four external segments, the internal multiplexer is automatically switched to the appropriate set of internal buses, and the associated external enable of ERAMLO, IO, ERAMHI, or EROM is issued. The external memory cycle is automatically stretched by the number of wait-states in the appropriate field of the **mwait** register.

When writing to external memory, the RWN pin goes low for the external cycle. The external data bus, DB[15:0], is driven by the DSP1628 starting halfway through the cycle. The data driven on the external data bus is automatically held after the cycle for one additional clock period unless an external read cycle immediately follows.

The DSP1628 has one external address bus and one external data bus for both memory spaces. Since some instructions provide the capability of simultaneous access to both X space and Y space, some provision must be made to avoid collisions for external accesses. The DSP1628 has a sequencer that does the external X access first, and then the external Y access, transparently to the programmer. Wait-states are maintained as programmed in the **mwait** register. For example, let two in-

structions be executed: the first reads a coefficient from EROM and writes data to ERAM; the second reads a coefficient from EROM and reads data from ERAM. The sequencer carries out the following steps at the external memory interface: read EROM, write ERAM, read EROM, and read ERAM. Each step is done in sequential one-instruction cycle steps, assuming zero wait-states are programmed. Note that the number of instruction cycles taken by the two instructions is four. Also, in this case, the write hold time is zero.

The DSP1628 allows writing into external instruction/coefficient memory. By setting bit 11, WEROM, of the **ioc** register (see Table 42), writing to (or reading from) data memory or memory-mapped I/O asserts the EROM strobe instead of ERAMLO, IO, or ERAMHI. Therefore, with WEROM set, EROM appears in both Y space (replacing ERAM) and X space, in its normal position.

Bit 14 of the **ioc** register (see Table 42), EXTROM, may be used with WEROM to download to a full 64K of external memory. When WEROM and EXTROM are both asserted, address bit 15 (AB15) is held low, aliasing the upper 32K of external memory into the lower 32K.

When an access to internal memory is made, the AB[15:0] bus holds the last valid external memory address. Asserting the RSTB pin low 3-states the AB[15:0] bus. After reset, the AB[15:0] value is undefined.

The leading edge of the memory segment enables can be delayed by approximately one-half a CKO period by programming the **ioc** register (see Table 42). This is used to avoid a situation in which two devices drive the data bus simultaneously.

Bits 7, 8, and 13 of the **ioc** register select the mode of operation for the CKO pin (see Table 42). Available options are a free-running unstretched clock, a wait-stated sequenced clock (runs through two complete cycles during a sequenced external memory access), and a wait-stated clock based on the internal instruction cycle. These clocks drop to the low-speed internal ring oscillator when SLOWCKI is enabled (see 4.13, Power Management). The high-to-low transitions of the wait-stated clock are synchronized to the high-to-low transition of the free-running clock. Also, the CKO pin provides either a continuously high level, a continuously low level, or changes at the rate of the internal processor clock. This last option, only available with the small-signal input clock options, enables the DSP1628 CKI input buffer to deliver a full-rate clock to other devices while the DSP1628 itself is in one of the low-power modes.

4 Hardware Architecture (continued)

4.6 Bit Manipulation Unit (BMU)

The BMU interfaces directly to the main accumulators in the DAU providing the following features:

- Barrel shifting—logical and arithmetic, left and right shift
- Normalization and extraction of exponent
- Bit-field extraction and insertion

These features increase the efficiency of the DSP in applications such as control or data encoding and decoding. For example, data packing and unpacking, in which short data words are packed into one 16-bit word for more efficient memory storage, is very easy.

In addition, the BMU provides two auxiliary accumulators, **aa0** and **aa1**. In one instruction cycle, 36-bit data can be shuffled, or swapped, between one of the main accumulators and one of the alternate accumulators. The **ar<0—3>** registers are 16-bit registers that control the operations of the BMU. They store a value that determines the amount of shift or the width and offset fields for bit extraction or insertion. Certain operations in the BMU set flags in the DAU **psw** register and the **alf** register (see Table 30, Processor Status Word (**psw**) Register, and Table 39, **alf** Register). The **ar<0—3>** registers can also be used as general-purpose registers.

The BMU instructions are detailed in Section 5.1. For a thorough description of the BMU, see the *DSP1611/17/18/27 Digital Signal Processor Information Manual*.

4.7 Serial I/O Units (SIOs)

The serial I/O ports on the DSP1628 device provide a serial interface to many codecs and signal processors with little, if any, external hardware required. Each high-speed, double-buffered port (**sdx** and **sdx2**) supports back-to-back transmissions of data. SIO and SIO2 are identical. The output buffer empty (OBE and OBE2) and input buffer full (IBF and IBF2) flags facilitate the reading and/or writing of each serial I/O port by program- or interrupt-driven I/O. There are four selectable active clock speeds.

A bit-reversal mode provides compatibility with either the most significant bit (MSB) first or least significant bit (LSB) first serial I/O formats (see Table 26, Serial I/O Control Registers (**sioc** and **sioc2**)). A multiprocessor I/O configuration is supported. This feature allows up to eight DSP161X devices to be connected together on an SIO port without requiring external glue logic.

The serial data may be internally looped back by setting the SIO loopback control bit, SIOLBC, of the **ioc** register. SIOLBC affects both the SIO and SIO2. The data output signals are wrapped around internally from the output to the input (DO1 to DI1 and DO2 to DI2). To ex-

ercise loopback, the SIO clocks (ICK1, ICK2, OCK1, and OCK2) should either all be in the active mode, 16-bit condition, or each pair should be driven from one external source in passive mode. Similarly, pins ILD1 (ILD2) and OLD1 (OLD2) must both be in active mode or tied together and driven from one external frame clock in passive mode. During loopback, DO1, DO2, DI1, DI2, ICK1, ICK2, OCK1, OCK2, ILD1, ILD2, OLD1, OLD2, SADD1, SADD2, SYNC1, SYNC2, DOEN1, and DOEN2 are 3-stated.

Setting DODLY = 1 (**sioc** and **sioc2**) delays DO by one phase of OCK so that DO changes on the falling edge of OCK instead of the rising edge (DODLY = 0). This reduces the time available for DO to drive DI and to be valid for the rising edge of ICK, but increases the hold time on DO by half a cycle on OCK.

Programmable Modes

Programmable modes of operation for the SIO and SIO2 are controlled by the serial I/O control registers (**sioc** and **sioc2**). These registers, shown in Table 26, are used to set the ports into various configurations. Both input and output operations can be independently configured as either active or passive. When active, the DSP1628 generates load and clock signals. When passive, load and clock signal pins are inputs.

Since input and output can be independently configured, each SIO has four different modes of operation. Each of the **sioc** registers is also used to select the frequency of active clocks for that SIO. Finally, these registers are used to configure the serial I/O data formats. The data can be 8 or 16 bits long, and can also be input/output MSB first or LSB first. Input and output data formats can be independently configured.

Multiprocessor Mode

The multiprocessor mode allows up to eight devices that support multiprocessor mode (codecs or DSP16XX devices) to be connected together to provide data transmission among any of the multiprocessor devices in the system. Either of the DSP1628's SIO ports (SIO or SIO2) may be independently used for the multiprocessor mode. The multiprocessor interface is a four-wire interface, consisting of a data channel, an address/protocol channel, a transmit/receive clock, and a sync signal (see Figure 6). The DI1 and DO1 pins of all the DSPs are connected to transmit and receive the data channel. The SADD1 pins of all the DSPs are connected to transmit and receive the address/protocol channel. ICK1 and OCK1 should be tied together and driven from one source. The SYNC1 pins of all the DSPs are connected.

In the configuration shown in Figure 6, the master DSP (DSP0) generates active SYNC1 and OCK1 signals while the slave DSPs use the SYNC1 and OCK1 signals in passive mode to synchronize operations. In addition, all DSPs must have their ILD1 and OLD1 signals in active mode.

4 Hardware Architecture (continued)

While ILD1 and OLD1 are not required externally for multiprocessor operation, they are used internally in the DSP's SIO. Setting the LD field of the master's **sioc** register to a logic level 1 will ensure that the active generation of SYNC1, ILD1, and OLD1 is derived from OCK1 (see Table 26). With this configuration, all DSPs should use ICK1 (tied to OCK1) in passive mode to avoid conflicts on the clock (CK) line (see the *DSP1611/17/18/27 Digital Signal Processor Information Manual* for more information).

Four registers (per SIO) configure the multiprocessor mode: the time-division multiplexed slot register (**tdms** or **tdms2**), the serial receive and transmit address register (**srta** or **srta2**), the serial data transmit register (**sdx** or **sdx2**), and the multiprocessor serial address/protocol register (**saddx** or **saddx2**).

Multiprocessor mode requires no external logic and uses a TDM interface with eight 16-bit time slots per frame. The transmission in any time slot consists of 16 bits of serial data in the data channel and 16 bits of address and protocol information in the address/protocol channel. The address information consists of the transmit address field of the **srta** register of the transmitting device. The address information is transmitted concurrently with the transmission of the first 8 bits of data. The protocol information consists of the transmit protocol field written to the **saddx** register and is transmitted concurrently with the last 8 bits of data (see Table 29, Multiprocessor Protocol Register). Data is received or recognized by other DSP(s) whose receive address matches the address in the address/protocol channel. Each SIO port has a user-programmable receive address and transmit address associated with it. The transmit and receive addresses are programmed in the **srta** register.

In multiprocessor mode, each device can send data in a unique time slot designated by the **tdms** register transmit slot field (bits 7—0). The **tdms** register has a fully decoded transmit slot field in order to allow one DSP1628 device to transmit in more than one time slot. This procedure is useful for multiprocessor systems with less than eight DSP1628 devices when a higher bandwidth is necessary between certain devices in that system. The DSP operating during time slot 0 also drives SYNC1.

In order to prevent multiple bus drivers, only one DSP can be programmed to transmit in a particular time slot. In addition, it is important to note that the address/protocol channel is 3-stated in any time slot that is not being driven.

Therefore, to prevent spurious inputs, the address/protocol channel should be pulled up to V_{DD} with a 5 k Ω resistor, or it should be guaranteed that the bus is driven in every time slot. (If the SYNC1 signal is externally generated, then this pull-up is required for correct initialization.)

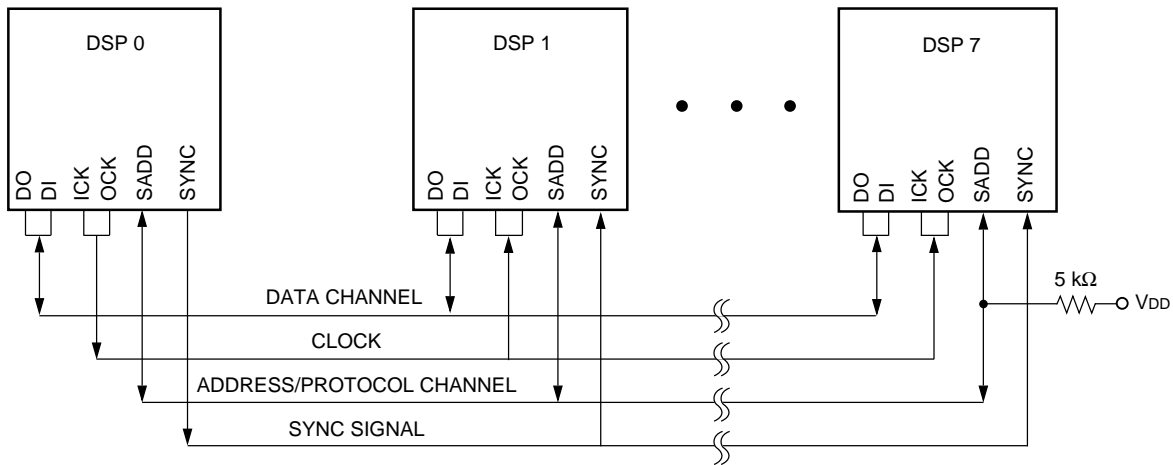
Each SIO also has a fully decoded transmitting address specified by the **srta** register transmit address field (bits 7—0). This is used to transmit information regarding the destination(s) of the data. The fully decoded receive address specified by the **srta** register receive address field (bits 15—8) determines which data will be received. The SIO protocol channel data is controlled via the **saddx** register. When the **saddx** register is written, the lower 8 bits contain the 8-bit protocol field. On a read, the high-order 8 bits read from **saddx** are the most recently received protocol field sent from the transmitting DSP's **saddx** output register. The low-order 8 bits are read as 0s.

An example use of the protocol channel is to use the top 3 bits of the **saddx** value as an encoded source address for the DSPs on the multiprocessor bus. This leaves the remaining 5 bits available to convey additional control information, such as whether the associated field is an opcode or data, or whether it is the last word in a transfer, etc. These bits can also be used to transfer parity information about the data. Alternatively, the entire field can be used for data transmission, boosting the bandwidth of the port by 50%.

Using SIO2

The SIO2 functions the same as the SIO. Please refer to Pin Multiplexing in Section 4.1 for a description of pin multiplexing of BIO, PHIF, VEC[3:0], and SIO2.

4 Hardware Architecture (continued)



5-4181 (F).a

Figure 6. Multiprocessor Communication and Connections

4.8 Parallel Host Interface (PHIF)

The DSP1628 has an 8-bit parallel host interface for rapid transfer of data with external devices. This parallel port is passive (data strobes provided by an external device) and supports either *Motorola* or *Intel* microcontroller protocols. The PHIF also provides for 8-bit or 16-bit data transfers. As a flexible host interface, it requires little or no glue logic to interface to other devices (e.g., microcontrollers, microprocessors, or another DSP).

The data path of the PHIF consists of a 16-bit input buffer, **pd_x0(in)**, and a 16-bit output buffer, **pd_x0(out)**. Two output pins, parallel input buffer full (PIBF) and parallel output buffer empty (POBE), indicate the state of the buffers. In addition, there are two registers used to control and monitor the PHIF's operation: the parallel host interface control register (**phifc**, see Table 32), and the PHIF status register (PSTAT, see Table 8). The PSTAT register, which reflects the state of the PIBF and POBE flags, can only be read by an external device when the PSTAT input pin is asserted. The **phifc** register defines the programmable options for this port.

The function of the pins, PIDS and PODS, is programmable to support both the *Intel* and *Motorola* protocols. The pin, PCSN, is an input that, when low, enables PIDS and PODS (or PRWN and PDS, depending on the protocol used). While PCSN is high, the DSP1628 ignores any activity on PIDS and/or PODS. If a DSP1628 is intended to be continuously accessed through the PHIF port, PCSN should be grounded. If PCSN is low and their respective bits in the **inc** register are set, the assertion of PIDS and PODS by an external device causes the DSP1628 device to recognize an interrupt.

Programmability

The parallel host interface can be programmed for 8-bit or 16-bit data transfers using bit 0, PMODE, of the **phifc** register. Setting PMODE selects 16-bit transfer mode. An input pin controlled by the host, PBSEL, determines an access of either the high or low bytes. The assertion level of the PBSEL input pin is configurable in software using bit 3 of the **phifc** register, PBSELF. Table 7 summarizes the port's functionality as controlled by the PSTAT and PBSEL pins and the PBSELF and PMODE fields.

For 16-bit transfers, if PBSELF is zero, the PIBF and POBE flags are set after the high byte is transferred. If PBSELF is one, the flags are set after the low byte is transferred. In 8-bit mode, only the low byte is accessed, and every completion of an input or output access sets PIBF or POBE.

Bit 1 of the **phifc** register, PSTROBE, configures the port to operate either with an *Intel* protocol where only the chip select (PCSN) and either of the data strobes (PIDS or PODS) are needed to make an access, or with a *Motorola* protocol where the chip select (PCSN), a data strobe (PDS), and a read/write strobe (PRWN) are needed. PIDS and PODS are negative assertion data strobes while the assertion level of PDS is programmable through bit 2, PSTRB, of the **phifc** register.

4 Hardware Architecture (continued)

Finally, the assertion level of the output pins, PIBF and POBE, is controlled through bit 4, PFLAG. When PFLAG is set low, PIBF and POBE output pins have positive assertion levels. By setting bit 5, PFLAGSEL, the logical OR of PIBF and POBE flags (positive assertion) is seen at the output pin PIBF. By setting bit 7 in **phifc**, PSOBEF, the polarity of the POBE flag in the status register, PSTAT, can be changed. PSOBEF has no effect on the POBE pin.

Pin Multiplexing

Please refer to Pin Multiplexing in Section 4.1 for a description of BIO, PHIF, VEC[3:0], and SIO2 pins.

Table 7. PHIF Function (8-bit and 16-bit Modes)

| PMODE Field | PSTAT Pin | PBSEL Pin | PBSELF Field = 0 | PBSELF Field = 1 |
|-------------|-----------|-----------|------------------|------------------|
| 0 (8-bit) | 0 | 0 | pdx0 low byte | reserved |
| 0 | 0 | 1 | reserved | pdx0 low byte |
| 0 | 1 | 0 | PSTAT | reserved |
| 0 | 1 | 1 | reserved | PSTAT |
| 1 (16-bit) | 0 | 0 | pdx0 low byte | pdx0 high byte |
| 1 | 0 | 1 | pdx0 high byte | pdx0 low byte |
| 1 | 1 | 0 | PSTAT | reserved |
| 1 | 1 | 1 | reserved | PSTAT |

Table 8. pstat Register as Seen on PB[7:0]

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|----------|---|---|---|---|---|------|------|
| Field | RESERVED | | | | | | PIBF | POBE |

4.9 Bit Input/Output Unit (BIO)

The BIO controls the directions of eight bidirectional control I/O pins, IOBIT[7:0]. If a pin is configured as an output, it can be individually set, cleared, or toggled. If a pin is configured as an input, it can be read and/or tested.

The lower half of the **sbit** register (see Table 37) contains current values (VALUE[7:0]) of the eight bidirectional pins IOBIT[7:0]. The upper half of the **sbit** register (DIREC[7:0]) controls the direction of each of the pins. A logic 1 configures the corresponding pin as an output; a logic 0 configures it as an input. The upper half of the **sbit** register is cleared upon reset.

The **cbit** register (see Table 38) contains two 8-bit fields, MODE/MASK[7:0] and DATA/PAT[7:0]. The values of DATA/PAT[7:0] are cleared upon reset. The meaning of a bit in either field depends on whether it has been configured as an input or an output in **sbit**. If a pin has been configured to be an output, the meanings are MODE and DATA. For an input, the meanings are MASK and PAT(tern). Table 9 shows the functionality of the MODE/MASK and DATA/PAT bits based on the direction selected for the associated IOBIT pin.

Those bits that have been configured as inputs can be individually tested for 1 or 0. For those inputs that are being tested, there are four flags produced: allt (all true), allf (all false), somet (some true), and somef (some

false). These flags can be used for conditional branch or special instructions. The state of these flags can be saved and restored by reading and writing bits 0 to 3 of the **alf** register (see Table 39).

Table 9. BIO Operations

| DIREC[n]* | MODE/MASK[n] | DATA/PAT[n] | Action |
|------------|--------------|-------------|---------------|
| 1 (Output) | 0 | 0 | Clear |
| 1 (Output) | 0 | 1 | Set |
| 1 (Output) | 1 | 0 | No Change |
| 1 (Output) | 1 | 1 | Toggle |
| 0 (Input) | 0 | 0 | No Test |
| 0 (Input) | 0 | 1 | No Test |
| 0 (Input) | 1 | 0 | Test for Zero |
| 0 (Input) | 1 | 1 | Test for One |

* $0 \leq n \leq 7$.

If a BIO pin is switched from being configured as an output to being configured as an input and then back to being configured as an output, the pin retains the previous output value.

Pin Multiplexing

Please refer to Pin Multiplexing in Section 4.1 for a description of BIO, PHIF, VEC[3:0], and SIO2 pins.

4 Hardware Architecture (continued)

4.10 Timer

The interrupt timer is composed of the **timerc** (control) register, the **timer0** register, the prescaler, and the counter itself. The timer control register (see Table 35, **timerc** Register) sets up the operational state of the timer and prescaler. The **timer0** register is used to hold the counter reload value (or period register) and to set the initial value of the counter. The prescaler slows the clock to the timer by a number of binary divisors to allow for a wide range of interrupt delay periods.

The counter is a 16-bit down counter that can be loaded with an arbitrary number from software. It counts down to 0 at the clock rate provided by the prescaler. Upon reaching 0 count, a vectored interrupt to program address 0x08 is issued to the DSP1628, providing the interrupt is enabled (bit 8 of **inc** and **ins** registers). The counter will then either wait in an inactive state for another command from software, or will automatically repeat the last interrupting period, depending upon the state of the RELOAD bit in the **timerc** register.

When RELOAD is 0, the counter counts down from its initial value to 0, interrupts the DSP1628, and then stops, remaining inactive until another value is written to the **timer0** register. Writing to the **timer0** register causes both the counter and the period register to be written with the specified 16-bit number. When RELOAD is 1, the counter counts down from its initial value to 0, interrupts the DSP1628, automatically reloads the specified initial value from the period register into the counter, and repeats indefinitely. This provides for either a single timed interrupt event or a regular interrupt clock of arbitrary period.

The timer can be stopped and started by software, and can be reloaded with a new period at any time. Its count value, at the time of the read, can also be read by software. Due to pipeline stages, stopping and starting the timer may result in one inaccurate count or prescaled period. When the DSP1628 is reset, the bottom 6 bits of the **timerc** register and the **timer0** register and counter are initialized to 0. This sets the prescaler to $CKO/2^*$, turns off the reload feature, disables timer counting, and initializes the timer to its inactive state. The act of resetting the chip does not cause a timer interrupt. Note that the period register is not initialized on reset.

The T0EN bit of the **timerc** register enables the clock to the timer. When T0EN is a 1, the timer counts down towards 0. When T0EN is a 0, the timer holds its current count.

The PRESCALE field of the **timerc** register selects one of 16 possible clock rates for the timer input clock (see Table 35, **timerc** Register).

Setting the DISABLE bit of the **timerc** register to a logic

1 shuts down the timer and the prescaler for power savings. Setting the TIMERDIS, bit 4, in the **powerc** register has the same effect of shutting down the timer. The DISABLE bit and the TIMERDIS bit are cleared by writing a 0 to their respective registers to restore the normal operating mode.

4.11 Error Correction Coprocessor

The error correction coprocessor (ECCP) performs full Viterbi decoding with single instructions for a wide range of maximum likelihood sequence estimation (MLSE) equalization and convolutional decoding. The ECCP operates in parallel with the DSP core, increasing the throughput rate, and single-instruction Viterbi decoding provides significant code compression required for a single DSP solution for modern digital cellular applications.

System Description

The ECCP is a loosely coupled, programmable, internal coprocessor that operates in parallel with the DSP1600 core. A complete Viterbi decoding for MLSE equalization or convolutional decoding is performed with a single DSP instruction.

The core communicates with the ECCP module via three interface registers. An address register, **ear**, is used to indirectly access the ECCP internal memory-mapped registers. A data register, **edr**, works in concert with the address register to indirectly read from or write to an ECCP internal memory-mapped register addressed by the contents of the address register. After each **edr** access, the contents of the address register is postincremented by one. Upon writing an ECCP op code to instruction register, **eir**, either MLSE equalization, convolutional decoding, a simple traceback operation, or ECCP reset is invoked.

The mode of operation of the ECCP is set up by writing appropriate fields of a memory-mapped control register. In MLSE equalization, the control register may be configured for 2-tap to 6-tap equalization. In convolutional decoding, the control register may be configured for constraint lengths 2 through 7 and code rates 1/1 through 1/6. One of two variants of the soft-decoded output may be programmed, or a hard-decoded output may be chosen.

Usually, convolutional decoding is performed after MLSE equalization. For receiver configuration with MLSE equalization followed by convolutional decoding, a Manhattan branch metric computation for convolutional decoding may be selected by setting a branch metric select bit in the control register.

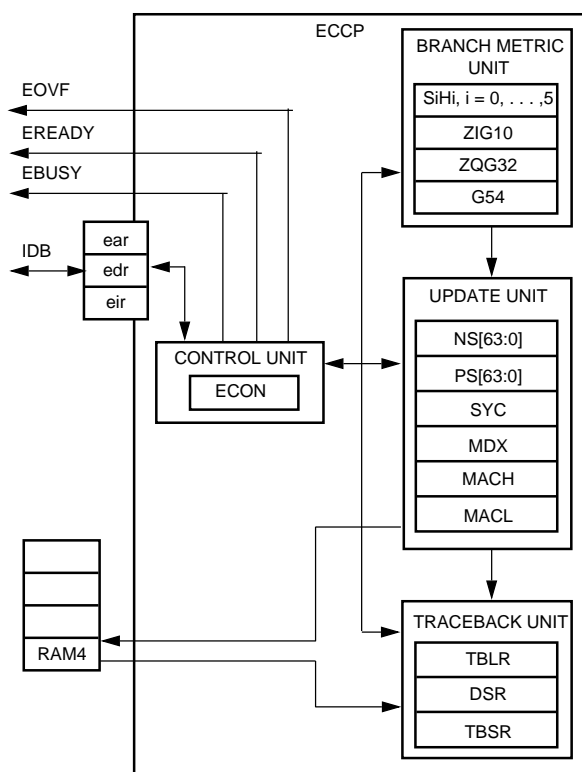
* Frequency of $CKO/2$ is equivalent to either $CKI/2$ for the PLL bypassed or related to CKI by the PLL multiplying factors. See Section 4.13, Clock Synthesis.

4 Hardware Architecture (continued)

In wideband low data rate applications, additive white Gaussian noise (AWGN) is the principle channel impairment, and Euclidean branch metric computation for convolutional decoding is selected by resetting the branch metric select bit to zero.

A traceback-length register is provided for programming the traceback decode length.

A block diagram of the coprocessor and its interface to the DSP1600 core is shown in the following figure:



5-4500 (F)

Figure 7. Error Correction Coprocessor Block Diagram/Programming Model

The ECCP internal registers are accessed indirectly through the address and data registers, **ear** and **edr**. The control register, **ECON**, and the traceback length register, **TBLR**, are used to program the operating mode of the ECCP. The symbol registers (**S0H0**—**S5H5**, **ZIG10**, **ZQG32**), the generating polynomial registers (**ZIG10**, **ZQG32**, **G54**), and the channel impulse registers (**S0H0**—**S5H5**) are used as input to the ECCP for MLSE or convolutional decoding. Following a Viterbi decoding operation, the decoded symbol is read out of the decoded symbol register, **DSR**. All internal states of these memory-mapped registers are accessible and controllable by the DSP program. During periods of si-

multaneous DSP-ECCP activity, however, ECCP internal **edr** registers as well as the shared bank of RAM, **RAM4**, are not accessible to the user's DSP code.

Branch Metric Unit: The branch metric unit of the ECCP performs full-precision real and complex arithmetic for computing 16-bit incremental branch metrics required for MLSE equalization and convolutional decoding.

MLSE Branch Metric Unit: To generate the estimated received complex signal at instance n , $E(n, k) = EI(n, k) + j EQ(n, k)$, at the receiver, all possible states, $k = 0$ to $2^C - 1$, taking part in the Viterbi state transition are convolved with the estimated channel impulse response, $H(n) = [h(n), h(n - 1), h(n - 2), \dots, h(n - C + 1)] T$, where the constraint length $C = \{2 \text{ to } 6\}$. Each in-phase and quadrature-phase part of the channel tap, $h(n) = hI(n) + j hQ(n)$, is quantized to an 8-bit 2's complement number.

The channel estimates are normalized prior to loading into the ECCP such that the worst-case summation of the $hI(n)$ or $hQ(n)$ are confined within a 10-bit 2's complement number. The in-phase and quadrature-phase parts of the received complex signal $Z(n) = ZI(n) + j ZQ(n)$ are also confined within a 10-bit 2's complement number. The Euclidean branch metric associated with each of the 2^C state transitions is calculated as:

$$BM(n, k) = XI(n, k)^2 + XQ(n, k)^2$$

where

$$XI(n, k) = \text{abs}\{ZI(n) - EI(n, k)\}$$

and

$$XQ(n, k) = \text{abs}\{ZQ(n) - EQ(n, k)\}$$

The absolute values of the difference signal are saturated at level **0xFF**. The sixteen most significant bits of this 17-bit incremental branch metric are retained for the add-compare-select operation of the Viterbi algorithm.

The in-phase and quadrature-phase parts of the received complex signal are stored in **ZIG10** and **ZQG32** registers, respectively. The complex estimated channel taps **H5** through **H0** are stored in **S5H5** through **S0H0** registers, such that the in-phase part of the channel occupies the upper byte and the quadrature-phase part of the channel occupies the lower byte.

Convolutional Branch Metric Unit: Two types of distance computation are implemented for convolutional decoding. Convolutional decoding over a Gaussian channel is supported with Euclidean distance measure for rate 1/1 and 1/2 convolutional encoding. Convolutional decoding preceded by the MLSE equalization or other linear/nonlinear equalization is supported with Manhattan distance measure for rate 1/1 through 1/6 convolutional encoding.

4 Hardware Architecture (continued)

Generating polynomials, $G(0), \dots, G(5)$, up to six-delays corresponding to a constraint length of seven, may take part in computing the estimated received signals, $E(0, k), \dots, E(5, k)$, within the ECCP associated with all possible state transitions, $k = 0, 1, 2C - 1$.

Six 8-bit soft symbols, $S(0), \dots, S(5)$, are loaded into the ECCP. The incremental branch metrics associated with all $2C$ state transitions are calculated as indicated in Table 10:

Table 10. Incremental Branch Metrics

| Distance Measure | Code Rate | 16-bit Incremental Branch Metric |
|------------------|------------|---|
| Euclidean | 1/1 | $(S(0) - E(0))^2$ |
| Euclidean | 1/2 | $[\sum(S(i) - E(i))^2] \gg 1, i = 0, 1$ |
| Manhattan | 1/1 | $[S(i) - E(i)] \ll 8, i = 0$ |
| Manhattan | 1/2 | $[(S(i) - E(i))] \ll 7, i = 0, 1$ |
| Manhattan | 1/3 or 1/4 | $[(S(i) - E(i))] \ll 6, i = 0, 1, 2, \text{ or } 3$ |
| Manhattan | 1/5 or 1/6 | $[(S(i) - E(i))] \ll 5, i = 0, 1, \dots, 4 \text{ or } 5$ |

The received 8-bit signals $S(5)$ through $S(0)$ are stored in the $S5H5$ through $S0H0$ registers. The generating polynomials $G(1)$ and $G(0)$ are stored in the upper and lower bytes of the $ZIG10$ register, respectively. The generating polynomials $G(3)$ and $G(2)$ are stored in the upper and lower bytes of the $ZQG32$ register, respectively. The generating polynomials $G(5)$ and $G(4)$ are stored in the upper and lower bytes of the $G54$ register, respectively.

Update Unit: The add-compare-select operation of the Viterbi algorithm is performed in this unit. At every time instant, there are $2C$ state transitions of which $2C - 1$ state transitions survive. The update unit selects and updates $2C - 1$ surviving sequences in the traceback RAM that consists of the 4th bank of the internal RAM, $RAM4$. The accumulated cost of the path p at the J th instant, $ACC(J, p)$, is the sum of the incremental branch metrics belonging to the path p up to the time instant J :

$$ACC(J, p) = \sum BM(j, p), j = 1, \dots, J$$

The update unit computes and stores full precision 24-bit resolution path metrics of the bit sequence. To assist the detection of a near overflow in the accumulated path cost, an internal vectored interrupt, $EOVF$, is provided.

Traceback Unit: The traceback unit selects a path with the smallest path metric among $2C - 1$ survivor paths at every instant. The last signal of the path corresponding to the maximum likelihood sequence is delivered to the decoder output. The depth of this last signal is programmable at the symbol rate. The traceback decoding starts from the minimum cost index associated with the state with the minimum cost, $\min \{Acc(j, p1), \dots, Acc(j, p2C - 1)\}$. If the end state is known, the traceback decoding may be forced in the direction of the right path by writing the desired end state into the minimum cost index register, $MIDX$.

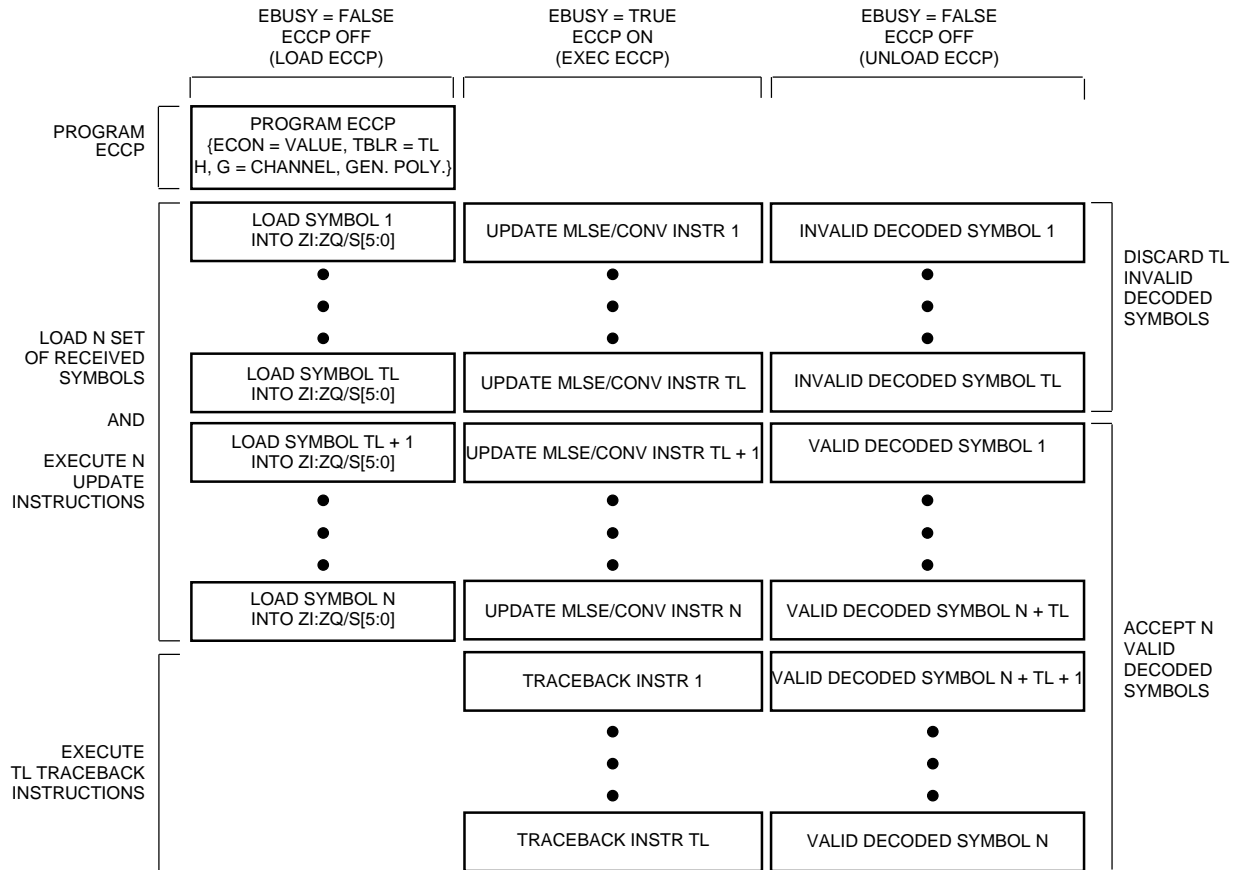
Interrupts and Flags: The ECCP interrupts the DSP1600 core when the ECCP has completed an instruction, $EREADY$, or when an overflow in the accumulated cost is imminent, $EOVF$. Also, an $EBUSY$ flag is provided to the core to indicate when the ECCP is in operation.

Traceback RAM: The fourth 1 Kword bank of dual-port RAM is shared between the DSP1600 core and the ECCP. $RAM4$, located in the Y memory space in the address range $0x0C00$ to $0x0FFF$, is used by the ECCP for storing traceback information. When the ECCP is active, i.e., the $EBUSY$ flag is asserted, the DSP core cannot access this traceback RAM.

4 Hardware Architecture (continued)

DSP Decoding Operation Sequence

The DSP operation sequence for invoking the ECCP for an MLSE equalization or convolutional decoding operation is explained with the operation flow diagram in Figure 7.



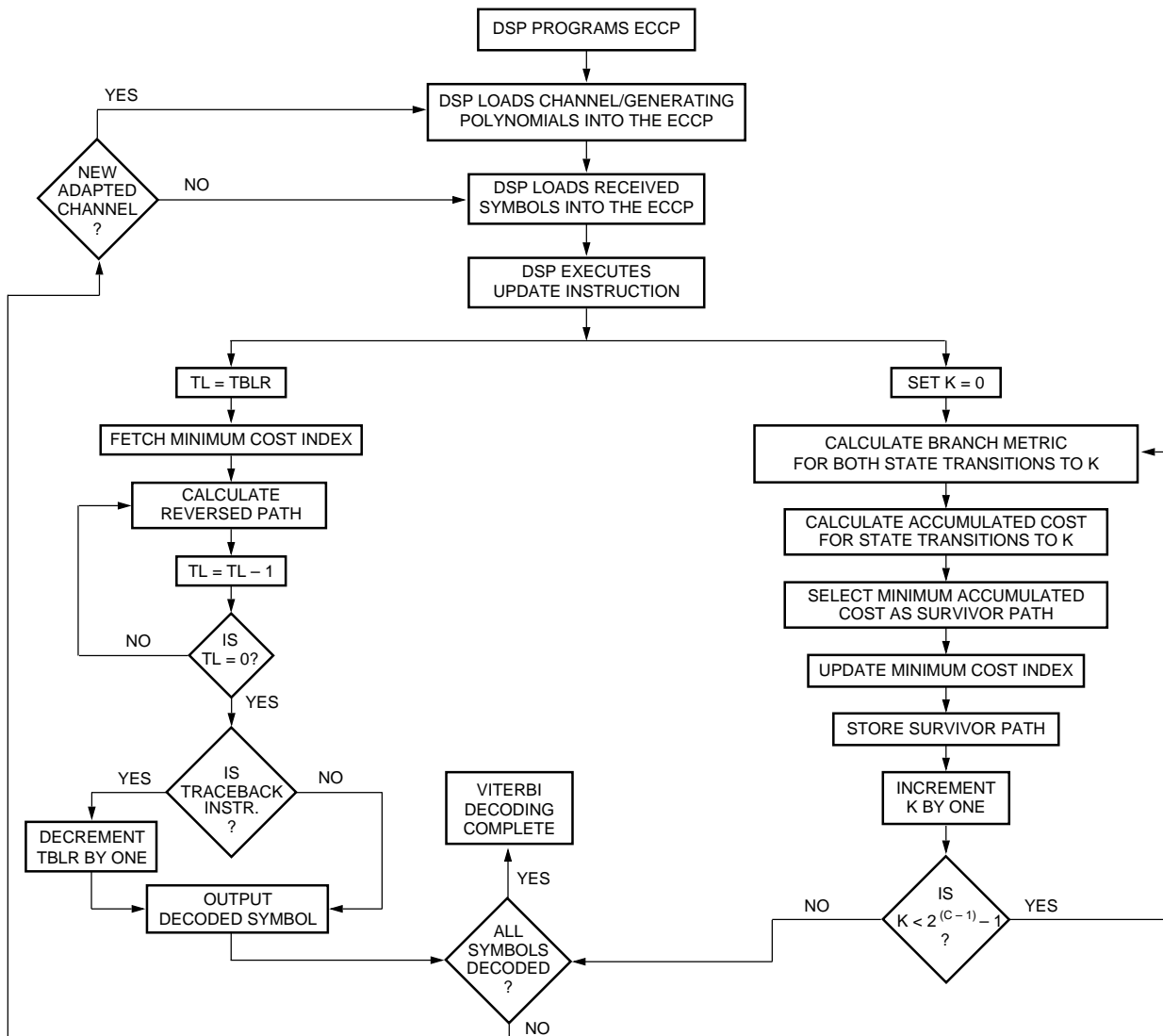
5-4501(F).a

Figure 8. DSP Core Operation Sequence

4 Hardware Architecture (continued)

Operation of the ECCP

To operate the ECCP, the user first programs its mode of operation by setting the control register, ECON, the traceback length register, TBLR, and appropriately initializing the present state accumulated costs. The complete Viterbi decoding operation is achieved by recursively loading the received symbols into the ECCP, executing the ECCP with an UpdateMLSE, an UpdateConv, or a TraceBack instruction, and unloading the decoded symbol from the ECCP. The operation of the ECCP is captured in the signal flow diagram in Figure 8.



5-4502(F)

Figure 9. ECCP Operation Sequence

4 Hardware Architecture (continued)

Software Architecture

The ECCP registers are grouped into two categories: the R-field registers and the internal memory-mapped registers.

R-Field Registers: Three registers (**ear**, **edr**, and **eir**) are defined in the core instruction set as programmable registers for executing the ECCP and establishing the data interface between the ECCP and the core. Reserved bits are always zero when read and should be written with zeros to make the program compatible with future chip revisions.

Address Register (ear**):** The address register holds the address of the ECCP internal memory-mapped registers. Each time the core accesses an internal ECCP register through **edr**, the content of the address register is postincremented by one. During a DSP compound addressing instruction, the same **edr** register is accessed for both the read and the write operation.

Data Register (edr**):** The contents of the ECCP internal memory-mapped registers are indirectly accessed by the DSP through this register. A write to the data register is directed to the ECCP internal register addressed by the contents of the address register. A read from the data register fetches the contents of the ECCP internal register addressed by the address register. Every access to the **edr** autoincrements the address register, **ear**.

Instruction Register (eir**):** Four instructions are defined for the ECCP operation. These instructions will be executed upon writing appropriate values in the **eir** register. Table 11 indicates the instruction encoding and their mnemonics.

Table 11. ECCP Instruction Encoding

| eir Value in hex | Instruction |
|------------------|-------------|
| 0000 | UpdateMLSE |
| 0001 | UpdateConv |
| 0002 | TraceBack |
| 0003 | Reserved |
| 0004 | ResetECCP |
| 0005—FFFF | Reserved |

The UpdateMLSE instruction and the UpdateConv instruction each perform an appropriate branch metric calculation, a complete Viterbi add-compare-select operation, and a concurrent traceback decoding operation. The TraceBack instruction performs the traceback decoding alone.

The ResetECCP instruction performs a proper reset operation to initialize various registers as described in Table 12.

Table 12. Reset State of ECCP Registers

| Register | Reset State |
|----------|--------------------|
| eir | 0x4 |
| — | 0xf (on pin reset) |
| ear | 0x0 |
| SYC | 0x0 |
| ECON | 0x0 |
| MIDX | 0x0 |
| MACH | 0xff |
| MACL | 0xffff |

During periods of ECCP activity, write operations to the **eir** and **edr** registers as well as the read operation of the **edr** register by the DSP code will be blocked. The ECCP address register, **ear**, however, can be read or written during ECCP operation to set up the ECCP address for the next **edr** access after the completion of the ECCP instruction. Note that the **eir** register can be read during ECCP activity.

4 Hardware Architecture (continued)

ECCP Internal Memory-Mapped Registers: Internal memory-mapped registers are defined in the ECCP address space for control and status purposes and to hold data. A summary of the contents of these registers is given in Table 13.

Table 13. Memory-Mapped Registers

| Address | Register | Register Bit Field |
|---------------|--|--|
| 0x0000—0x007F | Next State Register NS[0:63]—24-bit words split across two address locations | Bit 31:16 is addressed by even address. Bit 31:24 zero. Bit 23:16 most significant byte of path cost. |
| 0x0080—0x01FF | Reserved | Bit 15: 0 is addressed by odd address. Bit 15:0 lower 2 bytes of path cost. |
| 0x0200—0x027F | Present State Register PS[0:63]—24-bit words split across two address locations | Bit 31:16 is addressed by even address. Bit 31:24 zero. Bit 23:16 most significant byte of path cost. |
| 0x0280—0x03FF | Reserved | Bit 15:0 is addressed by odd address. Bit 15:0 lower 2 bytes of path cost. |
| 0x400 | Current Symbol Pointer SYC | Bit 5:0 is used. Bit 15:6 reserved. |
| 0x401 | Control Register ECON | Bit 0 is soft decision select. Bit 1 is Manhattan/Euclidean branch metric select. Bit 2 is soft/hard decision select. Bit 3 is reserved. Bit 7:4 is reserved. Bit 10:8 is code rate select. Bit 11 is reserved. Bit 14:12 is constraint length select. Bit 15 is reserved. |
| 0x402 | Traceback Length Register TBLR | Bit 5:0 is used. Bit 15:6 is reserved. |
| 0x403 | Received Symbol/Channel Tap Register S5H5 | Convolutional decoding case: Bit 7:0 is reserved. Bit 15:8 is S5. MLSE equalization case: Bit 7:0 is HQ5. Bit 15:8 is HI5. |
| 0x404 | Received Symbol/Channel Tap Register S4H4 | Convolutional decoding case: Bit 7:0 is reserved. Bit 15:8 is S4. MLSE equalization case: Bit 7:0 is HQ4. Bit 15:8 is HI4. |
| 0x405 | Received Symbol/Channel Tap Register S3H3 | Convolutional decoding case: Bit 7:0 is reserved. Bit 15:8 is S3. MLSE equalization case: Bit 7:0 is HQ3. Bit 15:8 is HI3. |
| 0x406 | Received Symbol/Channel Tap Register S2H2 | Convolutional decoding case: Bit 7:0 is reserved. Bit 15:8 is S2. MLSE equalization case: Bit 7:0 is HQ2. Bit 15:8 is HI2. |

4 Hardware Architecture (continued)

Table 13. Memory-Mapped Registers (continued)

| Address | Register | Register Bit Field |
|-------------|--|--|
| 0x407 | Received Symbol/Channel Tap Register S1H1 | Convolutional decoding case: Bit 7:0 is reserved. Bit 15:8 is S1. MLSE equalization case: Bit 7:0 is HQ1. Bit 15:8 is H11. |
| 0x408 | Received Symbol/Channel Tap Register S0H0 | Convolutional decoding case: Bit 7:0 is reserved. Bit 15:8 is S0. MLSE equalization case: Bit 7:0 is HQ0. Bit 15:8 is H10. |
| 0x409 | Decoded Symbol Register DSR | Bit 7:0 is zero. Bit 15:8 is decoded symbol. |
| 0x40a | Received Real Signal/Generating Polynomial ZIG10 | Convolutional case: Bit 7:0 is G0. Bit 15:8 is G1. MLSE case: Bit 9:0 is in-phase part of received signal. Bit 15:10 is reserved. |
| 0x40b | Received Imaginary Signal/Generating Polynomial ZQG32 | Convolutional case: Bit 7:0 is G2. Bit 15:8 is G3. MLSE case: Bit 9:0 is quadrature-phase part of received signal. Bit 15:10 is reserved. |
| 0x40c | Generating Polynomial G54 | Convolutional case: Bit 7:0 is G4. Bit 15:8 is G5. MLSE case: Bit 15:0 is reserved. |
| 0x40d | Minimum Cost Index Register MIDX | Bit 7:0 is used. Bit 15:8 is reserved. |
| 0x40e—f | Minimum Accumulated Cost Register MACH MACL | 0x040E Bit 15:8 is zero. Bit 7:0 is upper byte of the minimum accumulated cost 0x040F. Bit 15:0 is the lower 2 bytes of the minimum accumulated cost. |
| 0x410 | Traceback Shift Register TBSR | Traceback shift register (TBSR) Bit 7:0 TBSR. Bit 15:8 is reserved. |
| 0x411—0x7FF | Reserved Registers | Reserved. |

4 Hardware Architecture (continued)

4.12 JTAG Test Port

The DSP1628 uses a JTAG/IEEE 1149.1 standard five-wire test port (TDI, TDO, TCK, TMS, TRST) for self-test and hardware emulation. An instruction register, a boundary-scan register, a bypass register, and a device identification register have been implemented. The device identification register coding for the DSP1628 is shown in Table 41. The instruction register (IR) is 4 bits long. The instruction for accessing the device ID is 0xE (1110). The behavior of the instruction register is summarized in Table 14. Cell 0 is the LSB (closest to TDO).

Table 14. JTAG Instruction Register

| IR Cell #: | 3 | 2 | 1 | 0 |
|-----------------|---|---|---|---|
| parallel input? | Y | Y | N | N |
| always logic 1? | N | N | N | Y |
| always logic 0? | N | N | Y | N |

The first line shows the cells in the IR that capture from a parallel input in the capture-IR controller state. The second line shows the cells that always load a logic 1 in the capture-IR controller state. The third line shows the cells that always load a logic 0 in the capture-IR controller state. Cell 3 (MSB of IR) is tied to status signal PINT, and cell 2 is tied to status signal JINT. The state of these signals can therefore be captured during capture-IR and shifted out during SHIFT-IR controller states.

Boundary-Scan Register

All of the chip's inputs and outputs are incorporated in a JTAG scan path shown in Table 15. The types of boundary-scan cells are as follows:

- I = input cell
- O = 3-state output cell
- B = bidirectional (I/O) cell
- OE = 3-state control cell
- DC = bidirectional control cell

4 Hardware Architecture (continued)

Table 15. JTAG Boundary-Scan Register

Note: The direction of shifting is from TDI to cell 104 to cell 103 . . . to cell 0 of TDO.

| Cell | Type | Signal Name/Function | Cell | Type | Signal Name/Function |
|-------|------|---|------|------|----------------------|
| 0 | OE | Controls cells 1, 27—31 | 69 | B | OCK2/PCSN* |
| 1 | O | CKO | 70 | DC | Controls cell 71 |
| 2 | I | RSTB | 71 | B | DO2/PSTAT* |
| 3 | DC | Controls cell 4 | 72 | DC | Controls cell 73 |
| 4 | B | TRAP | 73 | B | SYNC2/PBSEL* |
| 5 | I | STOP† | 74 | DC | Controls cell 75 |
| 6 | O | IACK | 75 | B | ILD2/PIDS* |
| 7 | I | INT0 | 76 | DC | Controls cell 77 |
| 8 | OE | Controls cells 6, 10—25, 49, 50, 78, 79 | 77 | B | OLD2/PODS* |
| 9 | I | INT1 | 78 | O | IBF2/PIBF* |
| 10—25 | O | AB[0:15] | 79 | O | OBE2/POBE* |
| 26 | I | EXM | 80 | DC | Controls cell 81 |
| 27 | O | RWN | 81 | B | ICK2/PB0* |
| 28—31 | O | EROM, ERAMLO, ERAMHI, IO | 82 | DC | Controls cell 83 |
| 32—36 | B | DB[0:4] | 83 | B | DI2/PB1* |
| 37 | DC | Controls cells 32—36, 38—48 | 84 | DC | Controls cell 85 |
| 38—48 | B | DB[5:15] | 85 | B | DOEN2/PB2* |
| 49 | O | OBE1 | 86 | DC | Controls cell 87 |
| 50 | O | IBF1 | 87 | B | SADD2/PB3* |
| 51 | I | DI1 | 88 | DC | Controls cell 89 |
| 52 | DC | Controls cell 53 | 89 | B | IOBIT0/PB4* |
| 53 | B | ILD1 | 90 | DC | Controls cell 91 |
| 54 | DC | Controls cell 55 | 91 | B | IOBIT1/PB5* |
| 55 | B | ICK1 | 92 | DC | Controls cell 93 |
| 56 | DC | Controls cell 57 | 93 | B | IOBIT2/PB6* |
| 57 | B | OCK1 | 94 | DC | Controls cell 95 |
| 58 | DC | Controls cell 59 | 95 | B | IOBIT3/PB7* |
| 59 | B | OLD1 | 96 | DC | Controls cell 97 |
| 60 | OE | Controls cell 61 | 97 | B | VEC3/IOBIT4* |
| 61 | O | DO1 | 98 | DC | Controls cell 99 |
| 62 | DC | Controls cell 63 | 99 | B | VEC2/IOBIT5* |
| 63 | B | SYNC1 | 100 | DC | Controls cell 101 |
| 64 | DC | Controls cell 65 | 101 | B | VEC1/IOBIT6* |
| 65 | B | SADD1 | 102 | DC | Controls cell 103 |
| 66 | DC | Controls cell 67 | 103 | B | VEC0/IOBIT7* |
| 67 | B | DOEN1 | 104 | I | CKI‡ |
| 68 | DC | Controls cell 69 | — | — | — |

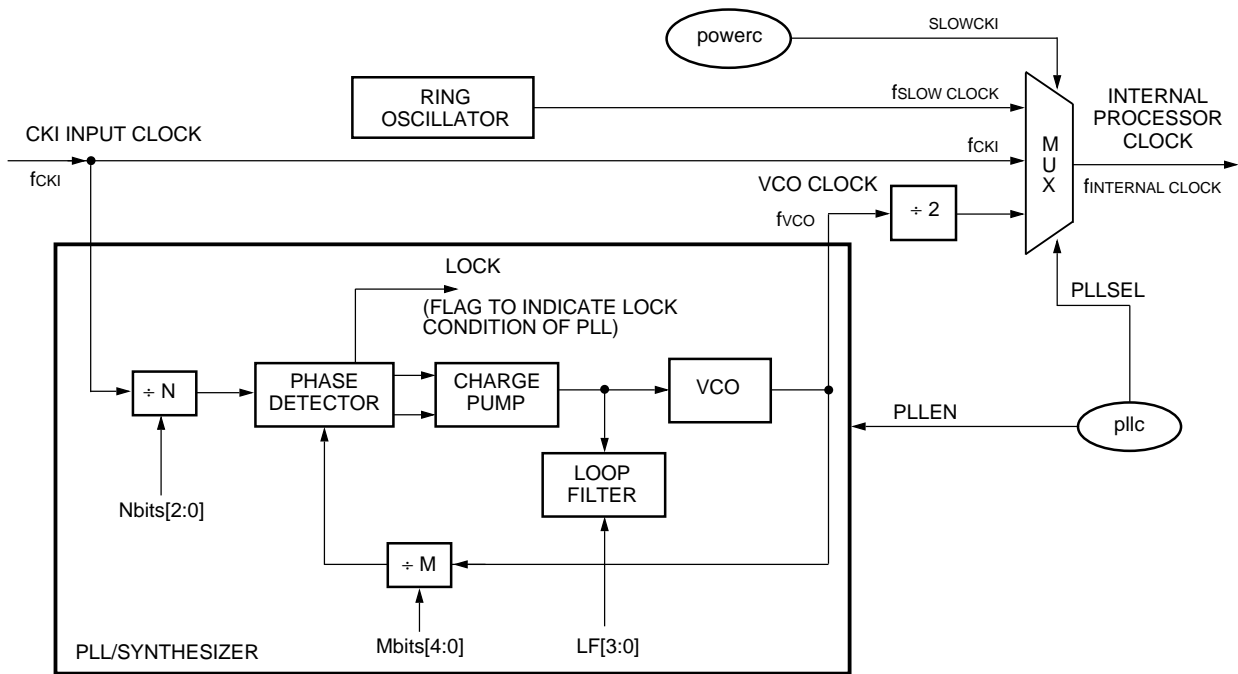
* Please refer to Pin Multiplexing in Section 4.1 for a description of pin multiplexing of BIO, PHIF, VEC[3:0], and SIO2.

† Note that shifting a zero into this cell in the mode to scan a zero into the chip will disable the processor clocks just as the STOP pin will.

‡ When the JTAG SAMPLE instruction is used, this cell will have a logic one regardless of the state of the pin.

4 Hardware Architecture (continued)

4.13 Clock Synthesis



5-4520 (F)

Figure 10. Clock Source Block Diagram

The DSP1628 provides an on-chip, programmable clock synthesizer. Figure 10 is the clock source diagram. The 1X CKI input clock, the output of the synthesizer, or a slow internal ring oscillator can be used as the source for the internal DSP clock. The clock synthesizer is based on a phase-locked loop (PLL), and the terms clock synthesizer and PLL are used interchangeably.

On powerup, CKI is used as the clock source for the DSP. This clock is used to generate the internal processor clocks and CKO, where $f_{CKI} = f_{CKO}$. Setting the appropriate bits in the **pllC** control register (described in Table 36) will enable the clock synthesizer to become the clock source. The **powerc** register, which is discussed in Section 4.14, can override the selection to stop clocks or force the use of the slow clock for low-power operation.

PLL Control Signals

The input to the PLL comes from one of the three mask-programmable clock options: CMOS, or small-signal. The PLL cannot operate without an external input clock.

To use the PLL, the PLL must first be allowed to stabilize and lock to the programmed frequency. After the PLL has locked, the LOCK flag is set and the lock detect circuitry is disabled. The synthesizer can then be used as the clock source. Setting the PLLSEL bit in the **pllC** register will switch sources from f_{CKI} to $f_{VCO}/2$ without glitching. It is important to note that the setting of the **pllC** register must be maintained. Otherwise, the PLL will seek the new set point. Every time the **pllC** register is written, the LOCK flag is reset.

4 Hardware Architecture (continued)

The frequency of the PLL output clock, f_{VCO} , is determined by the values loaded into the 3-bit N divider and the 5-bit M divider. When the PLL is selected and locked, the frequency of the internal processor clock is related to the frequency of CKI by the following equations:

$$f_{VCO} = f_{CKI} * M/N$$

$$f_{INTERNAL\ CLOCK} = f_{CKO} = f_{VCO} \div 2$$

The frequency of the VCO, f_{VCO} , must fall within the range listed in Table 63. Also note that f_{VCO} must be at least twice f_{CKI} .

The coding of the Mbits and Nbits is described as follows:

$$Mbits = M - 2$$

if (N = 1)

$$Nbits = 0x7$$

else

$$Nbits = N - 2$$

where N ranges from 1 to 8 and M ranges from 2 to 20.

The loop filter bits LF[3:0] should be programmed according to Table 64.

Two other bits in the **pllc** register control the PLL. Clearing the PLEN bit powers down the PLL; setting this bit powers up the PLL. Clearing the PLLSEL bit deselects the PLL so that the DSP is clocked by a 1X version of the CKI input; setting the PLLSEL bit selects the PLL-generated clock for the source of the DSP internal processor clock. The **pllc** register is cleared on reset and powerup. Therefore, the DSP comes out of reset with the PLL deselected and powered down. M and N should be changed only while the PLL is deselected. The values of M and N should not be changed when powering down or deselecting the PLL.

As previously mentioned, the PLL also provides a user flag, LOCK, to indicate when the loop has locked. When this flag is not asserted, the PLL output is unstable. The DSP should not be switched to the PLL-based clock without first checking that the lock flag is set. The lock flag is cleared by writing to the **pllc** register. When the PLL is deselected, it is necessary to wait for the PLL to relock before the DSP can be switched to the PLL-based clock. Before the input clock is stopped, the PLL should be powered down. Otherwise, the LOCK flag will not be reset and there may be no way to determine if the PLL is stable, once the input clock is applied again.

The lock-in time depends on the frequency of operation and the values programmed for M and N (see Table 64).

4 Hardware Architecture (continued)

PLL Programming Examples

The following section of code illustrates how the PLL would be initialized on powerup, assuming the following operating conditions:

- CKI input frequency = 10 MHz
- Internal clock and CKO frequency = 50 MHz
- VCO frequency = 100 MHz
- Input divide down count $N = 2$ (Set **Nbits[2:0]** = 000 to get $N = 2$, as described in Table 36.)
- Feedback down count $M = 20$ (Set **Mbits[4:0]** = 10010 to get $M = 18 + 2 = 20$, as described in Table 36.)

The device would come out of reset with the PLL disabled and deselected.

```

pllinit:  pllcr = 0x2912 /* Running CKI input clock at 10 MHz, set up counters in PLL */
          pllcr = 0xA912 /* Power on PLL, but PLL remains deselected */
          call pllwait /* Loop to check for LOCK flag assertion */
          pllcr = 0xE912 /* Select high-speed, PLL clock */
          goto start /* User's code, now running at 50 MHz */
pllwait:  if lock return
          goto pllwait
    
```

Programming examples which illustrate how to use the PLL with the various power management modes are listed in Section 4.14.

Latency

The switch between the CKI-based clock and the PLL-based clock is synchronous. This method results in the actual switch taking place several cycles after the PLLSEL bit is changed. During this time, actual code can be executed, but it will be at the previous clock rate. Table 16 shows the latency times for switching between CKI-based and PLL-based clocks. In the example given, the delay to switch to the PLL source is 1—4 CKO cycles and to switch back is 11—31 CKO cycles.

Table 16. Latency Times for Switching Between CKI and PLL-Based Clocks

| | Minimum Latency (cycles) | Maximum Latency (cycles) |
|-----------------------------|--------------------------|--------------------------|
| Switch to PLL-based clock | 1 | $N + 2$ |
| Switch from PLL-based clock | $M/N + 1$ | $M + M/N + 1$ |

Frequency Accuracy and Jitter

When using the PLL to multiply the input clock frequency up to the instruction clock rate, it is important to realize that although the average frequency of the internal clock and CKO will have about the same relative accuracy as the input clock, noise sources within the DSP will produce jitter on the PLL clock such that each individual clock period will have some error associated with it. The PLL is guaranteed only to have sufficiently low jitter to operate the DSP, and thus, this clock should not be used as an input to jitter-sensitive devices in the system.

VDDA and VSSA Connections

The PLL has its own power and ground pins, VDDA and VSSA. Additional filtering should be provided for VDDA in the form of a ferrite bead connected from VDDA to VDD and two decoupling capacitors (4.7 μ F tantalum in parallel with a 0.01 μ F ceramic) from VDDA to VSS. VSSA can be connected directly to the main ground plane. This recommendation is subject to change and may need to be modified for specific applications depending on the characteristics of the supply noise.

Note: For devices with the CMOS clock input option, the CKI2 pin should be connected to VSSA.

4 Hardware Architecture (continued)

4.14 Power Management

There are three different control mechanisms for putting the DSP1628 into low-power modes: the **powerc** control register, the STOP pin, and the AWAIT bit in the **alf** register. The PLL can also be disabled with the PLEN bit of the **pllc** register for more power saving.

Powerc Control Register Bits

The **powerc** register has 10 bits that power down various portions of the chip and select the clock source:

XTLOFF: Assertion of the XTLOFF bit powers down the small-signal input circuit, disabling the internal processor clock. Since the small-signal input circuit takes many cycles to stabilize, care must be taken with the turn-on sequence, as described later.

SLOWCKI: Assertion of the SLOWCKI bit selects the ring oscillator as the clock source for the internal processor clock instead of CKI or the PLL. When CKI or the PLL is selected, the ring oscillator is powered down. Switching of the clocks is synchronized so that no partial or short clock pulses occur. Two **nops** should follow the instruction that sets or clears SLOWCKI.

NOCK: Assertion of the NOCK bit synchronously turns off the internal processor clock, regardless of whether its source is provided by CKI, the PLL, or the ring oscillator. The NOCK bit can be cleared by resetting the chip with the RSTB pin, or asserting the INT0 or INT1 pins. Two **nops** should follow the instruction that sets NOCK. The PLL remains running, if enabled, while NOCK is set.

INT0EN: This bit allows the INT0 pin to asynchronously clear the NOCK bit, thereby allowing the device to continue program execution from where it left off without any loss of state. No chip reset is required. It is recommended that, when INT0EN is to be used, the INT0 interrupt be disabled in the **inc** register so that an unintended interrupt does not occur. After the program resumes, the INT0 interrupt in the **ins** register should be cleared.

INT1EN: This bit enables the INT1 pin to be used as the NOCK clear, exactly like INT0EN previously described.

The following control bits power down the peripheral I/O units of the DSP. These bits can be used to further reduce the power consumption during standard sleep mode.

SIO1DIS: This is a powerdown signal to the SIO1 I/O unit. It disables the clock input to the unit, thus eliminating any sleep power associated with the SIO1. Since the gating of the clocks may result in incomplete transactions, it is recommended that this option be used in applications where the SIO1 is not used or when reset may be used to reenable the SIO1 unit. Otherwise, the first transaction after reenabling the unit may be corrupted.

SIO2DIS: This bit powers down the SIO2 in the same way SIO1DIS powers down the SIO1.

PHIFDIS: This is a powerdown signal to the parallel host interface. It disables the clock input to the unit, thus eliminating any sleep power associated with the PHIF. Since the gating of the clocks may result in incomplete transactions, it is recommended that this option be used in applications where the PHIF is not used, or when reset may be used to reenable the PHIF. Otherwise, the first transaction after reenabling the unit may be corrupted.

TIMERDIS: This is a timer disable signal which disables the clock input to the timer unit. Its function is identical to the DISABLE field of the **timerc** control register. Writing a 0 to the TIMERDIS field will continue the timer operation.

Figure 11 shows a functional view of the effect of the bits of the **powerc** register on the clock circuitry. It shows only the high-level operation of each bit. Not shown are the bits that power down the peripheral units.

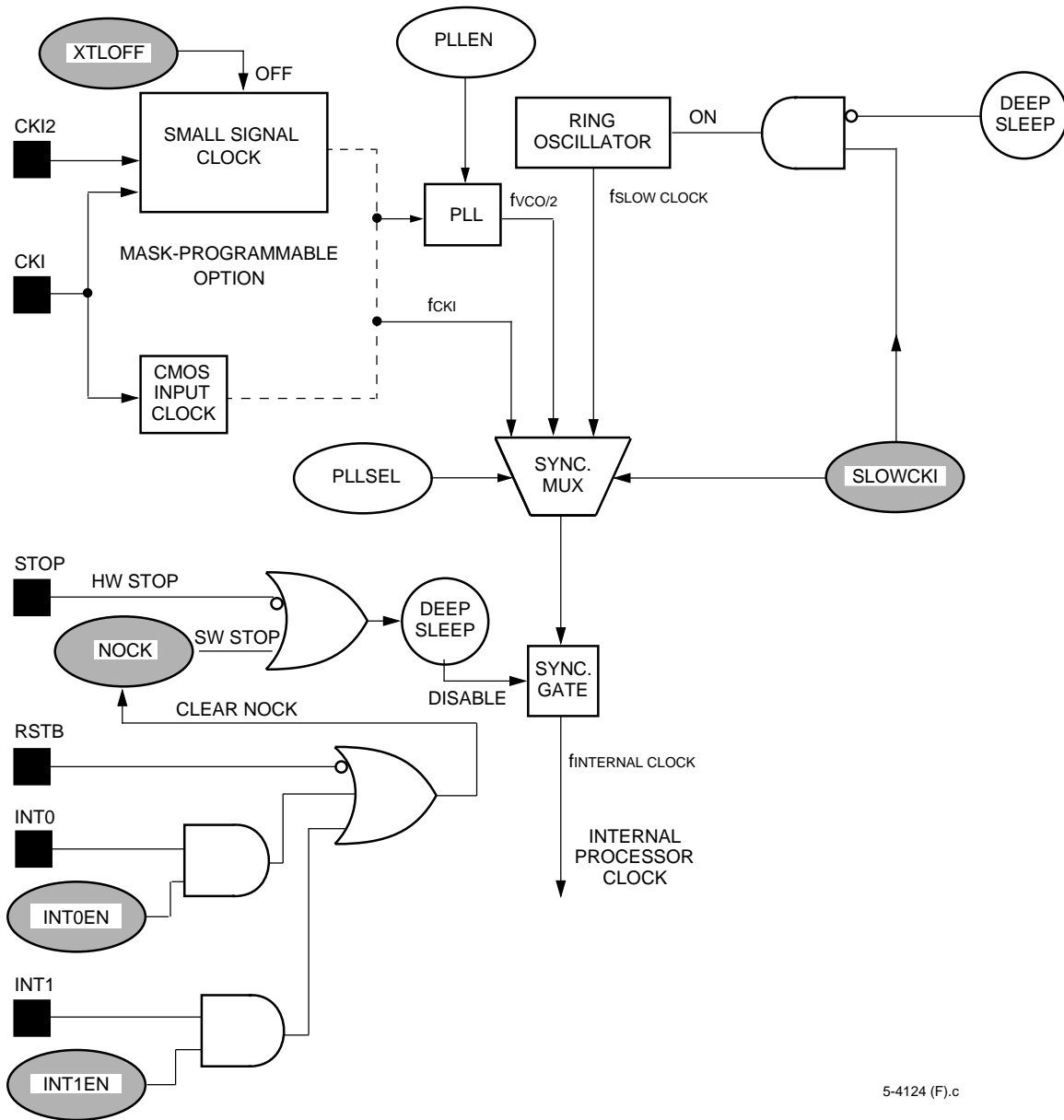
STOP Pin

Assertion (active-low) of the STOP pin has the same effect as setting the NOCK bit in the **powerc** register. The internal processor clock is synchronously disabled until the STOP pin is returned high. Once the STOP pin is returned high, program execution will continue from where it left off without any loss of state. No chip reset is required. The PLL remains running, if enabled, during STOP assertion.

The pllc Register Bits

The PLEN bit of the **pllc** register can be used to power down the clock synthesizer circuitry. Before shutting down the clock synthesizer circuitry, the system clock should be switched to either CKI using the PLLSEL bit of **pllc**, or to the ring oscillator using the SLOWCKI bit of **powerc**.

4 Hardware Architecture (continued)



Notes:
The functions in the shaded ovals are bits in the **powerc** control register. The functions in the nonshaded ovals are bits in the **pllcr** control register.

Deep sleep is the state arrived at either by a hardware or software stop of the internal processor clock.

The switching of the multiplexers and the synchronous gate is designed so that no partial clocks or glitching will occur.

When the deep sleep state is entered with the ring oscillator selected, the internal processor clock is turned off before the ring oscillator is powered down.

PLL select is the PLLSEL bit of **pllcr**; PLL powerdown is the PLEN bit of **pllcr**.

Figure 11. Power Management Using the powerc and the pllcr Registers

4 Hardware Architecture (continued)

Await Bit of the **alf** Register

Setting the AWAIT bit of the **alf** register causes the processor to go into the standard sleep state or power-saving standby mode. Operation of the AWAIT bit is the same as in the DSP1610, DSP1611, DSP1616, DSP1617, and DSP1618. In this mode, the minimum circuitry required to process an incoming interrupt remains active, and the PLL remains active if enabled. An interrupt will return the processor to the previous state, and program execution will continue. The action resulting from setting the AWAIT bit and the action resulting from setting bits in the **powerc** register are mostly independent. As long as the processor is receiving a clock, whether slow or fast, the DSP may be put into standard sleep mode with the AWAIT bit. Once the AWAIT bit is set, the STOP pin can be used to stop and later restart the processor clock, returning to the standard sleep state. If the processor clock is not running, however, the AWAIT bit cannot be set.

Power Management Sequencing

There are important considerations for sequencing the power management modes. The small-signal clock input circuit has a start-up delay which must be taken into account, and the PLL requires a delay to reach lock-in. Also, the chip may or may not need to be reset following a return from a low-power state.

Devices with a small-signal input clocking option may use the XTLOFF bit in the **powerc** register to power down the on-chip oscillator or small-signal circuitry, thereby reducing the power dissipation. When reenabling the oscillator or the small-signal circuitry, it is important to bear in mind that a start-up interval exists during which time the clocks are not stable. Two scenarios exist here:

1. Immediate Turn-Off, Turn-On with RSTB: This scenario applies to situations where the target device is not required to execute any code while the small-signal input circuit is powered down and where restart from a reset state can be tolerated. In this case, the processor clock derived from either the oscillator or the small-signal input is running when XTLOFF is asserted. This effectively stops the internal processor clock. When the system chooses to reenabling the oscillator or small-signal input, a reset of the device will be required. The reset pulse must be of sufficient duration for the small-signal start-up interval to be satisfied (required for the small-signal input circuit to reach its dc operating point). A minimum reset pulse of 20 μ s will be adequate. The falling edge of the reset signal, RSTB, will asynchronously clear the XTLOFF field, thus reenabling the power to the small-signal circuitry. The target DSP will then start execution from a reset state, following the rising edge of RSTB.
2. Running from Slow Clock While XTLOFF Active: The second scenario applies to situations where the device needs to continue execution of its target code. In this case, the device switches to the slow ring oscillator clock first, by enabling the SLOWCKI field. Then, if the small-signal input is being used, power down this circuitry by writing a 1 to the XTLOFF field. Two **nops** are needed in between the two write operations to the **powerc** register. The target device will then continue execution of its code at slow speed, while the small-signal input clock is turned off. Switching from the slow clock back to the high-speed clock is then accomplished in three user steps. First, XTLOFF is cleared. Then, a user-programmed routine sets the internal timer to a delay to wait for the small-signal input oscillations to become stable. When the timer counts down to zero, the high-speed clock is selected by clearing the SLOWCKI field, either in the timer's interrupt service routine or following a timer polling loop. If PLL operation is desired, then an additional routine is necessary to enable the PLL and wait for it to lock.

4 Hardware Architecture (continued)

Power Management Examples Without the PLL

The following examples show the more significant options for reducing the power dissipation. These are valid only if the **pllc** register is set to disable and deselect the PLL (PLLEN = 0, PLLSEL = 0).

Standard Sleep Mode. This is the standard sleep mode. While the processor is clocked with a high-speed clock, CKI, the **alf** register's AWAIT bit is set. Peripheral units may be turned off to further reduce the sleep power.

```

    powerc = 0X00F0    /* Turn off peripherals, core running with CKI */
sleep:a0 = 0x8000     /* Set alf register in cache loop if running from */
do 1 {              /* external memory with >1 wait state */
    alf = a0         /* Stop internal processor clock, interrupt circuits */
    nop              /* active */
}
nop                 /* Needed for bedtime execution. Only sleep power */
nop                 /* consumed here until.... interrupt wakes up the device */
cont: . . .         /* User code executes here */
    powerc = 0x0     /* Turn peripheral units back on */

```

Sleep with Slow Internal Clock. In this case, the ring oscillator is selected to clock the processor before the device is put to sleep. This will reduce the power dissipation while waiting for an interrupt to continue program execution.

```

    powerc = 0x40F0    /* Turn off peripherals and select slow clock */
    2*nop             /* Wait for it to take effect */
sleep:a0 = 0x8000     /* Set alf register in cache loop if running from */
do 1 {              /* external memory with >1 wait state */
    alf = a0         /* Stop internal processor clock, interrupt circuits */
    nop              /* active */
}
nop                 /* Needed for bedtime execution. Reduced sleep power */
nop                 /* consumed here.... Interrupt wakes up the device */
cont: . . .         /* User code executes here */
    powerc = 0x00F0    /* Select high-speed clock */
    2*nop             /* Wait for it to take effect */
    powerc = 0x0000    /* Turn peripheral units back on */

```

Note that, in this case, the wake-up latency is determined by the period of the ring oscillator clock.

Sleep with Slow Internal Clock and Small-Signal Disabled. If the target device contains the small-signal clock option, the clock input circuitry can be powered down to further reduce power. In this case, the slow clock must be selected first.

```

    powerc = 0x40F0    /* Turn off peripherals and select slow clock */
    2*nop             /* Wait for it to take effect */
    powerc = 0xC0F0    /* Turn off the small-signal input buffer */
sleep:a0 = 0x8000     /* Set alf register in cache loop if running from */
do 1 {              /* external memory with >1 wait state */
    alf = a0         /* Stop internal processor clock, interrupt circuits */
    nop              /* active */
}
nop                 /* Needed for bedtime execution. Reduced sleep power */
nop                 /* consumed here.... Interrupt wakes up the device */
    powerc = 0x40F0    /* Clear XTLOFF, reenable small-signal */
    call xtlwait      /* Wait until small-signal is stable */
cont: powerc = 0x00F0    /* Select high-speed clock */
    2*nop             /* Wait for it to take effect */
    powerc = 0x0000    /* Turn peripheral units back on */

```

Note that, in this case, the wake-up latency is dominated by the small-signal start-up period.

4 Hardware Architecture (continued)

Software Stop. In this case, all internal clocking is disabled. INT0, INT1, or RSTB may be used to reenble the clocks. The power management must be done in correct sequence.

```

    powerc = 0x4000    /* SLOWCKI asserted */
    2*nop              /* Wait for it to take effect */
    powerc = 0xD000    /* XTLOFF asserted if applicable and INTOEN asserted */
    inc = NOINT0       /* Disable the INT0 interrupt */
sopor:powerc = 0xF000 /* NOCK asserted, all clocks stop */
                    /* Minimum switching power consumed here */
    3*nop              /* Some nops will be needed */
                    /* INTO pin clears the NOCK field, clocking resumes */
cont: powerc = 0x4000 /* INTOEN cleared and XTLOFF cleared, if applicable*/
    call waitxtl      /* Wait for the small-signal to */
                    /* stabilize, if applicable*/
    powerc = 0x0      /* Clear SLOWCKI field, back to high speed */
    2*nop              /* Wait for it to take effect */
    ins = 0x0010      /* Clear the INT0 status bit */

```

In this case also, the wake-up latency is dominated by the small-signal start-up period.

The previous examples do not provide an exhaustive list of options available to the user. Many different clocking possibilities exist for which the target device may be programmed, depending on:

- The clock source to the processor.
- Whether the user chooses to power down the peripheral units.
- The operational state of the small-signal clock input, powered or unpowered.
- Whether the internal processor clock is disabled through hardware or software.
- The combination of power management modes the user chooses.
- Whether or not the PLL is enabled.

An example subroutine for xtlwait follows:

```

xtlwait:    timer0 = 0x2710    /* Load a count of 10,000 into the timer */
            timerc = 0x0010    /* Start the timer with a PRESCALE of two */
            inc = 0x0000       /* Disable the interrupts */
loop1:     a0 = ins            /* Poll the ins register */
            a0 = a0 & 0x0100    /* Check bit 8 (TIME) of the ins register */
            if eq goto loop1    /* Loop if the bit is not set */
            ins = 0x0100       /* Clear the TIME interrupt bit */
            return             /* Return to the main program */

```

4 Hardware Architecture (continued)

Power Management Examples with the PLL

The following examples show the more significant options for reducing power dissipation if operation with the PLL clock synthesizer is desired.

Standard Sleep Mode, PLL Running. This mode would be entered in the same manner as without the PLL. While the input to the clock synthesizer, CKI, remains running, the **alf** register's AWAITS bit is set. The PLL will continue to run and dissipate power. Peripheral units may be turned off to further reduce the sleep power.

```

    powerc = 0x00F0    /* Turn off peripherals, core running with PLL */
sleep:a0 = 0x8000      /* Set alf register in cache loop if running from */
do 1 {                /* external memory with >1 wait state */
    alf = a0           /* Stop internal processor clock, interrupt circuits */
    nop                /* active */
}
nop                    /* Needed for bedtime execution. Only sleep power plus PLL */
nop                    /* power consumed here.... Interrupt wakes up the device */
cont: . . .           /* User code executes here */
    powerc = 0x0      /* Turn peripheral units back on */

```

Sleep with Slow Internal Clock, PLL Running. In this case, the ring oscillator is selected to clock the processor before the device is put to sleep. This will reduce power dissipation while waiting for an interrupt to continue program execution.

```

    powerc = 0x40F0    /* Turn off peripherals and select slow clock */
    2*nop              /* Wait for slow clock to take effect */
sleep:a0 = 0x8000      /* Set alf register in cache loop if running from */
do 1 {                /* external memory with >1 wait state */
    alf = a0           /* Stop internal processor clock, interrupt circuits */
    nop                /* active */
}
nop                    /* Needed for bedtime execution. Reduced sleep power, PLL */
nop                    /* power, and ring oscillator power consumed here... */
cont: . . .           /* Interrupt wakes up the device */
    /* User code executes here */
    powerc = 0x00F0    /* Select high-speed PLL based clock */
    2*nop              /* Wait for it to take effect */
    powerc = 0x0000    /* Turn peripheral units back on */

```

4 Hardware Architecture (continued)

Sleep with Slow Internal Clock and Small-Signal Disabled, PLL Disabled. If the target device contains the small-signal clock option, the clock input circuitry can be powered down to further reduce power. In this case, the slow clock must be selected first, and then the PLL must be disabled, since the PLL cannot run without the clock input circuitry being active.

```

    powerc = 0x40F0    /* Turn off peripherals and select slow clock */
    2*nop              /* Wait for slow clock to take effect */
    pllcl = 0x29F2     /* Disable PLL (assume N = 1, M = 20, LF = 1001) */
    powerc = 0xC0F0    /* Disable small-signal input buffer */
sleep:a0 = 0x8000     /* Set alf register in cache loop if running from */
do 1 {               /* external memory with >1 wait state */
    alf = a0          /* Stop internal processor clock, interrupt circuits */
    nop              /* active */
}
nop                  /* Needed for bedtime execution. Reduced sleep power
nop                  /* consumed here.... Interrupt wakes up device */
    powerc = 0x40F0    /* Clear XTLOFF, leave PLL disabled */
    call xtlwait      /* Wait until small-signal is stable */
    pllcl = 0xE9F2    /* Enable PLL, continue to run off slow clock */
    call pllwait      /* Loop to check for LOCK flag assertion */
cont: powerc = 0x00F0 /* Select high-speed PLL based clock */
    2*nop              /* Wait for it to take effect */
    powerc = 0x0000    /* Turn peripherals back on */

```

Software Stop, PLL Disabled. In this case, all internal clocking is disabled. INT0, INT1, or RSTB may be used to reenable the clocks. The power management must be done in the correct sequence, with the PLL being disabled before shutting down the clock input buffer.

```

    powerc = 0x4000    /* SLOWCKI asserted */
    2*nop              /* Wait for slow clock to take effect */
    pllcl = 0x29F2     /* Disable PLL (assume N = 1, M = 20, LF = 1001) */
    powerc = 0xD000    /* XTLOFF asserted, if applicable and INTOEN
/* asserted */
sopor:powerc = 0xF000 /* NOCK asserted, all clocks stop */
/* Minimum switching power consumed here */
    3*nop              /* Some nops will be needed */
/* INTO pin clears NOCK field, clocking resumes */
cont: powerc = 0x4000 /* INTOEN cleared and XTLOFF cleared, if applicable */
    call xtlwait      /* Wait until small-signal is stable */
/* if applicable */
    pllcl = 0xE9F2    /* Enable PLL, continue to run off slow clock */
    call pllwait      /* Loop to check for LOCK flag assertion */
    powerc = 0x0       /* Select high-speed PLL based clock */
    2*nop              /* Wait for it to take effect */
    ins = 0x0010      /* Clear the INT0 status bit */

```

5 Software Architecture

5.1 Instruction Set

The DSP1628 processor has seven types of instructions: multiply/ALU, special function, control, F3 ALU, BMU, cache, and data move. The multiply/ALU instructions are the primary instructions used to implement signal processing algorithms. Statements from this group can be combined to generate multiply/accumulate, logical, and other ALU functions and to transfer data between memory and registers in the data arithmetic unit. The special function instructions can be conditionally executed based on flags from the previous ALU or BMU operation, the condition of one of the counters, or the value of a pseudorandom bit in the DSP1628 device. Special function instructions perform shift, round, and complement functions. The F3 ALU instructions enrich the operations available on accumulators. The BMU instructions provide high-performance bit manipulation. The control instructions implement the goto and call commands. Control instructions can also be executed conditionally. Cache instructions are used to implement low-overhead loops, conserve program memory, and decrease the execution time of certain multiply/ALU instructions. Data move instructions are used to transfer data between memory and registers or between accumulators and registers. See the *DSP1611/17/18/27 Digital Signal Processor Information Manual* for a detailed description of the instruction set.

The following operators are used in describing the instruction set:

- * 16 x 16-bit → 32-bit multiplication **or** register-indirect addressing when used as a prefix to an address register **or** denotes direct addressing when used as a prefix to an immediate
- + 36-bit addition†
- 36-bit subtraction†
- >> Arithmetic right shift
- >>> Logical right shift
- <<< Arithmetic left shift
- <<<< Logical left shift
- | 36-bit bitwise OR†
- & 36-bit bitwise AND†
- ^ 36-bit bitwise EXCLUSIVE OR†
- : Compound address swapping, accumulator shuffling
- ~ One's complement

† These are 36-bit operations. One operand is 36-bit data in an accumulator; the other operand may be 16, 32, or 36 bits.

Object Code Compatibility

The DSP1628 is object code compatible with the DSP1618 with the following exceptions:

- ECCP user flag, EBUSY, which indicates error correction coprocessor activity, has changed its condition field.

| Condition | | |
|-----------|----------|---------|
| CON | DSP1618 | DSP1628 |
| 11100 | ebusy | lock |
| 11101 | reserved | ebusy |

The EBUSY flag is used in conjunction with the **if CON F2** or **if CON goto/call/return** instructions to monitor the ECCP operation. The object code corresponding to `ifc EBUSY`, for example, must be modified to reflect the change in condition codes. Alternately, the source code can be assembled using DSP1628 development tools.

- The SIO and SIO2 interrupts (IBF, IBF2, OBE, and OBE2) are cleared one instruction cycle AFTER reading or writing the serial data registers (**sd_x[in]**, **sd_x2[in]**, **sd_x[out]**, or **sd_x2[out]**). To account for this added latency, the user must ensure that a single instruction (NOP or any other valid DSP16XX instruction) follows the **sd_x** register read or write instruction prior to exiting an interrupt service routine (via an `ireturn` or `goto pi` instruction) or before checking the **ins** register for the SIO flag status. Adding this instruction ensures that interrupts are not reported incorrectly following an `ireturn` or that stale flags are not read from the **ins** register. Refer to TECHNICAL ADVISORY #23.

Multiply/ALU Instructions

Note that the function statements and transfer statements in Table 17 are chosen independently. Any function statement (F1) can be combined with any transfer statement to form a valid multiply/ALU instruction. If either statement is not required, a single statement from either column also constitutes a valid instruction. The number of cycles to execute the instruction is a function of the transfer column. (An instruction with no transfer statement executes in one instruction cycle.) Whenever **PC**, **pt**, or **rM** is used in the instruction and points to external memory, the programmed number of wait-states must be added to the instruction cycle count. All multiply/ALU instructions require one word of program memory. The no-operation (**nop**) instruction is a special case encoding of a multiply/ALU instruction and executes in one cycle. The assembly-language representation of a **nop** is either **nop** or a single semicolon.

5 Software Architecture (continued)

A single-cycle squaring function is provided in DSP1628. By setting the $X = Y =$ bit in the **auc** register, any instruction that loads the high half of the **y** register also loads the **x** register with the same value. A subsequent instruction to multiply the **x** register and **y** register results in the square of the value being placed in the **p** register. The instruction **a0 = p p = x*y y = *r0++** with the $X = Y =$ bit set to one will read the value pointed to by **r0**, load it to both **x** and **y**, multiply the previously fetched value of **x** and **y**, and transfer the previous product to **a0**. A table of values pointed to by **r0** can thus be squared in a pipeline with one instruction cycle per each value. Multiply/ALU instructions that use $x = X$ transfer statements (such as **a0 = p p = x*y y = *r0++ x = *pt++**) are not recommended for squaring because **pt** will be incremented even though **x** is not loaded from the value pointed to by **pt**. Also, the same conflict wait occurrences from reading the same bank of internal memory or reading from external memory apply, since the X space fetch occurs (even though its value is not used).

Table 17. Multiply/ALU Instructions

| Function Statement | Transfer Statement† | Cycles (Out/In Cache)‡ |
|-------------------------------|----------------------|------------------------|
| $p = x * y$ | $y = Y \quad x = X$ | 2/1 |
| $aD = p \quad p = x * y$ | $y = aT \quad x = X$ | 2/1 |
| $aD = aS + p \quad p = x * y$ | $y[l] = Y$ | 1/1 |
| $aD = aS - p \quad p = x * y$ | $aT[l] = Y$ | 1/1 |
| $aD = p$ | $x = Y$ | 1/1 |
| $aD = aS + p$ | Y | 1/1 |
| $aD = aS - p$ | $Y = y[l]$ | 2/2 |
| $aD = y$ | $Y = aT[l]$ | 2/2 |
| $aD = aS + y$ | $Z:y \quad x = X$ | 2/2 |
| $aD = aS - y$ | $Z:y[l]$ | 2/2 |
| $aD = aS \& y$ | $Z:aT[l]$ | 2/2 |
| $aD = aS y$ | | |
| $aD = aS \wedge y$ | | |
| $aS - y$ | | |
| $aS \& y$ | | |

† The l in $[]$ is an optional argument that specifies the low 16 bits of **aT** or **y**.

‡ Add cycles for:

1. When an external memory access is made in X or Y space and wait-states are programmed, add the number of wait-states.
2. If an X space access and a Y space access are made to the same bank of DPRAM in one instruction, add one cycle.

Note: For transfer statements when loading the upper half of an accumulator, the lower half is cleared if the corresponding CLR bit in the **auc** register is zero. **auc** is cleared by reset.

Table 18. Replacement Table for Multiply/ALU Instructions

| Replace | Value | Meaning |
|------------|----------------------------|---|
| aD, aS, aT | a0, a1 | One of two DAU accumulators. |
| X | *pt++, *pt++i | X memory space location pointed to by pt . pt is postmodified by +1 and i , respectively. |
| Y | *rM, *rM++, *rM--, rM++j | RAM location pointed to by rM ($M = 0, 1, 2, 3$). rM is postmodified by 0, +1, -1, or j , respectively. |
| Z | *rMzp, *rMpz, *rMm2, *rMjk | Read/Write compound addressing. rM ($M = 0, 1, 2, 3$) is used twice. First, postmodified by 0, +1, -1, or j , respectively; and, second, postmodified by +1, 0, +2, or k , respectively. |

5 Software Architecture (continued)

Special Function Instructions

All forms of the special function require one word of program memory and execute in one instruction cycle. (If PC points to external memory, add programmed wait-states.)

| | | |
|---|---|--|
| $aD = aS \gg 1$ $aD = aS \gg 4$ $aD = aS \gg 8$ $aD = aS \gg 16$ | } | Arithmetic right shift (sign preserved) of 36-bit accumulators |
|---|---|--|

- $aD = aS$ — Load destination accumulator from source accumulator
- $aD = -aS$ — 2's complement
- $aD = \sim aS^\dagger$ — 1's complement
- $aD = \text{rnd}(aS)$ — Round upper 20 bits of accumulator
- $aDh = aSh + 1$ — Increment upper half of accumulator (lower half cleared)
- $aD = aS + 1$ — Increment accumulator
- $aD = y$ — Load accumulator with 32-bit **y** register value with sign extend
- $aD = p$ — Load accumulator with 32-bit **p** register value with sign extend

| | | |
|---|---|---|
| $aD = aS \ll 1$ $aD = aS \ll 4$ $aD = aS \ll 8$ $aD = aS \ll 16$ | } | Arithmetic left shift (sign not preserved) of the lower 32 bits of accumulators (upper 4 bits are sign-bit-extended from bit 31 at the completion of the shift) |
|---|---|---|

The above special functions can be conditionally executed, as in:

```
if CON instruction
```

and with an event counter

```
ifc CON instruction
```

which means:

```
if CON is true then
    c1 = c1 + 1
    instruction
    c2 = c1
else
    c1 = c1 + 1
```

The above special function statements can be executed unconditionally by writing them directly, e.g., $a0 = a1$.

Table 19. Replacement Table for Special Function Instructions

| Replace | Value | Meaning |
|----------|--|--|
| aD aS | a0, a1 | One of two DAU accumulators. |
| CON | mi, pl, eq, ne, gt, le, lvs, lvc, mvs, mvc, c0ge, c0lt, c1ge, c1lt, heads, tails, true, false, allt, allf, somet, somef, oddp, evenp, mns1, nmns1, npint, njint, lock, ebusy | See Table 21 for definitions of mnemonics. |

† This function is not available for the DSP16A.

5 Software Architecture (continued)

Control Instructions

All control instructions executed unconditionally execute in two cycles, except **icall** which takes three cycles. Control instructions executed conditionally execute in three instruction cycles. (If **PC**, **pt**, or **pr** point to external memory, add programmed wait-states.) Control instructions executed unconditionally require one word of program memory, while control instructions executed conditionally require two words. Control instructions cannot be executed from the cache.

```
goto JA†
goto pt
call JA†
call pt
icall‡
return      (goto pr)
ireturn     (goto pi)
```

[†] The **goto JA** and **call JA** instructions should not be placed in the last or next-to-last instruction before the boundary of a 4 Kwords page. If the **goto** or **call** is placed there, the program counter will have incremented to the next page and the jump will be to the next page, rather than to the desired current page.

[‡] The **icall** instruction is reserved for development system use.

The above control instructions, with the exception of **ireturn** and **icall**, can be conditionally executed. For example:

```
if le goto 0x0345
```

Table 20. Replacement Table for Control Instructions

| Replace | Value | Meaning |
|---------|---|--|
| CON | mi, pl, eq, ne, gt, le, nlvs, lvc, mvs, mvc, c0ge, c0lt, c1ge, c1lt, heads, tails, true, false, allt, allf, somet, somef, oddp, evenp, mns1, nmns1, npint, njint, lock, ebusy | See Table 21 for definitions of mnemonics. |
| JA | 12-bit value | Least significant 12 bits of absolute address within the same 4 Kwords memory section. |

5 Software Architecture (continued)

Conditional Mnemonics (Flags)

Table 21 lists mnemonics used in conditional execution of special function and control instructions.

Table 21. DSP1628 Conditional Mnemonics

| Test | Meaning | Test | Meaning |
|-------|--|-------|--|
| pl | Result is nonnegative (sign bit is bit 35). ≥ 0 | mi | Result is negative. < 0 |
| eq | Result is equal to 0. $= 0$ | ne | Result is not equal to 0. $\neq 0$ |
| gt | Result is greater than 0. > 0 | le | Result is less than or equal to 0. ≤ 0 |
| lvs | Logical overflow set.* | lvc | Logical overflow clear. |
| mvs | Mathematical overflow set.† | mvc | Mathematical overflow clear. |
| c0ge | Counter 0 greater than or equal to 0. | c0lt | Counter 0 less than 0. |
| c1ge | Counter 1 greater than or equal to 0. | c1lt | Counter 1 less than 0. |
| heads | Pseudorandom sequence bit set. | tails | Pseudorandom sequence bit clear. |
| true | The condition is always satisfied in an if-instruction. | false | The condition is never satisfied in an if-instruction. |
| allt | All True, all BIO input bits tested compared successfully. | allf | All False, no BIO input bits tested compared successfully. |
| somet | Some True, some BIO input bits tested compared successfully. | somef | Some False, some BIO input bits tested did not compare successfully. |
| oddp | Odd Parity, from BMU operation. | evenp | Even Parity, from BMU operation. |
| mns1 | Minus 1, result of BMU operation. | nmns1 | Not Minus 1, result of BMU operation. |
| npint | Not PINT, used by hardware development system. | njint | Not JINT, used by hardware development system. |
| lock | The PLL has achieved lock and is stable. | ebusy | ECCP Busy, indicates error correction coprocessor activity. |

* Result is not representable in the 36-bit accumulators (36-bit overflow).

† Bits 35—31 are not the same (32-bit overflow).

Notes:

Testing the state of the counters (**c0** or **c1**) automatically increments the counter by one.

The heads or tails condition is determined by a randomly set or cleared bit, respectively. The bit is randomly set with a probability of 0.5. A random rounding function can be implemented with either heads or tails. The random bit is generated by a ten-stage pseudorandom sequence generator (PSG) that is updated after either a heads or tails test. The pseudorandom sequence may be reset by writing any value to the **pi** register, except during an interrupt service routine (ISR). While in an ISR, writing to the **pi** register updates the register and does not reset the PSG. If not in an ISR, writing to the **pi** register resets the PSG. (The **pi** register is updated, but will be written with the contents of the PC on the next instruction.) **Interrupts must be disabled when writing to the pi register.** If an interrupt is taken after the **pi** write, but before **pi** is updated with the PC value, the **ireturn** instruction will not return to the correct location. If the RAND bit in the **auc** register is set, however, writing the **pi** register never resets the PSG.

5 Software Architecture (continued)

F3 ALU Instructions

These instructions are implemented in the DSP1600 core. They allow accumulator two-operand operations with either another accumulator, the **p** register, or a 16-bit immediate operand (IM16). The result is placed in a destination accumulator that can be independently specified. All operations are done with the full 36 bits. For the accumulator with accumulator operations, both inputs are 36 bits. For the accumulator with **p** register operations, the **p** register is sign-extended into bits 35—32 before the operation. For the accumulator high with immediate operations, the immediate is sign-extended into bits 35—32 and the lower bits 15—0 are filled with zeros, except for the AND operation, for which they are filled with ones. These conventions allow the user to do operations with 32-bit immediates by programming two consecutive 16-bit immediate operations. The F3 ALU instructions are shown in Table 22.

Table 22. F3 ALU Instructions

Note: The F3 ALU instructions that do not have a destination accumulator are used to set flags for conditional operations, i.e., bit test operations.

| F3 ALU Instructions [†] | |
|----------------------------------|--|
| Cacheable (one-cycle) | Not Cacheable (two-cycle) [‡] |
| aD = aS + aT | aD = aSh + IM16 |
| aD = aS - aT | aD = aSh - IM16 |
| aD = aS & aT | aD = aSh & IM16 |
| aD = aS aT | aD = aSh IM16 |
| aD = aS ^ aT | aD = aSh ^ IM16 |
| aS - aT | aSh - IM16 |
| aS & aT | aSh & IM16 |
| aD = aS + p | aD = aSl + IM16 |
| aD = aS - p | aD = aSl - IM16 |
| aD = aS & p | aD = aSl & IM16 |
| aD = aS p | aD = aSl IM16 |
| aD = aS ^ p | aD = aSl ^ IM16 |
| aS - p | aSl - IM16 |
| aS & p | aSl & IM16 |

[†] If PC points to external memory, add programmed wait-states.

[‡] The **h** and **l** are required notation in these instructions.

F4 BMU Instructions

The bit manipulation unit in the DSP1628 provides a set of efficient bit manipulation operations on accumulators. It contains four auxiliary registers, **ar<0—3>** (**arM**, **M** = 0, 1, 2, 3), two alternate accumulators (**aa0—aa1**), which can be shuffled with the working set, and four flags (oddp, evenp, mns1, and nmns1). The flags are testable by conditional instructions and can be read and written via bits 4—7 of the **alf** register. The BMU also sets the LMI, LEQ, LLV, and LMV flags in the **psw** register:

LMI = 1 if negative (i.e., bit 35 = 1)

LEQ = 1 if zero (i.e., bits 35—0 are 0)

LLV = 1 if (a) 36-bit overflow, or if (b) illegal shift on field width/offset condition

LMV = 1 if bits 31—35 are not the same (32-bit overflow)

The BMU instructions and cycle times follow. (If PC points to external memory, add programmed wait-states.) All BMU instructions require 1 word of program memory unless otherwise noted. Please refer to the *DSP1611/17/18/27 Digital Signal Processor Information Manual* for further discussion of the BMU instructions.

5 Software Architecture (continued)

■ Barrel Shifter

| | |
|------------------------------|--|
| $aD = aS \gg IM16$ | Arithmetic right shift by immediate (36-bit, sign filled in); 2-cycle, 2-word. |
| $aD = aS \gg arM$ | Arithmetic right shift by arM (36-bit, sign filled in); 1-cycle. |
| $aD = \overline{aS} \gg aS$ | Arithmetic right shift by aS (36-bit, sign filled in); 2-cycle. |
| $aD = aS \ggg IM16$ | Logical right shift by immediate (32-bit shift, 0s filled in); 2-cycle, 2-word. |
| $aD = aS \ggg arM$ | Logical right shift by arM (32-bit shift, 0s filled in); 1-cycle. |
| $aD = \overline{aS} \ggg aS$ | Logical right shift by aS (32-bit shift, 0s filled in); 2-cycle. |
| $aD = aS \ll IM16$ | Arithmetic left shift [†] by immediate (36-bit shift, 0s filled in); 2-cycle, 2-word. |
| $aD = aS \ll arM$ | Arithmetic left shift [†] by arM (36-bit shift, 0s filled in); 1-cycle. |
| $aD = \overline{aS} \ll aS$ | Arithmetic left shift [†] by aS (36-bit shift, 0s filled in); 2-cycle. |
| $aD = aS \lll IM16$ | Logical left shift by immediate (36-bit shift, 0s filled in); 2-cycle, 2-word. |
| $aD = aS \lll arM$ | Logical left shift by arM (36-bit shift, 0s filled in); 1-cycle. |
| $aD = \overline{aS} \lll aS$ | Logical left shift by aS (36-bit shift, 0s filled in); 2-cycle. |

† Not the same as the special function arithmetic left shift. Here, the guard bits in the destination accumulator are shifted into, not sign-extended.

■ Normalization and Exponent Computation

| | |
|-----------------------------|---|
| $aD = \text{exp}(aS)$ | Detect the number of redundant sign bits in accumulator; 1-cycle. |
| $aD = \text{norm}(aS, arM)$ | Normalize aS with respect to bit 31, with exponent in arM ; 1-cycle. |

■ Bit Field Extraction and Insertion

| | |
|----------------------------------|--|
| $aD = \text{extracts}(aS, IM16)$ | Extraction with sign extension, field specified as immediate; 2-cycle, 2-word. |
| $aD = \text{extracts}(aS, arM)$ | Extraction with sign extension, field specified in arM ; 1-cycle. |
| $aD = \text{extractz}(aS, IM16)$ | Extraction with zero extension, field specified as immediate; 2-cycle, 2-word. |
| $aD = \text{extractz}(aS, arM)$ | Extraction with zero extension, field specified in arM ; 1-cycle. |
| $aD = \text{insert}(aS, IM16)$ | Bit field insertion, field specified as immediate; 2-cycle, 2-word. |
| $aD = \text{insert}(aS, arM)$ | Bit field insertion, field specified in arM ; 2-cycle. |

Note: The bit field to be inserted or extracted is specified as follows. The width (in bits) of the field is the upper byte of the operand (immediate or **arM**), and the offset from the LSB is in the lower byte.

■ Alternate Accumulator Set

| | |
|---------------|--|
| $aD = aS:aa0$ | Shuffle accumulators with alternate accumulator 0 (aa0); 1-cycle. |
| $aD = aS:aa1$ | Shuffle accumulators with alternate accumulator 1 (aa1); 1-cycle. |

Note: The alternate accumulator gets what was in aS. aD gets what was in the alternate accumulator.

Table 23. Replacement Table for F3 ALU Instructions and F4 BMU Instructions

| Replace | Value | Meaning |
|------------|-----------|--|
| aD, aT, aS | a0 or a1 | One of the two accumulators. |
| IM16 | immediate | 16-bit data, sign-, zero-, or one-extended as appropriate. |
| arM | ar<0—3> | One of the auxiliary BMU registers. |

5 Software Architecture (continued)

Cache Instructions

Cache instructions require one word of program memory. The **do** instruction executes in one instruction cycle, and the **redo** instruction executes in two instruction cycles. (If PC points to external memory, add programmed wait-states.) Control instructions and long immediate values cannot be stored inside the cache. The instruction formats are as follows:

```
do K {
    instr1
    instr2
    .
    .
    .
    instrN
}
redo K
```

Table 24. Replacement Table for Cache Instructions

| Replace | Instruction Encoding | Meaning |
|---------|----------------------|---|
| K | cloop [†] | Number of times the instructions are to be executed taken from bits 0—6 of the cloop register. |
| | 1 to 127 | Number of times the instructions to be executed is encoded in the instruction. |
| N | 1 to 15 | 1 to 15 instructions can be included. |

[†] The assembly-language statement, do **cloop** (or redo **cloop**), is used to specify that the number of iterations is to be taken from the **cloop** register. K is encoded as 0 in the instruction encoding to select **cloop**.

When the cache is used to execute a block of instructions, the cycle timings of the instructions are as follows:

1. In the first pass, the instructions are fetched from program memory and the cycle times are the normal out-of-cache values, except for the last instruction in the block of N instructions. This instruction executes in two cycles.
2. During pass two through pass K – 1, each instruction is fetched from cache and the in-cache timings apply.
3. During the last (Kth) pass, the block of instructions is fetched from cache and the in-cache timings apply, except that the timing of the last instruction is the same as if it were out-of-cache.
4. If any of the instructions access external memory, programmed wait-states must be added to the cycle counts.

The **redo** instruction treats the instructions currently in the cache memory as another loop to be executed K times. Using the **redo** instruction, instructions are reexecuted from the cache without reloading the cache.

The number of iterations, K, for a **do** or **redo** can be set at run time by first moving the number of iterations into the **cloop** register (7 bits unsigned), and then issuing the do **cloop** or redo **cloop**. At the completion of the loop, the value of **cloop** is decremented to 0; hence, **cloop** needs to be written before each do **cloop** or redo **cloop**.

5 Software Architecture (continued)

Data Move Instructions

Data move instructions normally execute in two instruction cycles. (If PC or **rM** point to external memory, any programmed wait-states must be added. In addition, if PC and **rM** point to the same bank of DPRAM, then one cycle must be added.) Immediate data move instructions require two words of program memory; all other data move instructions require only one word. The only exception to these statements is a special case immediate load (short immediate) instruction. If a YAAU register is loaded with a 9-bit short immediate value, the instruction requires only one word of memory and executes in one instruction cycle. All data move instructions, except those doing long immediate loads, can be executed from within the cache. The data move instructions are as follows:

R = IM16
aT[l] = R
SR = IM9
Y = R
R = Y
Z : R
R = aS[l]
DR = *(OFFSET)
*(OFFSET) = DR

Table 25. Replacement Table for Data Move Instructions

| Replace | Value | Meaning |
|---------|---|---|
| R | Any of the registers in Table 55 | — |
| DR | r<0—3>, a0[l], a1[l], y[l], p, pl, x, pt, pr, psw | Subset of registers accessible with direct addressing. |
| aS, aT | a0, a1 | High half of accumulator. |
| Y | *rM, *rM++, *rM--, *rM++j | Same as in multiply/ALU instructions. |
| Z | *rMzp, *rMpz, *rMm2, *rMjk | Same as in multiply/ALU instructions. |
| IM16 | 16-bit value | Long immediate data. |
| IM9 | 9-bit value | Short immediate data for YAAU registers. |
| OFFSET | 5-bit value from instruction 11-bit value in base register | Value in bits [15:5] of ybase register form the 11 most significant bits of the base address. The 5-bit offset is concatenated to this to form a 16-bit address. |
| SR | r<0—3>, rb, re, j, k | Subset of registers for short immediate. |

Notes:

sioc, **sioc2**, **tdms**, **tdms2**, **srtA**, and **srtA2** registers are not readable.

When signed registers less than 16 bits wide (**c0**, **c1**, **c2**) are read, their contents are sign-extended to 16 bits. When unsigned registers less than 16 bits wide are read, their contents are zero-extended to 16 bits.

Loading an accumulator with a data move instruction does not affect the flags.

5 Software Architecture (continued)

5.2 Register Settings

Tables 26 through 42 describe the programmable registers of the DSP1628 device. Table 44 describes the register settings after reset.

Note that the following abbreviations are used in the tables:

- x = don't care
- R = read only
- W = read/write

The reserved (rsrvd) bits in the tables should always be written with zeros to make the program compatible with future chip versions.

Table 26. Serial I/O Control Registers

sioc

| Bit | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|-------|----|-----|---|-----|-----|-----|-----|-----|------|------|
| Field | DODLY | LD | CLK | | MSB | OLD | ILD | OCK | ICK | OLEN | ILEN |

| Field | Value | Description |
|-------|-------|---|
| DODLY | 0 | DO changes on the rising edge of OCK. |
| | 1 | DO changes on the falling edge of OCK. This delay in driving DO increases the hold time on DO by half a cycle of OCK. |
| LD | 0 | In active mode, ILD1 and/or OLD1 = ICK1/16, active SYNC1 = ICK1/[128/256 [†]]. |
| | 1 | In active mode, ILD1 and/or OLD1 = OCK1/16, active SYNC1 = OCK1/[128/256 [†]]. |
| CLK | 00 | Active clock = CKI/2 (1X). |
| | 01 | Active clock = CKI/6 (1X). |
| | 10 | Active clock = CKI/8 (1X). |
| | 11 | Active clock = CKI/10 (1X). |
| MSB | 0 | LSB first. |
| | 1 | MSB first. |
| OLD | 0 | OLD1 is an input (passive mode). |
| | 1 | OLD1 is an output (active mode). |
| ILD | 0 | ILD1 is an input (passive mode). |
| | 1 | ILD1 is an output (active mode). |
| OCK | 0 | OCK1 is an input (passive mode). |
| | 1 | OCK1 is an output (active mode). |
| ICK | 0 | ICK1 is an input (passive mode). |
| | 1 | ICK1 is an output (active mode). |
| OLEN | 0 | 16-bit output. |
| | 1 | 8-bit output. |
| ILEN | 0 | 16-bit input. |
| | 1 | 8-bit input. |

sioc2[‡]

| Bit | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|--------|-----|------|---|------|------|------|------|------|-------|-------|
| Field | DODLY2 | LD2 | CLK2 | | MSB2 | OLD2 | ILD2 | OCK2 | ICK2 | OLEN2 | ILEN2 |

[†] See **tdms** register, SYNC field.

[‡] The bit definitions of the **sioc2** register are identical to the **sioc** register bit definitions.

5 Software Architecture (continued)

Table 27. Time-Division Multiplex Slot Registers

tdms

| | | | | | | | | | | |
|--------------|--------|------|---------------|---|---|---|---|---|---|------|
| Bit | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Field | SYNCSP | MODE | TRANSMIT SLOT | | | | | | | SYNC |

| Field | Value | Description |
|---------------------|----------------|--|
| SYNCSP [†] | 0 [‡] | SYNC1 = ICK1/128 if LD = 0 [†] . SYNC1 = OCK1/128 if LD = 1 [†] . |
| | 1 | SYNC1 = ICK1/256 if LD = 0 [†] . SYNC1 = OCK1/256 if LD = 1 [†] . |
| MODE | 0 | Multiprocessor mode off; DOEN1 is an input (passive mode). |
| | 1 | Multiprocessor mode on; DOEN1 is an output (active mode). |
| TRANSMIT SLOT | 1xxxxxx | Transmit slot 7. |
| | x1xxxxx | Transmit slot 6. |
| | xx1xxxx | Transmit slot 5. |
| | xxx1xxx | Transmit slot 4. |
| | xxxx1xx | Transmit slot 3. |
| | xxxxx1x | Transmit slot 2. |
| SYNC | 1 | Transmit slot 0, SYNC1 is an output (active mode). |
| | 0 | SYNC1 is an input (passive mode). |

tdms2[§]

| | | | | | | | | | | |
|--------------|----------------------|-------|----------------|---|---|---|---|---|---|-------|
| Bit | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Field | SYNCSP2 [†] | MODE2 | TRANSMIT SLOT2 | | | | | | | SYNC2 |

[†] See **sioc** register, LD field.

[‡] Select this mode when in multiprocessor mode.

[§] The **tdms2** register bit definitions are identical to the **tdms** register bit definitions.

5 Software Architecture (continued)

Table 28. Serial Receive/Transmit Address Registers

srta

| | | | | | | | | | | | | | | | | |
|--------------|-----------------|----|----|----|----|----|---|---|------------------|---|---|---|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Field | RECEIVE ADDRESS | | | | | | | | TRANSMIT ADDRESS | | | | | | | |

| Field | Value | Description |
|------------------|----------|---------------------|
| RECEIVE ADDRESS | 1xxxxxxx | Receive address 7. |
| | x1xxxxxx | Receive address 6. |
| | xx1xxxxx | Receive address 5. |
| | xxx1xxxx | Receive address 4. |
| | xxxx1xxx | Receive address 3. |
| | xxxxx1xx | Receive address 2. |
| | xxxxxx1x | Receive address 1. |
| | xxxxxxx1 | Receive address 0. |
| TRANSMIT ADDRESS | 1xxxxxxx | Transmit address 7. |
| | x1xxxxxx | Transmit address 6. |
| | xx1xxxxx | Transmit address 5. |
| | xxx1xxxx | Transmit address 4. |
| | xxxx1xxx | Transmit address 3. |
| | xxxxx1xx | Transmit address 2. |
| | xxxxxx1x | Transmit address 1. |
| | xxxxxxx1 | Transmit address 0. |

srta2[†]

| | | | | | | | | | | | | | | | | |
|--------------|------------------|----|----|----|----|----|---|---|-------------------|---|---|---|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Field | RECEIVE ADDRESS2 | | | | | | | | TRANSMIT ADDRESS2 | | | | | | | |

[†] The **srta2** field definitions are identical to the **srta** register field definitions.

Table 29. Multiprocessor Protocol Registers

saddx

| | | |
|------------------|---------------------------|----------------------------|
| Bit Field | 15—8 | 7—0 |
| Write | X | Write Protocol Field [7:0] |
| Read | Read Protocol Field [7:0] | 0 |

saddx2[‡]

| | | |
|------------------|----------------------------|-----------------------------|
| Bit Field | 15—8 | 7—0 |
| Write | X | Write Protocol2 Field [7:0] |
| Read | Read Protocol2 Field [7:0] | 0 |

[‡] The **saddx2** field definitions are identical to the **saddx** register field definitions.

5 Software Architecture (continued)

Table 30. Processor Status Word (psw) Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|-----------|----|----|----|----|----|-------|-----------|---|---|---|-------|-----------|---|---|---|
| Field | DAU FLAGS | | | | X | X | a1[V] | a1[35:32] | | | | a0[V] | a0[35:32] | | | |

| Field | Value | Description |
|------------|-------|--|
| DAU FLAGS* | Wxxx | LMI—logical minus when set (bit 35 = 1). |
| | xWxx | LEQ—logical equal when set (bit [35:0] = 0). |
| | xxWx | LLV—logical overflow when set. |
| | xxxW | LMV—mathematical overflow when set. |
| a1[V] | W | Accumulator 1 (a1) overflow when set. |
| a1[35:32] | Wxxx | Accumulator 1 (a1) bit 35. |
| | xWxx | Accumulator 1 (a1) bit 34. |
| | xxWx | Accumulator 1 (a1) bit 33. |
| | xxxW | Accumulator 1 (a1) bit 32. |
| a0[V] | W | Accumulator 0 (a0) overflow when set. |
| a0[35:32] | Wxxx | Accumulator 0 (a0) bit 35. |
| | xWxx | Accumulator 0 (a0) bit 34. |
| | xxWx | Accumulator 0 (a0) bit 33. |
| | xxxW | Accumulator 0 (a0) bit 32. |

* The DAU flags can be set by either BMU or DAU operations.

Table 31. Arithmetic Unit Control (auc) Register†

| Bit | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|------|------|-----|---|---|-----|---|-------|---|
| Field | RAND | X=Y= | CLR | | | SAT | | ALIGN | |

| Field | Value | Description |
|-------|-------|---|
| RAND | 0 | Pseudorandom sequence generator (PSG) reset by writing the pi register only outside an interrupt service routine. |
| | 1 | PSG never reset by writing the pi register. |
| X=Y= | 0 | Normal operation. |
| | 1 | All instructions which load the high half of the y register also load the x register, allowing single-cycle squaring with $p = x * y$. |
| CLR | 1xx | Clearing yl is disabled (enabled when 0). |
| | x1x | Clearing a1l is disabled (enabled when 0). |
| | xx1 | Clearing a0l is disabled (enabled when 0). |
| SAT | 1x | a1 saturation on overflow is disabled (enabled when 0). |
| | x1 | a0 saturation on overflow is disabled (enabled when 0). |
| ALIGN | 00 | $a0, a1 \leftarrow p$. |
| | 01 | $a0, a1 \leftarrow p/4$. |
| | 10 | $a0, a1 \leftarrow p \times 4$ (and zeros written to the two LSBs). |
| | 11 | $a0, a1 \leftarrow p \times 2$ (and zero written to the LSB). |

† The **auc** is 9 bits [8:0]. The upper 7 bits [15:9] are always zero when read and should always be written with zeros to make the program compatible with future chip versions. The **auc** register is cleared at reset.

5 Software Architecture (continued)

Table 32. Parallel Host Interface Control (phifc) Register

| Bit | 15—7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|----------|--------|----------|-------|--------|-------|---------|-------|
| Field | Reserved | PSOBEF | PFLAGSEL | PFLAG | PBSELF | PSTRB | PSTROBE | PMODE |

| Field | Value | Description |
|----------|-------|---|
| PMODE | 0 | 8-bit data transfers. |
| | 1 | 16-bit data transfers. |
| PSTROBE | 0 | <i>Intel</i> protocol: PIDS and PODS data strobes. |
| | 1 | <i>Motorola</i> protocol: PRWN and PDS data strobes. |
| PSTRB | 0 | When PSTROBE = 1, PODS pin (PDS) active-low. |
| | 1 | When PSTROBE = 1, PODS pin (PDS) active-high. |
| PBSELF | 0 | In either mode, PBSEL pin = 0 -> pd_x0 low byte. See Table 7. |
| | 1 | If PMODE = 0, PBSEL pin = 1 -> pd_x0 low byte. If PMODE = 1, PBSEL pin = 0 -> pd_x0 high byte. |
| PFLAG | 0 | PIBF and POBE pins active-high. |
| | 1 | PIBF and POBE pins active-low. |
| PFLAGSEL | 0 | Normal. |
| | 1 | PIBF flag ORed with POBE flag and output on PIBF pin; POBE pin unchanged (output buffer empty). |
| PSOBEF | 0 | Normal. |
| | 1 | POBE flag as read through PSTAT register is active-low. |

Table 33. Interrupt Control (inc) Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7—6 | 5—4 | 3 | 2 | 1 | 0 |
|-------|-------|------|--------|------|------|------|------|------|------|----------|------|------|-----|-----|
| Field | JINT* | rsvd | EREADY | EOVF | rsvd | OBE2 | IBF2 | TIME | rsvd | INT[1:0] | PIBF | POBE | OBE | IBF |

* JINT is a JTAG interrupt and is controlled by the HDS. It may be made unmaskable by the Lucent Technologies development system tools.

Encoding: A 0 disables an interrupt; a 1 enables an interrupt.

Table 34. Interrupt Status (ins) Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7—6 | 5—4 | 3 | 2 | 1 | 0 |
|-------|------|------|--------|------|------|------|------|------|------|----------|------|------|-----|-----|
| Field | JINT | rsvd | EREADY | EOVF | rsvd | OBE2 | IBF2 | TIME | rsvd | INT[1:0] | PIBF | POBE | OBE | IBF |

Encoding: A 0 indicates no interrupt. A 1 indicates an interrupt has been recognized and is pending or being serviced. If a 1 is written to bits 4, 5, 8, 12, or 13 of **ins**, the corresponding interrupt is cleared.

5 Software Architecture (continued)

Table 35. timerc Register

| Bit | 15—7 | 6 | 5 | 4 | 3—0 |
|-------|----------|---------|--------|------|----------|
| Field | Reserved | DISABLE | RELOAD | T0EN | PRESCALE |

| Field | Value | Description |
|----------|-------|--|
| DISABLE | 0 | Timer enabled. |
| | 1 | Timer and prescaler disabled. The period register and timer0 are not reset. |
| RELOAD | 0 | Timer stops after counting down to 0. |
| | 1 | Timer automatically reloads and repeats indefinitely. |
| T0EN | 0 | Timer holds current count. |
| | 1 | Timer counts down to 0. |
| PRESCALE | — | See table below. |

PRESCALE Field

| PRESCALE | Frequency of Timer Interrupts | PRESCALE | Frequency of Timer Interrupts |
|----------|-------------------------------|----------|-------------------------------|
| 0000 | CKO/2 | 1000 | CKO/512 |
| 0001 | CKO/4 | 1001 | CKO/1024 |
| 0010 | CKO/8 | 1010 | CKO/2048 |
| 0011 | CKO/16 | 1011 | CKO/4096 |
| 0100 | CKO/32 | 1100 | CKO/8192 |
| 0101 | CKO/64 | 1101 | CKO/16384 |
| 0110 | CKO/128 | 1110 | CKO/32768 |
| 0111 | CKO/256 | 1111 | CKO/65536 |

Table 36. Phase-Locked Loop Control (pll) Register

| Bit | 15 | 14 | 13 | 12 | 11—8 | 7—5 | 4—0 |
|-------|-------|--------|-----|----------|---------|------------|------------|
| Field | PLLEN | PLLSEL | ICP | Reserved | LF[3:0] | Nbits[2:0] | Mbits[4:0] |

| Field | Value | Description |
|------------|-------|--|
| PLLEN | 0 | PLL powered down. |
| | 1 | PLL powered up. |
| PLLSEL | 0 | DSP internal clock taken directly from CKI. |
| | 1 | DSP internal clock taken from PLL. |
| ICP | — | Charge pump current selection (see Table 64 for proper value). |
| Reserved | 0 | — |
| LF[3:0] | — | Loop filter setting (see Table 64 for proper value). |
| Nbits[2:0] | — | Encodes N, $1 \leq N \leq 8$, where $N = \text{Nbits}[2:0] + 2$, unless $\text{Nbits}[2:0] = 111$, then $N = 1$. |
| Mbits[4:0] | — | Encodes M, $2 \leq M \leq 20$, where $M = \text{Mbits}[4:0] + 2$, $f_{\text{INTERNAL CLOCK}} = f_{\text{CKI}} \times (M/(2N))$. |

5 Software Architecture (continued)

Table 37. sbit Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|------------|----|-------------------------------------|----|----|----|---|------------|---|---|---|---|---|---|---|---|
| Field | DIREC[7:0] | | | | | | | VALUE[7:0] | | | | | | | | |
| Field | Value | | Description | | | | | | | | | | | | | |
| DIREC | 1xxxxxxx | | IOBIT7 is an output (input when 0). | | | | | | | | | | | | | |
| | x1xxxxxx | | IOBIT6 is an output (input when 0). | | | | | | | | | | | | | |
| | xx1xxxxx | | IOBIT5 is an output (input when 0). | | | | | | | | | | | | | |
| | xxx1xxxx | | IOBIT4 is an output (input when 0). | | | | | | | | | | | | | |
| | xxxx1xxx | | IOBIT3 is an output (input when 0). | | | | | | | | | | | | | |
| | xxxxx1xx | | IOBIT2 is an output (input when 0). | | | | | | | | | | | | | |
| | xxxxxx1x | | IOBIT1 is an output (input when 0). | | | | | | | | | | | | | |
| | xxxxxxx1 | | IOBIT0 is an output (input when 0). | | | | | | | | | | | | | |
| VALUE | Rxxxxxxx | | Reads the current value of IOBIT7. | | | | | | | | | | | | | |
| | xRxxxxxx | | Reads the current value of IOBIT6. | | | | | | | | | | | | | |
| | xxRxxxxx | | Reads the current value of IOBIT5. | | | | | | | | | | | | | |
| | xxxRxxxx | | Reads the current value of IOBIT4. | | | | | | | | | | | | | |
| | xxxxRxxx | | Reads the current value of IOBIT3. | | | | | | | | | | | | | |
| | xxxxxRxx | | Reads the current value of IOBIT2. | | | | | | | | | | | | | |
| | xxxxxxRx | | Reads the current value of IOBIT1. | | | | | | | | | | | | | |
| | xxxxxxxR | | Reads the current value of IOBIT0. | | | | | | | | | | | | | |

Table 38. cbit Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|----------------|----|-------------|----|----------------|----|---|---------------|---|---|---|---------------|---|---|---|---|
| Field | MODE/MASK[7:4] | | | | MODE/MASK[3:0] | | | DATA/PAT[7:4] | | | | DATA/PAT[3:0] | | | | |
| DIREC[n]* | MODE/MASK[n] | | DATA/PAT[n] | | Action | | | | | | | | | | | |
| 1 (Output) | 0 | | 0 | | Clear | | | | | | | | | | | |
| 1 (Output) | 0 | | 1 | | Set | | | | | | | | | | | |
| 1 (Output) | 1 | | 0 | | No Change | | | | | | | | | | | |
| 1 (Output) | 1 | | 1 | | Toggle | | | | | | | | | | | |
| 0 (Input) | 0 | | 0 | | No Test | | | | | | | | | | | |
| 0 (Input) | 0 | | 1 | | No Test | | | | | | | | | | | |
| 0 (Input) | 1 | | 0 | | Test for Zero | | | | | | | | | | | |
| 0 (Input) | 1 | | 1 | | Test for One | | | | | | | | | | | |

* $0 \leq n \leq 7$.

5 Software Architecture (continued)

Table 39. **alf** Register

| | | | |
|--------------|-------|-------|-------|
| Bit | 15 | 14 | 13—0 |
| Field | AWAIT | LOWPR | FLAGS |

| Field | Value | Action |
|-------|-------|---|
| AWAIT | 1 | Power-saving standby mode or standard sleep enabled. |
| | 0 | Normal operation. |
| LOWPR | 1 | The internal DPRAM is addressed beginning at 0x0000 in X space. |
| | 0 | The internal DPRAM is addressed beginning at 0xc000 in X space. |
| FLAGS | — | See table below. |

| Bit | Flag | Use |
|------|----------|------------------------|
| 13—9 | Reserved | — |
| 8 | ebusy* | ECCP BUSY |
| 7 | nmns1 | NOT-MINUS-ONE from BMU |
| 6 | mns1 | MINUS-ONE from BMU |
| 5 | evenp | EVEN PARITY from BMU |
| 4 | oddp | ODD PARITY from BMU |
| 3 | somef | SOME FALSE from BIO |
| 2 | somet | SOME TRUE from BIO |
| 1 | allf | ALL FALSE from BIO |
| 0 | allt | ALL TRUE from BIO |

* The ebusy flag cannot be written by the user.

Table 40. **mwait** Register

| | | | | |
|--------------|-----------|-------------|---------|-------------|
| Bit | 15—12 | 11—8 | 7—4 | 3—0 |
| Field | EROM[3:0] | ERAMHI[3:0] | IO[3:0] | ERAMLO[3:0] |

If the EXM pin is high and the INT1 is low upon reset, the **mwait** register is initialized to all 1s (15 wait-states for all external memory). Otherwise, the **mwait** register is initialized to all 0s (0 wait-states) upon reset.

Table 41. **DSP1628 32-Bit JTAG ID Register**

| | | | | | | |
|--------------|----------|--------|-------|---------|---------|-------|
| Bit | 31 | 30 | 29—28 | 27—19 | 18—12 | 11—0 |
| Field | RESERVED | SECURE | CLOCK | ROMCODE | PART ID | 0x03B |

| Field | Value | Mask-Programmable Features |
|----------|-------|---|
| RESERVED | 0 | — |
| SECURE | 0 | Nonsecure ROM option. |
| | 1 | Secure ROM option. |
| CLOCK | 01 | Small-signal input clock option. |
| | 11 | CMOS level input clock option. |
| ROMCODE | — | Users ROMCODE ID: The ROMCODE ID is the 9-bit binary value of the following expression: (20 x value for first letter) + (value of second letter), where the values of the letters are in the following table. For example, ROMCODE GK is (20 x 6) + (9) = 129 or 0 1000 0001. |
| PART ID | 0x2A | DSP1628 |

| | | | | | | | | | | | | | | | | | | | | |
|-----------------------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|
| ROMCODE Letter | A | B | C | D | E | F | G | H | J | K | L | M | N | P | R | S | T | U | W | Y |
| Value | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

5 Software Architecture (continued)

Table 42. *ioc* Register*

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8—7 | 6—4 | 3—0 |
|-------|----------|--------|------|-------|-------|-------|--------|----------|----------|-----------|
| Field | Reserved | EXTROM | CKO2 | EBIOH | WEROM | ESIO2 | SIOLBC | CKO[1:0] | Reserved | DENB[3:0] |

* The field definitions for the *ioc* register are different from the DSP1610.

ioc Fields

| <i>ioc</i> Field | Description |
|------------------|--|
| EXTROM | If 1, sets AB15 low during external memory accesses when WEROM = 1. |
| CKO2 | CKO configuration (see below). |
| EBIOH | If 1, enables high half of BIO, IOBIT[4:7], and disables VEC[3:0] from pins. |
| WEROM | If 1, allows writing into external program (X) memory. |
| ESIO2 | If 1, enables SIO2 and low half of BIO, and disables PHIF from pins. |
| SIOLBC | If 1, DO1 and DO2 looped back to DI1 and DI2. |
| CKO[1:0] | CKO configuration (see below). |
| DENB3 | If 1, delay EROM. |
| DENB2 | If 1, delay ERAMHI. |
| DENB1 | If 1, delay IO. |
| DENB0 | If 1, delay ERAMLO. |

| CKO2 | CKO1 | CKO0 | CKO Output | | Description |
|------|------|------|---------------|---------------------------------|--|
| — | — | — | 1X | PLL | — |
| 0 | 0 | 0 | CKI | $CKI \times M/(2N)$ | Free-running clock. ^{1, 2} |
| 0 | 0 | 1 | $CKI/(1 + W)$ | $CKI \times (M/(2N)) / [1 + W]$ | Wait-stated clock. ^{1–3} |
| 0 | 1 | 0 | 1 | 1 | Held high. |
| 0 | 1 | 1 | 0 | 0 | Held low. |
| 1 | 0 | 0 | CKI | CKI | Output of CKI buffer. |
| 1 | 0 | 1 | $CKI/(1 + W)$ | $CKI \times (M/(2N)) / [1 + W]$ | Sequenced, wait-stated clock. ^{1–4} |
| 1 | 1 | 0 | Reserved | | |
| 1 | 1 | 1 | Reserved | | |

1. The phase of CKI is synchronized by the rising edge of RSTB.
2. When SLOWCKI is enabled in the **powerc** register, these options reflect the low-speed internal ring oscillator.
3. The wait-stated clock reflects the internal instruction cycle and may be stretched based on the **mwait** register setting (see Table 40). During sequenced external memory accesses, it completes one cycle.
4. The sequenced wait-stated clock completes two cycles during a sequenced external memory access and may be stretched based on the **mwait** register setting (see Table 40).

5 Software Architecture (continued)

Table 43. powerc Register

The **powerc** register configures various power management modes.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9—8 | 7 | 6 | 5 | 4 | 3—1 | 0 |
|-------|--------|---------|------|--------|-------|--------|-------|---------|---------|---------|----------|-------|---------|
| Field | XTLOFF | SLOWCKI | NOCK | INT0EN | rsrvd | INT1EN | rsrvd | SIO1DIS | SIO2DIS | PHIFDIS | TIMERDIS | rsrvd | ECCPDIS |

Note: The reserved (rsrvd) bits should always be written with zeros to make the program compatible with future chip versions.

powerc fields

| Field | Description |
|----------|---|
| XTLOFF | 1 = power down small-signal clock input. |
| SLOWCKI | 1 = select ring oscillator clock (internal slow clock). |
| NOCK | 1 = disable internal processor clock. |
| INT0EN | 1 = INT0 clears NOCK field. |
| INT1EN | 1 = INT1 clears NOCK field. |
| SIO1DIS | 1 = disable SIO1. |
| SIO2DIS | 1 = disable SIO2. |
| PHIFDIS | 1 = disable PHIF. |
| TIMERDIS | 1 = disable timer. |
| ECCPDIS | 1 = disable ECCP. |

5 Software Architecture (continued)

Table 44. Register Settings After Reset

A • indicates that this bit is unknown on powerup reset and unaffected on subsequent reset. An S indicates that this bit shadows the PC. P indicates the value on an input pin, i.e., the bit in the register reflects the value on the corresponding input pin.

| Register | Bits 15—0 | Register | Bits 15—0 |
|----------|-------------------|----------|-------------------|
| r0 | •••••••••••••••• | inc | 0000000000000000 |
| r1 | •••••••••••••••• | ins | 0000010000000110 |
| r2 | ••••••~•••••••••• | sdx2 | ••••••~•••••••••• |
| r3 | ••••~•••••••••••• | saddx | ••••••~•••••••••• |
| j | ••••••~•••••••••• | cloop | 00000000•••••••• |
| k | ••••••~•••••••••• | mwait | 0000000000000000† |
| rb | 0000000000000000 | saddx2 | ••••••~•••••••••• |
| re | 0000000000000000 | sioc2 | ••••••0000000000 |
| pt | ••••••~•••••••••• | cbit | ••••••~•••••••••• |
| pr | ••••••~••••~••••• | sbit | 00000000PPPPPPPP |
| pi | SSSSSSSSSSSSSSSS | ioc | 0000000000000000 |
| i | ••••••~••••~••••• | jtag | ••••••~••••~••••• |
| p | ••••••~••••~••••• | eir | 000000000001111 |
| pl | ••••~••••~••••~•• | a0 | ••••~••••~••••~•• |
| pllc | 0000000000000000 | | |
| x | ••••~••••~••••~•• | a0l | ••••~••••~••••~•• |
| y | ••••~••••~••••~•• | a1 | ••••~••••~••••~•• |
| yl | ••••~••••~••••~•• | a1l | ••••~••••~••~•••• |
| auc | 0000000000000000 | timerc | ••••~••••~••••~•• |
| psw | ••••00••••~••••~• | timer0 | 0000000000000000 |
| c0 | ••••~••••~••••~•• | tdms2 | ••••~••••~••••~•• |
| c1 | ••••~••••~••••~•• | srta2 | ••••~••••~••~•••• |
| c2 | ••••~••••~••~•••• | powerc | 0000000000000000 |
| sioc | ••••~••••~••~•••• | edr | ••••~••••~••~•••• |
| srta | ••••~••••~••~•••• | ar0 | ••••~••••~••~•••• |
| sdx | ••••~••••~••~•••• | ar1 | ••••~••••~••~•••• |
| tdms | ••••~••••~••~•••• | ar2 | ••••~••••~••~•••• |
| phfc | 0000000000000000 | ar3 | ••••~••••~••~•••• |
| pdx0 | 0000000000000000 | ear | 0000000000000000 |
| ybase | ••••~••••~••~•••• | alf | 00000000••••~••• |

† If EXM is high and INT1 is low when RSTB goes high, **mwait** will contain all ones instead of all zeros.

5 Software Architecture (continued)

5.3 Instruction Set Formats

This section defines the hardware-level encoding of the DSP1628 device instructions.

Multiply/ALU Instructions

Format 1: Multiply/ALU Read/Write Group

| Field | T | | | | | D | S | F1 | | | | | X | Y | | | |
|-------|----|----|----|----|----|----|---|----|---|---|---|---|---|---|---|---|--|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |

Format 1a: Multiply/ALU Read/Write Group

| Field | T | | | | | \overline{aT} | S | F1 | | | | | X | Y | | | |
|-------|----|----|----|----|----|-----------------|---|----|---|---|---|---|---|---|---|---|--|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |

Format 2: Multiply/ALU Read/Write Group

| Field | T | | | | | D | S | F1 | | | | | X | Y | | | |
|-------|----|----|----|----|----|----|---|----|---|---|---|---|---|---|---|---|--|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |

Format 2a: Multiply/ALU Read/Write Group

| Field | T | | | | | \overline{aT} | S | F1 | | | | | X | Y | | | |
|-------|----|----|----|----|----|-----------------|---|----|---|---|---|---|---|---|---|---|--|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |

Special Function Instructions

Format 3: F2 ALU Special Functions

| Field | T | | | | | D | S | F2 | | | | | CON | | | | |
|-------|----|----|----|----|----|----|---|----|---|---|---|---|-----|---|---|---|--|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |

Format 3a: F3 ALU Operations

| Field | T | | | | | D | S | F3 | | | | | SRC2 | aT | 0 | 1 |
|-------|--------------------------|----|----|----|----|----|---|----|---|---|---|---|------|----|---|---|
| | Immediate Operand (IM16) | | | | | | | | | | | | | | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Format 3b: BMU Operations

| Field | T | | | | | D | S | F4[3—1] | | | | 0 | F4[0] | AR | | | |
|-------|--------------------------|----|----|----|----|----|---|---------|---|---|---|---|-------|----|---|---|--|
| | Immediate Operand (IM16) | | | | | | | | | | | | | | | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |

5 Software Architecture (continued)

Control Instructions

Format 4: Branch Direct Group

| Field | T | | | | JA | | | | | | | | | | | |
|-------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Format 5: Branch Indirect Group

| Field | T | | | | | B | | | reserved | | | | | | | 0 |
|-------|----|----|----|----|----|----|---|---|----------|---|---|---|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Format 6: Conditional Branch Qualifier/Software Interrupt (icall)

Note that a branch instruction immediately follows except for a software interrupt (icall).

| Field | T | | | | | SI | reserved | | | | CON | | | | | |
|-------|----|----|----|----|----|----|----------|---|---|---|-----|---|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Data Move Instructions

Format 7: Data Move Group

| Field | T | | | | | $\bar{a}T$ | R | | | | | | Y/Z | | | |
|-------|----|----|----|----|----|------------|---|---|---|---|---|---|-----|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Format 8: Data Move (immediate operand — 2 words)

| Field | T | | | | | D | R | | | | | | reserved | | | |
|-------|--------------------------|----|----|----|----|----|---|---|---|---|---|---|----------|---|---|---|
| | Immediate Operand (IM16) | | | | | | | | | | | | | | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Format 9: Short Immediate Group

| Field | T | | | | | I | Short Immediate Operand (IM9) | | | | | | | | | |
|-------|----|----|----|----|----|----|-------------------------------|---|---|---|---|---|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Format 9a: Direct Addressing

| Field | T | | | | | R/W | DR | | | | 1 | OFFSET | | | | |
|-------|----|----|----|----|----|-----|----|---|---|---|---|--------|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Cache Instructions

Format 10: Do/Redo

| Field | T | | | | | N | | | | | K | | | | | |
|-------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

5 Software Architecture (continued)

Field Descriptions

Table 45. T Field

Specifies the type of instruction.

| T | Operation | Format |
|-------|------------------------------|--------|
| 0000x | goto JA | 4 |
| 00010 | Short imm j, k, rb, re | 9 |
| 00011 | Short imm r0, r1, r2, r3 | 9 |
| 00100 | Y = a1[<i>i</i>] | F1 1 |
| 00101 | Z : aT[<i>i</i>] | F1 2a |
| 00110 | Y | F1 1 |
| 00111 | aT[<i>i</i>] = Y | F1 1a |
| 01000 | Bit 0 = 0, aT = R | 7 |
| 01000 | Bit 0 = 1, aTl = R | 7 |
| 01001 | Bit 10 = 0, R = a0 | 7 |
| 01001 | Bit 10 = 1, R = a0l | 7 |
| 01010 | R = IM16 | 8 |
| 01011 | Bit 10 = 0, R = a1 | 7 |
| 01011 | Bit 10 = 1, R = a1l | 7 |
| 01100 | Y = R | 7 |
| 01101 | Z : R | 7 |
| 01110 | do, redo | 10 |
| 01111 | R = Y | 7 |
| 1000x | call JA | 4 |
| 10010 | ifc CON | F2 3 |
| 10011 | if CON | F2 3 |
| 10100 | Y = y[<i>i</i>] | F1 1 |
| 10101 | Z : y[<i>i</i>] | F1 2 |
| 10110 | x = Y | F1 1 |
| 10111 | y[<i>i</i>] = Y | F1 1 |
| 11000 | Bit 0 = 0, branch indirect | 5 |
| 11000 | Bit 0 = 1, F3 ALU | 3a |
| 11001 | y = a0 x = X | F1 1 |
| 11010 | Cond. branch qualifier | 6 |
| 11011 | y = a1 x = X | F1 1 |
| 11100 | Y = a0[<i>i</i>] | F1 1 |
| 11101 | Z : y x = X | F1 2 |
| 11110 | Bit 5 = 0, F4 ALU (BMU) | 3b |
| 11110 | Bit 5 = 1, direct addressing | 9a |
| 11111 | y = Y x = X | F1 1 |

Table 46. D Field

Specifies a destination accumulator.

| D | Register |
|---|---------------|
| 0 | Accumulator 0 |
| 1 | Accumulator 1 |

Table 47. aT Field

Specifies transfer accumulator.

| aT | Register |
|----|---------------|
| 0 | Accumulator 1 |
| 1 | Accumulator 0 |

Table 48. S Field

Specifies a source accumulator.

| S | Register |
|---|---------------|
| 0 | Accumulator 0 |
| 1 | Accumulator 1 |

Table 49. F1 Field

Specifies the multiply/ALU function.

| F1 | Operation |
|------|----------------------------|
| 0000 | aD = p p = x * y |
| 0001 | aD = aS + p p = x * y |
| 0010 | p = x * y |
| 0011 | aD = aS - p p = x * y |
| 0100 | aD = p |
| 0101 | aD = aS + p |
| 0110 | nop |
| 0111 | aD = aS - p |
| 1000 | aD = aS y |
| 1001 | aD = aS ^ y |
| 1010 | aS & y |
| 1011 | aS - y |
| 1100 | aD = y |
| 1101 | aD = aS + y |
| 1110 | aD = aS & y |
| 1111 | aD = aS - y |

Table 50. X Field

Specifies the addressing of ROM data in two-operand multiply/ALU instructions. Specifies the high or low half of an accumulator or the y register in one-operand multiply/ALU instructions.

| X | Operation |
|---------------------------------|-----------|
| Two-Operand Multiply/ALU | |
| 0 | *pt++ |
| 1 | *pt++i |
| One-Operand Multiply/ALU | |
| 0 | aTl, yl |
| 1 | aTh, yh |

5 Software Architecture (continued)

Table 51. Y Field

Specifies the form of register indirect addressing with postmodification.

| Y | Operation |
|------|-----------|
| 0000 | *r0 |
| 0001 | *r0++ |
| 0010 | *r0-- |
| 0011 | *r0++j |
| 0100 | *r1 |
| 0101 | *r1++ |
| 0110 | *r1-- |
| 0111 | *r1++j |
| 1000 | *r2 |
| 1001 | *r2++ |
| 1010 | *r2-- |
| 1011 | *r2++j |
| 1100 | *r3 |
| 1101 | *r3++ |
| 1110 | *r3-- |
| 1111 | *r3++j |

Table 52. Z Field

Specifies the form of register indirect compound addressing with postmodification.

| Z | Operation |
|------|-----------|
| 0000 | *r0zp |
| 0001 | *r0pz |
| 0010 | *r0m2 |
| 0011 | *r0jk |
| 0100 | *r1zp |
| 0101 | *r1pz |
| 0110 | *r1m2 |
| 0111 | *r1jk |
| 1000 | *r2zp |
| 1001 | *r2pz |
| 1010 | *r2m2 |
| 1011 | *r2jk |
| 1100 | *r3zp |
| 1101 | *r3pz |
| 1110 | *r3m2 |
| 1111 | *r3jk |

Table 53. F2 Field

Specifies the special function to be performed.

| F2 | Operation |
|------|---------------|
| 0000 | aD = aS >> 1 |
| 0001 | aD = aS << 1 |
| 0010 | aD = aS >> 4 |
| 0011 | aD = aS << 4 |
| 0100 | aD = aS >> 8 |
| 0101 | aD = aS << 8 |
| 0110 | aD = aS >> 16 |
| 0111 | aD = aS << 16 |
| 1000 | aD = p |
| 1001 | aDh = aSh + 1 |
| 1010 | aD = ~aS |
| 1011 | aD = rnd(aS) |
| 1100 | aD = y |
| 1101 | aD = aS + 1 |
| 1110 | aD = aS |
| 1111 | aD = -aS |

Table 54. CON Field

Specifies the condition for special functions and conditional control instructions.

| CON | Condition | CON | Condition |
|-------------|-----------|-------|-----------|
| 00000 | mi | 01110 | true |
| 00001 | pl | 01111 | false |
| 00010 | eq | 10000 | gt |
| 00011 | ne | 10001 | le |
| 00100 | lvs | 10010 | allt |
| 00101 | lvc | 10011 | allf |
| 00110 | mvs | 10100 | somet |
| 00111 | mvc | 10101 | somef |
| 01000 | heads | 10110 | oddp |
| 01001 | tails | 10111 | evenp |
| 01010 | c0ge | 11000 | mns1 |
| 01011 | c0lt | 11001 | nmns1 |
| 01100 | c1ge | 11010 | npint |
| 01101 | c1lt | 11011 | njint |
| | | 11100 | lock |
| | | 11101 | ebusy |
| Other codes | Reserved | — | — |

5 Software Architecture (continued)

Table 55. R Field

Specifies the register for data move instructions.

| R | Register | R | Register |
|--------|----------|--------|----------|
| 000000 | r0 | 100000 | inc |
| 000001 | r1 | 100001 | ins |
| 000010 | r2 | 100010 | sdx2 |
| 000011 | r3 | 100011 | saddx |
| 000100 | j | 100100 | cloop |
| 000101 | k | 100101 | mwait |
| 000110 | rb | 100110 | saddx2 |
| 000111 | re | 100111 | sioc2 |
| 001000 | pt | 101000 | cbit |
| 001001 | pr | 101001 | sbit |
| 001010 | pi | 101010 | ioc |
| 001011 | i | 101011 | jtag |
| 001100 | p | 101100 | Reserved |
| 001101 | pl | 101101 | Reserved |
| 001110 | pllc | 101110 | Reserved |
| 001111 | Reserved | 101111 | eir |
| 010000 | x | 110000 | a0 |
| 010001 | y | 110001 | a0l |
| 010010 | yl | 110010 | a1 |
| 010011 | auc | 110011 | a1l |
| 010100 | psw | 110100 | timerc |
| 010101 | c0 | 110101 | timer0 |
| 010110 | c1 | 110110 | tdms2 |
| 010111 | c2 | 110111 | srta2 |
| 011000 | sioc | 111000 | powerc |
| 011001 | srta | 111001 | edr |
| 011010 | sdx | 111010 | ar0 |
| 011011 | tdms | 111011 | ar1 |
| 011100 | phifc | 111100 | ar2 |
| 011101 | pdx0 | 111101 | ar3 |
| 011110 | Reserved | 111110 | ear |
| 011111 | ybase | 111111 | alf |

Table 56. B Field

Specifies the type of branch instruction (except software interrupt).

| B | Operation |
|-----|-----------|
| 000 | return |
| 001 | ireturn |
| 010 | goto pt |
| 011 | call pt |
| 1xx | Reserved |

Table 57. DR Field

| DR Value | Register |
|----------|----------|
| 0000 | r0 |
| 0001 | r1 |
| 0010 | r2 |
| 0011 | r3 |
| 0100 | a0 |
| 0101 | a0l |
| 0110 | a1 |
| 0111 | a1l |
| 1000 | y |
| 1001 | yl |
| 1010 | p |
| 1011 | pl |
| 1100 | x |
| 1101 | pt |
| 1110 | pr |
| 1111 | psw |

Table 58. I Field

Specifies a register for short immediate data move instructions.

| I | Register |
|----|----------|
| 00 | r0/j |
| 01 | r1/k |
| 10 | r2/rb |
| 11 | r3/re |

Table 59. SI Field

Specifies when the conditional branch qualifier instruction should be interpreted as a software interrupt instruction.

| SI | Operation |
|----|--------------------------|
| 0 | Not a software interrupt |
| 1 | Software interrupt |

5 Software Architecture (continued)

N Field

Number of instructions to be loaded into the cache. Zero implies redo operation.

K Field

Number of times the N instructions in cache are to be executed. Zero specifies use of value in **cloop** register.

JA Field

12-bit jump address.

R/W Field

A zero specifies a write, $*(OFFSET) = DR$.

A one specifies a read, $DR = *(OFFSET)$.

Table 60. F3 Field

Specifies the operation in an F3 ALU instruction.

| F3 | Operation | | |
|------|-----------------|---|---------------|
| 1000 | $aD = aS[h, l]$ | | {aT, IM16, p} |
| 1001 | $aD = aS[h, l]$ | ^ | {aT, IM16, p} |
| 1010 | $aS[h, l]$ | & | {aT, IM16, p} |
| 1011 | $aS[h, l]$ | - | {aT, IM16, p} |
| 1101 | $aD = aS[h, l]$ | + | {aT, IM16, p} |
| 1110 | $aD = aS[h, l]$ | & | {aT, IM16, p} |
| 1111 | $aD = aS[h, l]$ | - | {aT, IM16, p} |

Table 61. SRC2 Field

Specifies operands in an F3 ALU instruction.

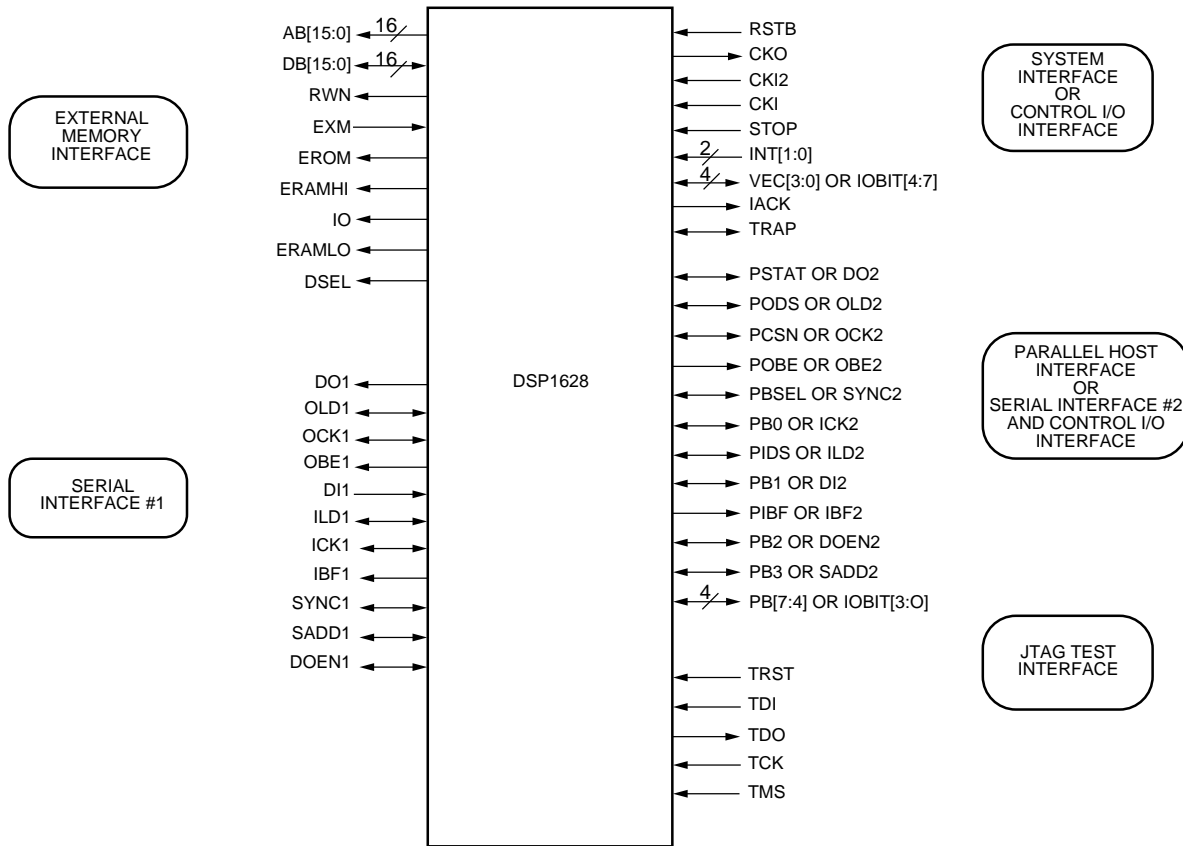
| SRC2 | Operands |
|------|-----------|
| 00 | aSI, IM16 |
| 10 | aSh, IM16 |
| 01 | aS, aT |
| 11 | aS, p |

Table 62. BMU Encodings

| F4 | AR | Operation |
|------|------|----------------------------------|
| 0000 | 00xx | $aD = aS \gg arM$ |
| 0001 | 00xx | $aD = aS \ll arM$ |
| 0000 | 10xx | $aD = aS \ggg arM$ |
| 0001 | 10xx | $aD = aS \lll arM$ |
| 1000 | 0000 | $aD = \overline{aS} \gg aS$ |
| 1001 | 0000 | $aD = \overline{aS} \ll aS$ |
| 1000 | 1000 | $aD = \overline{aS} \ggg aS$ |
| 1001 | 1000 | $aD = \overline{aS} \lll aS$ |
| 1100 | 0000 | $aD = aS \gg IM16$ |
| 1101 | 0000 | $aD = aS \ll IM16$ |
| 1100 | 1000 | $aD = aS \ggg IM16$ |
| 1101 | 1000 | $aD = aS \lll IM16$ |
| 0000 | 1100 | $aD = \text{exp}(aS)$ |
| 0001 | 11xx | $aD = \text{norm}(aS, arM)$ |
| 1110 | 0000 | $aD = \text{extracts}(aS, IM16)$ |
| 0010 | 00xx | $aD = \text{extracts}(aS, arM)$ |
| 1110 | 0100 | $aD = \text{extractz}(aS, IM16)$ |
| 0010 | 01xx | $aD = \text{extractz}(aS, arM)$ |
| 1110 | 1000 | $aD = \text{insert}(aS, IM16)$ |
| 1010 | 10xx | $aD = \text{insert}(aS, arM)$ |
| 0111 | 0000 | $aD = aS:aa0$ |
| 0111 | 0001 | $aD = aS:aa1$ |

Note: xx encodes the auxiliary register to be used; 00 (ar0), 01(ar1), 10 (ar2), or 11(ar3).

6 Signal Descriptions



5-4006 (C).h

Figure 12. DSP1628 Pinout by Interface

Figure 12 shows the pinout for the DSP1628. The signals can be separated into five interfaces as shown. These interfaces and the signals that comprise them are described below.

6.1 System Interface

The system interface consists of the clock, interrupt, and reset signals for the processor.

RSTB

Reset: Negative assertion. A high-to-low transition causes the processor to enter the reset state. The **auc**, **powerc**, **sioc**, **sioc2**, **phifc**, **pdx0**, **tdms**, **tdms2**, **timerc**, **timer0**, **sbit** (upper byte), **inc**, **ins** (except OBE, OBE2, and PODS status bits set), **alf** (upper 2 bits, AWAIT and LOWPR), **ioc**, **rb**, and **re** registers are cleared. The **mwait** register is initialized to all 0s (zero wait-states) unless the EXM pin is high and the INT1 pin is low. In that case, the **mwait** register is initialized to all 1s (15 wait-states).

Reset clears IACK, VEC[3:0]/IOBIT[4:7], IBF, and IBF2. The DAU condition flags are not affected by reset. IOBIT[7:0] are initialized as inputs. If any of the IOBIT pins are switched to outputs (by writing **sbit**), their initial value will be logic zero (see Figure 44, Register Settings After Reset).

Upon negation of the signal, the processor begins execution at location 0x0000 in the active memory map (see Section 4.4, Memory Maps and Wait-States).

CKI

Input Clock: A mask-programmable option selects one of three possible input buffers for the CKI pin (see Section 7, Mask-Programmable Options, and Table 1, Pin Descriptions). The internal CKI from the output of the selected input buffer can then drive the internal processor clock directly (1X) or drive the on-chip PLL (see Section 4.13). The PLL allows the CKI input clock to be at a lower frequency than the internal processor clock.

6 Signal Descriptions (continued)

CKI2

Input Clock 2: Used with mask-programmable input clock options which require an external small signal differential across CKI and CKI2 (see Table 1, Pin Descriptions). When the CMOS option is selected, this pin should be tied to VSSA.

STOP

Stop Input Clock: Negative assertion. A high-to-low transition synchronously stops all of the internal processor clocks leaving the processor in a defined state. Returning the pin high will synchronously restart the processor clocks to continue program execution from where it left off without any loss of state. This hardware feature has the same effect as setting the NOCK bit in the **powerc** register (see Table 43).

CKO

Clock Out: Buffered output clock with options programmable via the **ioc** register (see Table 42). The selectable CKO options (see Tables 42 and 33) are as follows:

- A free-running output clock at the frequency of the internal processor clock; runs at the internal ring oscillator frequency when SLOWCKI is enabled.
- A wait-stated clock based on the internal instruction cycle; runs at the internal ring oscillator frequency when SLOWCKI is enabled.
- A sequenced, wait-stated clock based on the EMI sequencer cycle; runs at the internal ring oscillator frequency when SLOWCKI is enabled.
- A free-running output clock that runs at the CKI rate, independent of the **powerc** register setting. This option is only available with the small-signal clock options. When the PLL is selected, the CKO frequency equals the input CKI frequency regardless of how the PLL is programmed.
- A logic 0.
- A logic 1.

INT[1:0]

Processor Interrupts 0 and 1: Positive assertion. Hardware interrupt inputs to the DSP1628. Each is enabled via the **inc** register. When enabled and asserted, each cause the processor to vector to the memory location described in Table 4. INT1 is used in conjunction with EXM to select the desired reset initialization of the **mwait** register (see Table 40). **When both INT0 and RSTB are asserted, all output and bidirectional pins (except TDO, which 3-states by JTAG control) are put in a 3-state condition.**

VEC[3:0]

Interrupt Output Vector: These four pins indicate which interrupt is currently being serviced by the device. Table 4 shows the code associated with each interrupt condition. VEC[3:0] are multiplexed with IOBIT[4:7].

IACK

Interrupt Acknowledge: Positive assertion. IACK signals when an interrupt is being serviced by the DSP1628. IACK remains asserted while in an interrupt service routine, and is cleared when the ireturn instruction is executed.

TRAP

Trap Signal: Positive assertion. When asserted, the processor is put into the trap condition, which normally causes a branch to the location 0x0046. The hardware development system (HDS) can configure the trap pin to cause an HDS trap, which causes a branch to location 0x0003. Although normally an input, the pin can be configured as an output by the HDS. As an output, the pin can be used to signal an HDS breakpoint in a multiple processor environment.

6 Signal Descriptions (continued)

6.2 External Memory Interface

The external memory interface is used to interface the DSP1628 to external memory and I/O devices. It supports read/write operations from/to program and data memory spaces. The interface supports four external memory segments. Each external memory segment can have an independent number of software-programmable wait-states. One hardware address is decoded, and an enable line is provided, to allow glueless I/O interfacing.

AB[15:0]

External Memory Address Bus: Output only. This 16-bit bus supplies the address for read or write operations to the external memory or I/O. During external memory accesses, AB[15:0] retain the value of the last valid external access.

DB[15:0]

External Memory Data Bus: This 16-bit bidirectional data bus is used for read or write operations to the external memory or I/O.

RWN

Read/Write Not: When a logic 1, the pin indicates that the memory access is a read operation. When a logic 0, the memory access is a write operation.

EXM

External Memory Select: Input only. This signal is latched into the device on the rising edge of RSTB. The value of EXM latched in determines whether the internal ROM is addressable in the instruction/coefficient memory map. If EXM is low, internal ROM is addressable. If EXM is high, only external ROM is addressable in the instruction/coefficient memory map (see Table 5, Instruction/Coefficient Memory Maps). EXM chooses between MAP1 or MAP2 and between MAP3 or MAP4.

EROM

External ROM Enable Signal: Negative assertion. When asserted, the signal indicates an access to external program memory (see Table 5, Instruction/Coefficient Memory Maps). This signal's leading edge can be delayed via the **ioc** register (see Table 42).

ERAMHI

External RAM High Enable Signal: Negative assertion. When asserted, the signal indicates an access to external data memory addresses 0x8000 through 0xFFFF (see Table 6, Data Memory Map). This signal's leading edge can be delayed via the **ioc** register (see Table 42).

ERAMLO

External RAM Low Enable Signal: Negative assertion. When asserted, the signal indicates an access to external data memory addresses 0x4100 through 0x7FFF (see Table 6, Data Memory Map). This signal's leading edge can be delayed via the **ioc** register (see Table 42).

IO

External I/O Enable Signal: Negative assertion. When asserted, the signal indicates an access to external data memory addresses 0x4000 through 0x40FF (see Table 6, Data Memory Map). This memory segment is intended for memory-mapped I/O. This signal's leading edge can be delayed via the **ioc** register (see Table 42).

6 Signal Descriptions (continued)

6.3 Serial Interface #1

The serial interface pins implement a full-featured synchronous/asynchronous serial I/O channel. In addition, several pins offer a glueless TDM interface for multiprocessor communication applications (see Figure 6, Multiprocessor Communications and Connections).

DI1

Data Input: Serial data is latched on the rising edge of ICK1, either LSB or MSB first, according to the **sioc** register MSB field (see Table 26).

ICK1

Input Clock: The clock for serial input data. In active mode, ICK1 is an output; in passive mode, ICK1 is an input, according to the **sioc** register ICK field (see Table 26). Input has typically 0.7 V hysteresis.

ILD1

Input Load: The clock for loading the input buffer, **sd_x[in]**, from the input shift register **isr**. A falling edge of ILD1 indicates the beginning of a serial input word. In active mode, ILD1 is an output; in passive mode, ILD1 is an input, according to the **sioc** register ILD field (see Table 26). Input has typically 0.7 V hysteresis.

IBF1

Input Buffer Full: Positive assertion. IBF1 is asserted when the input buffer, **sd_x[in]**, is filled. IBF1 is negated by a read of the buffer, as in **a0 = sd_x**. IBF1 is also negated by asserting RSTB.

DO1

Data Output: The serial data output from the output shift register (**osr**), either LSB or MSB first (according to the **sioc** register MSB field). DO1 changes on the rising edges of OCK1. DO1 is 3-stated when DOEN1 is high.

DOEN1

Data Output Enable: Negative assertion. An input when not in the multiprocessor mode. DO1 and SADD1 are enabled only if DOEN1 is low. DOEN1 is bidirectional when in the multiprocessor mode (**tdms** register MODE field set). In the multiprocessor mode, DOEN1 indicates a valid time slot for a serial output.

OCK1

Output Clock: The clock for serial output data. In active mode, OCK1 is an output; in passive mode, OCK1 is an input, according to the **sioc** register OCK field (see Table 26). Input has typically 0.7 V hysteresis.

OLD1

Output Load: The clock for loading the output shift register, **osr**, from the output buffer **sd_x[out]**. A falling edge of OLD1 indicates the beginning of a serial output word. In active mode, OLD1 is an output; in passive, OLD1 is an input, according to the **sioc** register OLD field (see Table 26). Input has typically 0.7 V hysteresis.

OBE1

Output Buffer Empty: Positive assertion. OBE1 is asserted when the output buffer, **sd_x[out]**, is emptied (moved to the output shift register for transmission). It is cleared with a write to the buffer, as in **sd_x = a0**. OBE1 is also set by asserting RSTB.

SADD1

Serial Address: Negative assertion. A 16-bit serial bit stream typically used for addressing during multiprocessor communication between multiple DSP16xx devices. In multiprocessor mode, SADD1 is an output when the **tdms** time slot dictates a serial transmission; otherwise, it is an input. Both the source and destination DSP can be identified in the transmission. SADD1 is always an output when not in multiprocessor mode and can be used as a second 16-bit serial output. See the *DSP1611/17/18/27 Digital Signal Processor Information Manual* for additional information. SADD1 is 3-stated when DOEN1 is high. When used on a bus, SADD1 should be pulled high through a 5 k Ω resistor.

SYNC1

Multiprocessor Synchronization: Typically used in the multiprocessor mode, a falling edge of SYNC1 indicates the first word (time slot 0) of a TDM I/O stream and causes the resynchronization of the active ILD1 and OLD1 generators. SYNC1 is an output when the **tdms** register SYNC field is set (i.e., selects the master DSP and uses time slot 0 for transmit). As an input, SYNC1 must be tied low unless part of a TDM interface. When used as an output, SYNC1 = [ILD1/OLD1]/8 or 16, depending on the setting of the SYNCSP field of the **tdms** register. When configured as described above, SYNC1 can be used to generate a slow clock for SIO operations. Input has typically 0.7 V hysteresis.

6 Signal Descriptions (continued)

6.4 Parallel Host Interface or Serial Interface #2 and Control I/O Interface

This interface pin multiplexes a parallel host interface with a second serial I/O interface and a 4-bit I/O interface. The interface selection is made by writing the ESIO2 bit in the **ioc** register (see Table 42 and Section 4.1). The functions and signals for the second SIO correspond exactly with those in SIO #1. Therefore, the pin descriptions below discuss only PHIF and BIO pin functionality.

PB[7:0]

Parallel I/O Data Bus: This 8-bit bidirectional bus is used to input data to, or output data from, the PHIF.

Note that PB[3:0] are pin multiplexed with SIO2 functionality, and PB[7:4] are pin multiplexed with BIO unit pins IOBIT[3:0] (see Section 4.1).

PCSN

Peripheral Chip Select Not: Negative assertion. PCSN is an input. While PCSN is low, the data strobes PIDS and PODS are enabled. While PCSN is high, the DSP1628 ignores any activity on PIDS and PODS.

PBSEL

Peripheral Byte Select: An input pin, configurable in software. Selects the high or low byte of **pdx0** available for host accesses.

PSTAT

Peripheral Status Select: PSTAT is an input. When a logic 0, the PHIF will output the **pdx0[out]** register on the PB bus. When a logic 1, the PHIF will output the contents of the PSTAT register on PB[7:0].

PIDS

Parallel Input Data Strobe: An input pin, software configurable to support both *Intel* and *Motorola* protocols.

In *Intel* mode: Negative assertion. PIDS is pulled low by an external device to indicate that data is available on the PB bus. The DSP latches data on the PB bus on the rising edge (low-to-high transition) of PIDS or PCSN, whichever comes first.

In *Motorola* mode: PIDS/PRWN functions as a read/write strobe. The external device sets PIDS/PRWN to a logic 0 to indicate that data is available on the PB bus (write operation by the external device). A logic 1 on PIDS/PRWN indicates an external read operation by the external device.

PODS

Parallel Output Data Strobe: An input pin, software configurable to support both *Intel* and *Motorola* protocols.

In *Intel* mode: Negative assertion. When PODS is pulled low by an external device, the DSP1628 places the contents of the parallel output register, **pdx0**, onto the PB bus.

In *Motorola* mode: Software-configurable assertion level. The external device uses PODS/PDS as its data strobe for both read and write operations.

PIBF

Parallel Input Buffer Full: An output pin with positive assertion; configurable in software. This flag is cleared after reset, indicating an empty input buffer **pdx0[in]**.

PIBF is set immediately after the rising edge of PIDS or PCSN, indicating that data has been latched into the **pdx0[in]** register. When the DSP1628 reads the contents of this register, emptying the buffer, the flag is cleared.

Configured in software, PIBF may become the logical OR of the PIBF and POBE flags.

POBE

Parallel Output Buffer Empty: An output pin with positive assertion; configurable in software. This flag is set after reset, indicating an empty output buffer **pdx0[out]**.

POBE is set immediately after the rising edge of PODS or PCSN, indicating that the data in **pdx0[out]** has been driven onto the PB bus. When the DSP1628 writes to **pdx0[out]**, filling the buffer, this flag is cleared.

6.5 Control I/O Interface

This interface is used for status and control operations provided by the bit I/O unit of the DSP1628. It is pin multiplexed with the PHIF and VEC[3:0] pins (see Section 4.1). Setting the ESIO2 and EBIOH bits in the **ioc** register provides a full 8-bit BIO interface at the associated pins.

IOBIT[7:0]

I/O Bits [7:0]: Each of these bits can be independently configured as either an input or an output. As outputs, they can be independently set, toggled, or cleared. As inputs, they can be tested independently or in combinations for various data patterns.

6 Signal Descriptions (continued)

6.6 JTAG Test Interface

The JTAG test interface has features that allow programs and data to be downloaded into the DSP via four pins. This provides extensive test and diagnostic capability. In addition, internal circuitry allows the device to be controlled through the JTAG port to provide on-chip in-circuit emulation. Lucent Technologies provides hardware and software tools to interface to the on-chip HDS via the JTAG port.

Note: The DSP1628 provides all JTAG/IEEE 1149.1 standard test capabilities including boundary scan. See the *DSP1611/17/18/27 Digital Signal Processor Information Manual* for additional information on the JTAG test interface.

TDI

Test Data Input: JTAG serial input signal. All serial-scanned data and instructions are input on this pin. This pin has an internal pull-up resistor.

TDO

Test Data Output: JTAG serial output signal. Serial-scanned data and status bits are output on this pin.

TMS

Test Mode Select: JTAG mode control signal that, when combined with TCK, controls the scan operations. This pin has an internal pull-up resistor.

TCK

Test Clock: JTAG serial shift clock. This signal clocks all data into the port through TDI, and out of the port through TDO, and controls the port by latching the TMS signal inside the state-machine controller.

TRST

Test Reset: Negative assertion. JTAG test reset. When asserted low, asynchronously resets JTAG TAP controller. In an application environment, this pin must be asserted prior to or concurrent with RSTB. This pin has an internal pull-up resistor.

7 Mask-Programmable Options

The DSP1628 contains a ROM that is mask-programmable. The selection of several programmable features is made when a custom ROM is encoded. These features select the input clock options, the instruction/coefficient memory map option, and the hardware emulation or ROM security option, as summarized in Table 63.

Table 63. DSP1628 ROM Options

| Features | Options | Comments |
|--------------|----------------------------|--|
| Input Clock | CMOS Level Small Signal | 2.7 V 2.7 V |
| Memory Map | DSP1628x16 DSP1628x08 | 16 Kwords DPRAM 8 Kwords DPRAM |
| ROM Security | Nonsecure | Specify and link 1628hds.v# *, allows emulation. |
| | Secure | Specify and link crc16.v# †, no emulation capability. |

* 1628hds.v# (# indicates the current version number) is the relocatable HDS object code. It uses approximately 140 words and must reside in the first 4 Kwords of ROM.

† crc16.v# is the cyclic redundancy check object code. It uses approximately 75 words and must reside in the first 4 Kwords of ROM. See the *DSP1600 Support Tools Manual* for detailed information.

7.1 Input Clock Options

For all input options, the input clock CKI can run at some fraction of the internal clock frequency by setting the PLL multiplication factors appropriately (see Section 4.13, Clock Synthesis). When the PLL is bypassed, the input clock CKI frequency is the internal clock frequency.

7.2 Memory Map Options

The DSP1628 offers a DSP1628x16 or a DSP1628x08 where the difference is in the memory maps. The DSP1628x16 contains 16 Kwords of internal RAM (DPRAM). The DSP1628x08 supports the use of only 8 Kwords of DPRAM. See Section 4.4 Memory Maps and Wait-States for further description.

7.3 ROM Security Options

The DSP1600 hardware development system (HDS) provides on-chip in-circuit emulation and requires that the relocatable HDS code be linked to the application code. This code's object file is called 1628hds.v#, where # is a unique version identifier. Refer to the DSP1628-ST software tools release for more specific information. If on-chip in-circuit emulation is desired, a nonsecure ROM must be chosen. If ROM security is desired with the DSP1628, the HDS cannot be used. To provide testing of the internal ROM contents on a secure ROM device, a cyclic redundancy check (CRC) program is called by and linked with the user's source code. The CRC code resides in the first 4 Kwords of ROM.

See the *DSP1600 Support Tools Manual* for more detailed information.

8 Device Characteristics

8.1 Absolute Maximum Ratings

Stresses in excess of the absolute maximum ratings can cause permanent damage to the device. These are absolute stress ratings only. Functional operation of the device is not implied at these or any other conditions in excess of those given in the operational sections of the data sheet. Exposure to absolute maximum ratings for extended periods can adversely affect device reliability.

External leads can be bonded and soldered safely at temperatures of up to 300 °C.....(TBD for 144-pin PBGA)

Voltage Range on VDD with Respect to Ground Using Devices Designed for 3 V Operation–0.5 V to +4.6 V

Voltage Range on Any Pin VSS – 0.5 V to VDD + 0.5 V

Power Dissipation..... 1 W

Ambient Temperature Range –40 °C to +85 °C

Storage Temperature Range.....–65 °C to +150 °C

8.2 Handling Precautions

All MOS devices must be handled with certain precautions to avoid damage due to the accumulation of static charge. Although input protection circuitry has been incorporated into the devices to minimize the effect of this static buildup, proper precautions should be taken to avoid exposure to electrostatic discharge during handling and mounting. Lucent Technologies employs a human-body model for ESD susceptibility testing. Since the failure voltage of electronic devices is dependent on the current, voltage, and hence, the resistance and capacitance, it is important that standard values be employed to establish a reference by which to compare test data. Values of 100 pF and 1500 Ω are the most common and are the values used in the Lucent Technologies human-body model test circuit. The breakdown voltage for the DSP1628 is greater than 2000 V.

8.3 Recommended Operating Conditions

Table 64. Recommended Operating Conditions

| Maximum Instruction Rate (MIPS) | Device Speed | Input Clock | Package | Supply Voltage VDD (V) | | Ambient Temperature TA (°C) | |
|---------------------------------|--------------|--------------------|-------------------------|------------------------|-----|-----------------------------|-----|
| | | | | Min | Max | Min | Max |
| 52 | 19.2 ns | CMOS, small-signal | PBGA BQFP or TQFP | 2.7 | 3.3 | –40 | 85 |
| 80 | 12.5 ns | CMOS, small-signal | PBGA BQFP or TQFP | 2.7 | 3.3 | –40 | 85 |

The ratio of the instruction cycle rate to the input clock frequency is 1:1 without the PLL (referred to as 1X operation) and M/(2N) with the PLL selected (see Section 4.13). Device speeds greater than 50 MIPS do not support 1X operation; use the PLL.

8 Device Characteristics (continued)

8.4 Package Thermal Considerations

The recommended operating temperature specified above is based on the maximum power, package type, and maximum junction temperature. The following equations describe the relationship between these parameters. If the applications' maximum power is less than the worst-case value, this relationship determines a higher maximum ambient temperature or the maximum temperature measured at top dead center of the package.

$$T_A = T_J - P \times \Theta_{JA}$$

$$T_{TDC} = T_J - P \times \Theta_{J-TDC}$$

where T_A is the still-air ambient temperature and T_{TDC} is the temperature measured by a thermocouple at the top dead center of the package.

| | |
|---|-------------------------|
| Maximum Junction Temperature (T_J) in 100-Pin BQFP | 100 °C |
| 100-pin BQFP Maximum Thermal Resistance in Still-Air-Ambient (Θ_{JA}) | 55 °C/W |
| 100-pin BQFP Maximum Thermal Resistance, Junction to Top Dead Center (Θ_{J-TDC}) | 12 °C/W |
| Maximum Junction Temperature (T_J) in 100-Pin TQFP | 100 °C |
| 100-pin TQFP Maximum Thermal Resistance in Still-Air-Ambient (Θ_{JA}) | 64 °C/W |
| 100-pin TQFP Maximum Thermal Resistance, Junction to Top Dead Center (Θ_{J-TDC}) | 6 °C/W |
| Maximum Junction Temperature (T_J) in 144-Pin PBGA | 100 °C |
| 144-pin PBGA Maximum Thermal Resistance in Still-Air-Ambient (Θ_{JA}) | TBD (estimated 30 °C/W) |
| 144-pin PBGA Maximum Thermal Resistance, Junction to Top Dead Center (Θ_{J-TDC}) | TBD |

WARNING: Due to package thermal constraints, proper precautions in the user's application should be taken to avoid exceeding the maximum junction temperature of 100 °C. Otherwise, the device will be affected adversely.

The applications' maximum power, the package type, and the maximum ambient temperature determine the maximum activity factors for the error correction coprocessor as well as for the DSP core and its peripherals. The following equations describe the relationship between these parameters. If the applications' maximum power is less than the worst-case value, this relationship permits higher activity factors. For these calculations, refer to Section 4.13, Power Management and Section 9.1, Power Dissipation.

$$P = \text{MIPS} \times [\text{AF}_{\text{ECCP}}(\text{PECCP}/\text{MIPS}) + \text{AF}_{\text{DSP}}(\text{PDSP}/\text{MIPS}) + (1 - \text{AF}_{\text{DSP}})(\text{PSLEEP}/\text{MIPS})]$$

$$P \times \Theta_{JA} + 85 \text{ °C} \leq 125 \text{ °C}$$

where:

| | |
|---------------------|--|
| P | = Maximum power in mW |
| MIPS | = Device speed (internal clock speed ÷ 10 ⁶) |
| AF _{ECCP} | = Activity factor for error correction coprocessor (ECCP) |
| AF _{DSP} | = Activity factor for DSP core and peripherals |
| AF _{SLEEP} | = Activity factor for sleep mode operation = 1 – AF _{DSP} |
| PECCP | = Power dissipation in mW for ECCP |
| PDSP | = Power dissipation in mW for DSP core and peripherals |
| PSLEEP | = Power dissipation in mW for sleep mode operation |

For example, for a TQFP device operating at 50 MIPS in a 3 V application with 40% ECCP activity, 100% DSP activity, and 0% sleep activity, the equation would look like this:

$$50 \text{ MIPS} \times [0.4 (35 \text{ mW}/50 \text{ MIPS}) + 1.0(125 \text{ mW}/50 \text{ MIPS}) + 0] = 139 \text{ mW}$$

$$139 \text{ mW} \times 64 \text{ °C}/\text{W} + 85 \text{ °C} = 94 \text{ °C} \leq 100 \text{ °C}$$

The above example demonstrates the maximum operating capability in the TQFP package.

Note: The power calculations listed are for internal power dissipation only. The external power dissipation due to output pins switching must also be included.

9 Electrical Characteristics and Requirements

The following electrical characteristics are preliminary and are subject to change. Electrical characteristics refer to the behavior of the device under specified conditions. Electrical requirements refer to conditions imposed on the user for proper operation of the device. The parameters below are valid for the conditions described in Section 8.3, Recommended Operating Conditions.

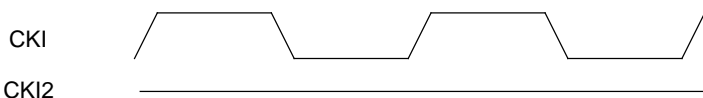
Table 61. Electrical Characteristics and Requirements

| Parameter | Symbol | Min | Max | Unit |
|---|------------------|-----------------------|-----------------------|------|
| Input Voltage: | | | | |
| Low | V _{IL} | -0.3 | 0.3 * V _{DD} | V |
| High | V _{IH} | 0.7 * V _{DD} | V _{DD} + 0.3 | V |
| Input Current (except TMS, TDI): | | | | |
| Low (V _{IL} = 0 V, V _{DD} = 5.25 V) | I _{IL} | -5 | — | μA |
| High (V _{IH} = 5.25 V, V _{DD} = 5.25 V) | I _{IH} | — | 5 | μA |
| Input Current (TMS, TDI): | | | | |
| Low (V _{IL} = 0 V, V _{DD} = 5.25 V) | I _{IL} | -100 | — | μA |
| High (V _{IH} = 5.25 V, V _{DD} = 5.25 V) | I _{IH} | — | 5 | μA |
| Output Low Voltage: | | | | |
| Low (I _{OL} = 2.0 mA) | V _{OL} | — | 0.4 | V |
| Low (I _{OL} = 50 μA) | V _{OL} | — | 0.2 | V |
| Output High Voltage: | | | | |
| High (I _{OH} = -2.0 mA) | V _{OH} | V _{DD} - 0.7 | — | V |
| High (I _{OH} = -50 μA) | V _{OH} | V _{DD} - 0.2 | — | V |
| Output 3-State Current: | | | | |
| Low (V _{DD} = 5.25 V, V _{IL} = 0 V) | I _{OZL} | -10 | — | μA |
| High (V _{DD} = 5.25 V, V _{IH} = 5.25 V) | I _{OZH} | — | 10 | μA |
| Input Capacitance | C _I | — | 5 | pF |

Table 62. Electrical Requirements for Mask-Programmable Input Clock Options

| Parameter | Symbol | Min | Max | Unit | Note |
|--|------------------|-----------------------|-----------------------|------|--------|
| CKI CMOS Level Input Voltage: | | | | | |
| Low | V _{IL} | -0.3 | 0.3 * V _{DD} | V | — |
| High | V _{IH} | 0.7 * V _{DD} | V _{DD} + 0.3 | V | — |
| Small-Signal Peak-to-Peak Voltage (on CKI) | V _{pp} | 0.6 | — | V | Note 1 |
| Small-Signal Input Duty Cycle | DC _{yc} | 45 | 55 | % | Note 2 |
| Small-Signal Input Voltage Range (pins: CKI, CKI2) | V _{in} | 0.2 * V _{DD} | 0.6 * V _{DD} | V | — |
| Small-Signal Buffer Frequency Range | f _{ss} | — | 35 | MHz | — |

Note 1. The small-signal buffer must be used in single-ended mode where an ac waveform (sine or square) is applied to CKI and a dc voltage approximately equal to the average value of CKI is applied to CKI2, as shown in the figure below. The maximum allowable ripple on CKI2 is 100 mV.



Note 2. Duty cycle for a sine wave is defined as the percentage of time during each clock cycle that the voltage on CKI exceeds the voltage on CKI2.

9 Electrical Characteristics and Requirements (continued)

Table 63. PLL Electrical Specifications, VCO Frequency Ranges

| Parameter | Symbol | Min | Max | Unit | Note |
|---|-----------|-----|-----|--------|------|
| VCO frequency range ($V_{DD} = 3\text{ V} \pm 10\%$) | f_{VCO} | 50 | 160 | MHz | 1 |
| Input Jitter at CKI | — | — | 200 | ps-rms | — |

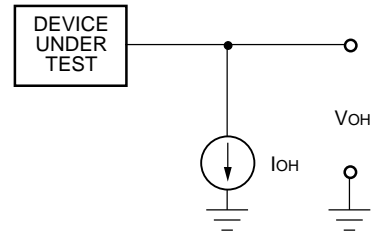
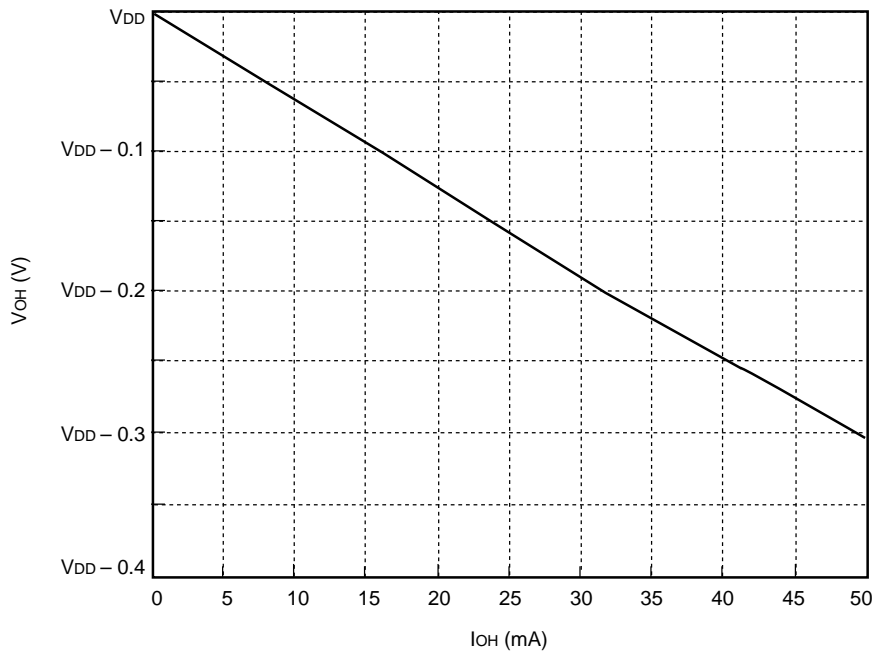
Table 64. PLL Electrical Specifications and pllcr Register Settings

| M | V_{DD} | pllcr13 (ICP) | pllcr12 Reserved | pllcr[11:8] (LF[3:0]) | Typical Lock-in Time (μs) (See Note 2.) |
|-------|---------------|------------------|---------------------|--------------------------|---|
| 23—24 | 2.7 V – 3.3 V | 1 | 0 | 1011 | 30 |
| 21—22 | 2.7 V – 3.3 V | 1 | 0 | 1010 | 30 |
| 19—20 | 2.7 V – 3.3 V | 1 | 0 | 1001 | 30 |
| 16—18 | 2.7 V – 3.3 V | 1 | 0 | 1000 | 30 |
| 12—15 | 2.7 V – 3.3 V | 1 | 0 | 0111 | 30 |
| 8—11 | 2.7 V – 3.3 V | 1 | 0 | 0110 | 30 |
| 2—7 | 2.7 V – 3.3 V | 1 | 0 | 0100 | 30 |

Note 1. The M and N counter values in the **pllcr** register must be set so that the VCO will operate in the appropriate range (see Table 63). Choose the lowest value of N and then the appropriate value of M for $f_{\text{INTERNAL CLOCK}} = f_{\text{CKI}} \times (M/(2N)) = f_{VCO}/2$.

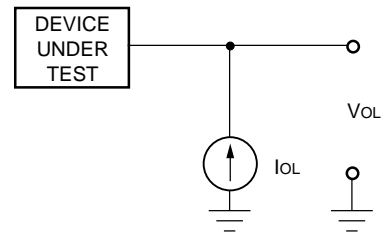
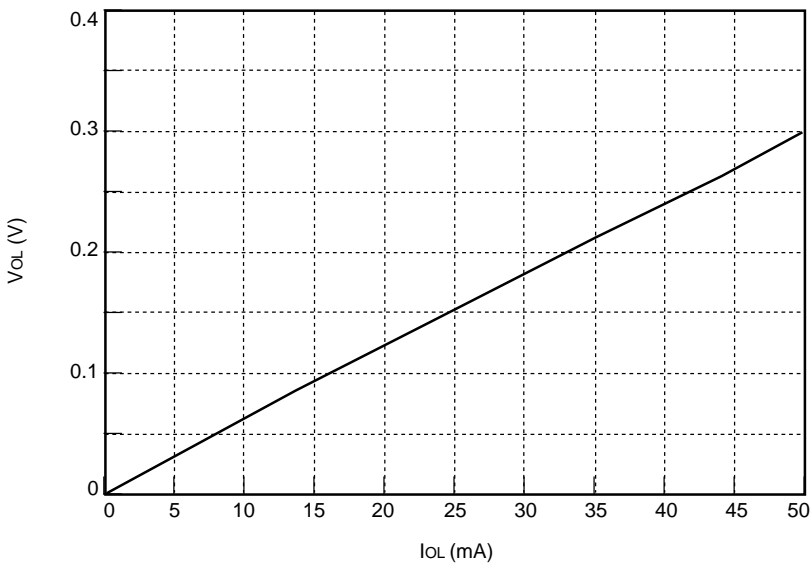
Note 2. Lock-in time represents the time following assertion of the PLLEN bit of the **pllcr** register during which the PLL output clock is unstable. The DSP must operate from the 1X CKI input clock or from the slow ring oscillator while the PLL is locking. Completion of the lock-in interval is indicated by assertion of the LOCK flag.

9 Electrical Characteristics and Requirements (continued)



5-4007 (C).a

Figure 9. Plot of VoH vs. IOH Under Typical Operating Conditions



5-4008 (C).b

Figure 10. Plot of VoL vs. IOL Under Typical Operating Conditions

9 Electrical Characteristics and Requirements (continued)

9.1 Power Dissipation

Power dissipation is highly dependent on DSP program activity and the frequency of operation. The typical power dissipation listed is for a selected application. The following electrical characteristics are preliminary and are subject to change.

Table 65. Power Dissipation and Wake-Up Latency

| Operating Mode (Unused inputs at VDD or VSS) | Typical Power Dissipation (mW) | | Wake-Up Latency | |
|--|--|---|-------------------------------------|---------------------------------|
| | 3 V | 3 V | 3 V | 3 V |
| ECCP Operation CKI = 40 MHz | PECCP | | | |
| | 28.1 | — | — | — |
| | I/O Units ON, ECCP OFF powerc[7:4,0] = 0x01 | I/O Units OFF, ECCP OFF powerc[7:4,0] = 0xF1 | (PLL Not Used During Wake State) | (PLL Used During Wake State) |
| Normal Operation ioc = 0x0180 PLL Disabled CKI & CKO = 40 MHz CMOS Small Signal CKI & CKO = 0 MHz CMOS Small Signal | PDSP | | | |
| | 93.7 | 91.2 | — | — |
| | 96.3 | 93.7 | — | — |
| | 0.17 | 0.17 | — | — |
| | 2.75 | 2.75 | — | — |
| Normal Operation ioc = 0x0180 PLL Enabled pllcc = 0xFC0E CKI = 10 MHz CKO = 40 MHz CMOS Small Signal | PDSP | | | |
| | 96.7 | 94.2 | — | — |
| | 99.3 | 96.7 | — | — |
| Power Management Modes CKO = 40 MHz | | PSLEEP | | |
| Standard Sleep, External Interrupt alf[15] = 1, ioc = 0x0180 PLL Disabled During Sleep CMOS Small Signal | PSLEEP | | | |
| | 14.0 | 9.3 | 3T* | 3T* + tL† |
| | 16.3 | 12.0 | 3T* | 3T* + tL† |
| Standard Sleep, External Interrupt alf[15] = 1, ioc = 0x0180 PLL Enabled During Sleep CMOS Small Signal | PSLEEP | | | |
| | 16.5 | 11.2 | — | 3T* |
| | 18.9 | 14.0 | — | 3T* |
| Sleep with Slow Internal Clock Small Signal Enabled powerc[15:14] = 01, alf[15] = 1, ioc = 0x0180 PLL Disabled During Sleep CMOS Small Signal | PSLEEP | | | |
| | 0.7 | 0.5 | 5.0 μs | 5.0 μs + tL† |
| | 3.7 | 3.5 | 5.0 μs | 5.0 μs + tL† |
| Sleep with Slow Internal Clock Small Signal Enabled powerc[15:14] = 01, alf[15] = 1, ioc = 0x0180 PLL Enabled During Sleep CMOS Small Signal | PSLEEP | | | |
| | 3.3 | 2.9 | — | 5.0 μs |
| | 6.1 | 5.5 | — | 5.0 μs |

* T = CKI clock cycle for 1X input clock option or T = CKI clock cycle divided by M/(2N) for PLL clock option (see Section 4.12).

† tL = PLL lock time (see Table 64).

9 Electrical Characteristics and Requirements (continued)

Table 65. Power Dissipation and Wake-Up Latency (continued)

| Operating Mode (Unused inputs at VDD or VSS) | Typical Power Dissipation (mW) | | Wake-Up Latency | |
|---|--|---|-------------------------------------|---------------------------------|
| | 3 V | 3 V | 3 V | 3 V |
| VDD= | I/O Units ON, ECCP OFF powerc[7:4,0] = 0x01 | I/O Units OFF, ECCP OFF powerc[7:4,0] = 0xF1 | (PLL Not Used During Wake State) | (PLL Used During Wake State) |
| Sleep with Slow Internal Clock Small Signal Disabled powerc[15:14] = 11, alf[15] = 1, ioc = 0x0180 PLL Disabled During Sleep Small Signal | 0.40 | 0.30 | 20 μs | 20 μs + tL [†] |
| Software Stop powerc[15:12] = 0011 PLL Disabled During STOP CMOS | 0.060 | 0.060 | 3T* | 3T* + tL [†] |
| Software Stop powerc[15:12] = 1111 PLL Disabled During STOP Small Signal | 0.060 | 0.060 | 20 μs | 20 μs + tL [†] |
| Hardware Stop (STOP = Vss) powerc[15:12] = 0000 PLL Disabled During STOP CMOS | 0.060 | 0.060 | 3T* | — |
| Small Signal | 1.20 | 1.20 | 3T* | — |
| Hardware Stop (STOP = Vss) powerc[15:12] = 0000 PLL Enabled During STOP CMOS | 2.5 | 2.5 | 3T* | 3T* |
| Small Signal | 3.6 | 3.6 | 3T* | 3T* |

* T = CKI clock cycle for 1X input clock option or T = CKI clock cycle divided by M/(2N) for PLL clock option (see Section 4.12).

† tL = PLL lock time (see Table 64).

The power dissipation listed is for internal power dissipation only. Total power dissipation can be calculated on the basis of the application by adding $C \times V_{DD}^2 \times f$ for each output, where C is the additional load capacitance and f is the output frequency.

Power dissipation due to the input buffers is highly dependent on the input voltage level. At full CMOS levels, essentially no dc current is drawn. However, for levels between the power supply rails, especially at or near the threshold of $V_{DD}/2$, high currents can flow. Although input and I/O buffers may be left untied (since the input voltage levels of the input and I/O buffers are designed to remain at full CMOS levels when not driven by the DSP), it is still recommended that unused input and I/O pins be tied to VSS or VDD through a 10 kΩ resistor to avoid application ambiguities. Further, if I/O pins are tied high or low, they should be pulled fully to VSS or VDD.

WARNING: The device needs to be clocked for at least six CKI cycles during reset after powerup. Otherwise, high currents may flow.

10 Timing Characteristics for 2.7 V Operation

The following timing characteristics and requirements are preliminary information and are subject to change. Timing characteristics refer to the behavior of the device under specified conditions. Timing requirements refer to conditions imposed on the user for proper operation of the device. All timing data is valid for the following conditions:

$T_A = -40\text{ }^\circ\text{C}$ to $+85\text{ }^\circ\text{C}$ (See Section 8.3.)

$V_{DD} = 3\text{ V} \pm 10\%$, $V_{SS} = 0\text{ V}$ (See Section 8.3.)

Capacitance load on outputs (C_L) = 50 pF, except for CKO, where $C_L = 20\text{ pF}$

Output characteristics can be derated as a function of load capacitance (C_L).

All outputs: $0.03\text{ ns/pF} \leq dt/dC_L \leq 0.07\text{ ns/pF}$ for $10 \leq C_L \leq 100\text{ pF}$
at V_{IH} for rising edge and at V_{IL} for falling edge

For example, if the actual load capacitance is 30 pF instead of 50 pF, the derating for a rising edge is $(30 - 50)\text{ pF} \times 0.06\text{ ns/pF} = 1.2\text{ ns}$ **less** than the specified rise time or delay that includes a rise time.

Test conditions for inputs:

- Rise and fall times of 4 ns or less
- Timing reference levels for delays = V_{IH} , V_{IL}

Test conditions for outputs (unless noted otherwise):

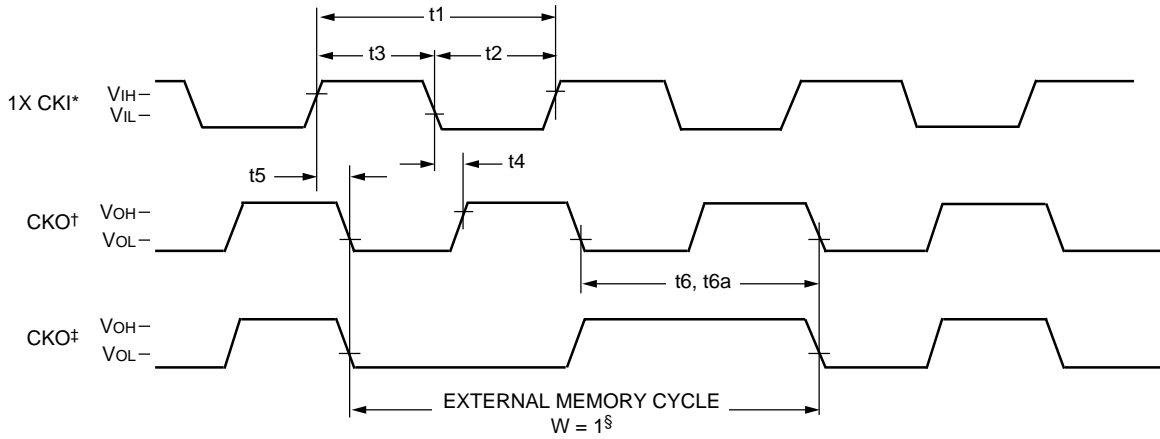
- $C_{LOAD} = 50\text{ pF}$; except for CKO, where $C_{LOAD} = 20\text{ pF}$
- Timing reference levels for delays = V_{IH} , V_{IL}
- 3-state delays measured to the high-impedance state of the output driver

For the timing diagrams, see Table 62 for input clock requirements.

Unless otherwise noted, CKO in the timing diagrams is the free-running CKO.

10 Timing Characteristics for 2.7 V Operation (continued)

10.1 DSP Clock Generation



5-4009 (C).a

* See Table 62 for input clock electrical requirements.

† Free-running clock.

‡ Wait-stated clock (see Table 38).

§ W = number of wait-states.

Figure 11. I/O Clock Timing Diagram

Table 66. Timing Requirements for Input Clock

| Abbreviated Reference | Parameter | 19.2 ns and 12.5 ns* | | |
|-----------------------|----------------------------------|----------------------|-----|------|
| | | Min | Max | Unit |
| t ₁ | Clock In Period (high to high) | 20 | —† | ns |
| t ₂ | Clock In Low Time (low to high) | 10 | — | ns |
| t ₃ | Clock In High Time (high to low) | 10 | — | ns |

* Device speeds greater than 50 MIPS do not support 1X operation. Use the PLL.

† Device is fully static, t₁ is tested at 100 ns for 1X input clock option, and memory hold time is tested at 0.1 s.

Table 67. Timing Characteristics for Input Clock and Output Clock

| Abbreviated Reference | Parameter | 19.2 ns | | 12.5 ns | | Unit |
|-----------------------|---|---------|-----|---------|-----|------|
| | | Min | Max | Min | Max | |
| t ₄ | Clock Out High Delay | — | 14 | — | 10 | ns |
| t ₅ | Clock Out Low Delay (high to low) | — | 14 | — | 10 | ns |
| t ₆ | Clock Out Period (low to low) | T* | — | T* | — | ns |
| t _{6a} | Clock Out Period with SLOWCKI Bit Set in powerc Register (low to low) | 0.74 | 3.8 | 0.74 | 3.8 | μs |

* T = internal clock period, set by CKI or by CKI and the PLL parameters.

10 Timing Characteristics for 2.7 V Operation (continued)

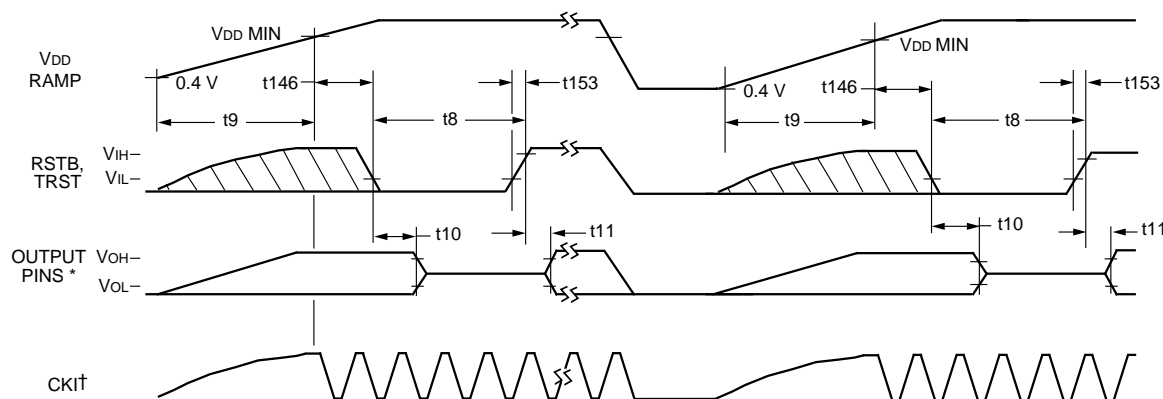
10.2 Reset Circuit

The DSP1628 has two external reset pins: RSTB and TRST. At initial powerup, or if the supply voltage falls below VDD MIN* and a device reset is required, both TRST and RSTB must be asserted to initialize the device. Figure 12 shows two separate events:

1. Chip reset at initial powerup.
2. Chip reset following a drop in power supply.

Note: The TRST pin must be asserted even if the JTAG controller is not used by the application.

* See Table 60, Recommended Operating Conditions.



5-4010 (C).a

* When both INT0 and RSTB are asserted, all output and bidirectional pins (except TDO, which 3-states by JTAG control) are put in a 3-state condition. With RSTB asserted and INT0 not asserted, EROM, ERAMHI, ERAMLO, IO, and RWN outputs remain high, and CKO remains a free-running clock.

† See Table 62 for input clock electrical requirements.

Figure 12. Powerup Reset and Chip Reset Timing Diagram

Table 68. Timing Requirements for Powerup Reset and Chip Reset

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|--------------|-----|------|
| t8 | RSTB and TRST Reset Pulse (low to high) | 6T | — | ns |
| t9 | VDD Ramp | — | 10 | ms |
| t146 | VDD MIN to RSTB Low | CMOS | 2T | ns |
| | | Small-signal | 20 | μs |
| t153 | RSTB (low to high) | — | 54 | ns |

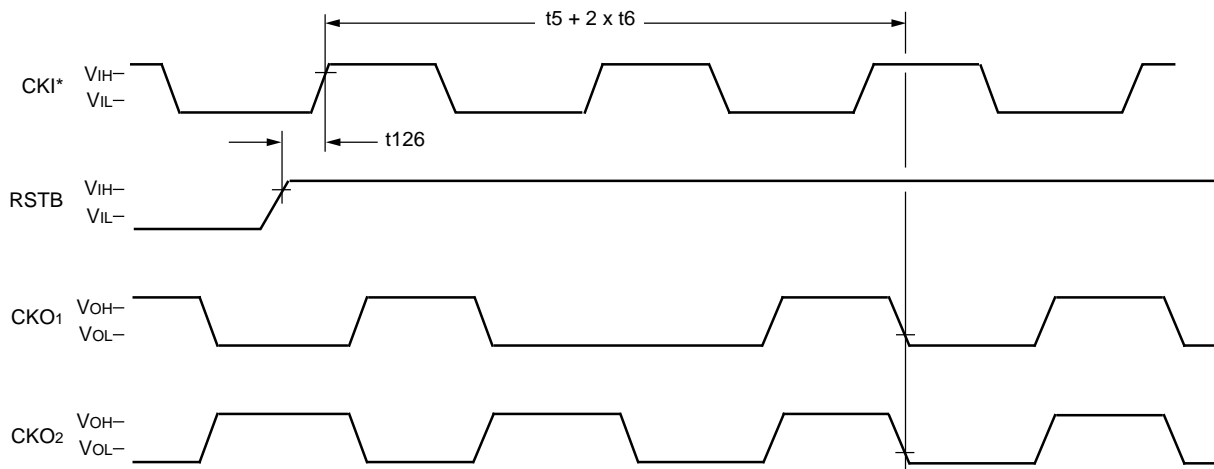
Table 69. Timing Characteristics for Powerup Reset and Chip Reset

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|------------------------------------|-----|-----|------|
| t10 | RSTB Disable Time (low to 3-state) | — | 100 | ns |
| t11 | RSTB Enable Time (high to valid) | — | 100 | ns |

Note: The device needs to be clocked for at least six CKI cycles during reset after powerup. Otherwise, high currents may flow.

10 Timing Characteristics for 2.7 V Operation (continued)

10.3 Reset Synchronization



5-4011 (C).a

* See Table 62 for input clock electrical requirements.

Note 1: CKO1 and CKO2 are two possible CKO states before reset. CKO is free-running.

Note 2: If the rising edge of RSTB (low to high) is captured instead by the falling edge of CKO (high to low), CKO and CKI will be in-phase at $t_5 + 2 \times t_6$.

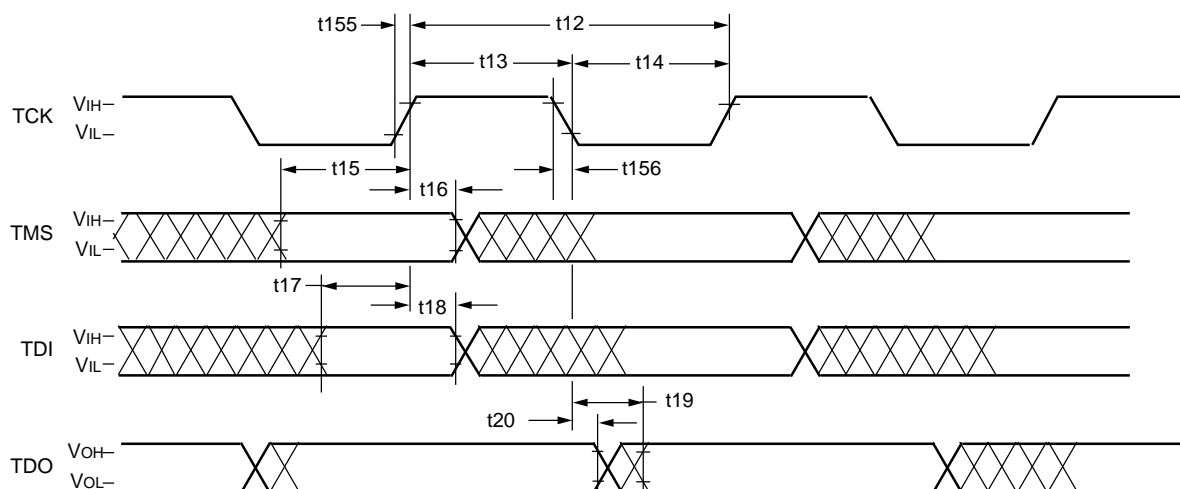
Figure 13. Reset Synchronization Timing

Table 70. Timing Requirements for Reset Synchronization Timing

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|----------------------------|-----|-----------|------|
| t126 | Reset Setup (high to high) | 3 | $T/2 - 1$ | ns |

10 Timing Characteristics for 2.7 V Operation (continued)

10.4 JTAG I/O Specifications



5-4017 (C)

Figure 14. JTAG Timing Diagram

Table 71. Timing Requirements for JTAG Input/Output

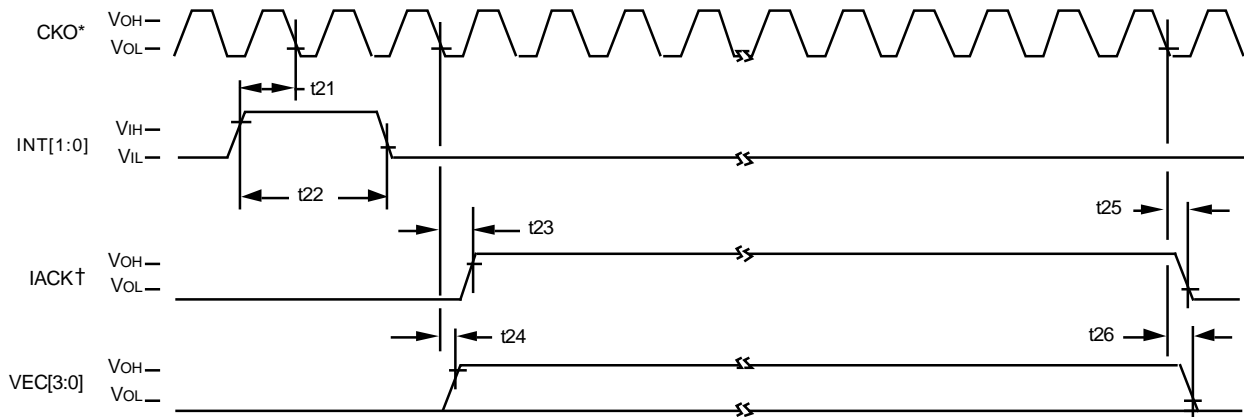
| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--|------|-----|------|
| t12 | TCK Period (high to high) | 50 | — | ns |
| t13 | TCK High Time (high to low) | 22.5 | — | ns |
| t14 | TCK Low Time (low to high) | 22.5 | — | ns |
| t155 | TCK Rise Transition Time (low to high) | 0.6 | — | V/ns |
| t156 | TCK Fall Transition Time (high to low) | 0.6 | — | V/ns |
| t15 | TMS Setup Time (valid to high) | 7.5 | — | ns |
| t16 | TMS Hold Time (high to invalid) | 2 | — | ns |
| t17 | TDI Setup Time (valid to high) | 7.5 | — | ns |
| t18 | TDI Hold Time (high to invalid) | 2 | — | ns |

Table 72. Timing Characteristics for JTAG Input/Output

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---------------------------|-----|-----|------|
| t19 | TDO Delay (low to valid) | — | 19 | ns |
| t20 | TDO Hold (low to invalid) | 0 | — | ns |

10 Timing Characteristics for 2.7 V Operation (continued)

10.5 Interrupt



5-4018 (C).

* CKO is free-running.

† IACK assertion is guaranteed to be enclosed by VEC[3:0] assertion.

Figure 15. Interrupt Timing Diagram

Table 73. Timing Requirements for Interrupt

Note: Interrupt is asserted during an interruptible instruction and no other pending interrupts.

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|----------------------------------|-----|-----|------|
| t21 | Interrupt Setup (high to low) | 19 | — | ns |
| t22 | INT Assertion Time (high to low) | 2T | — | ns |

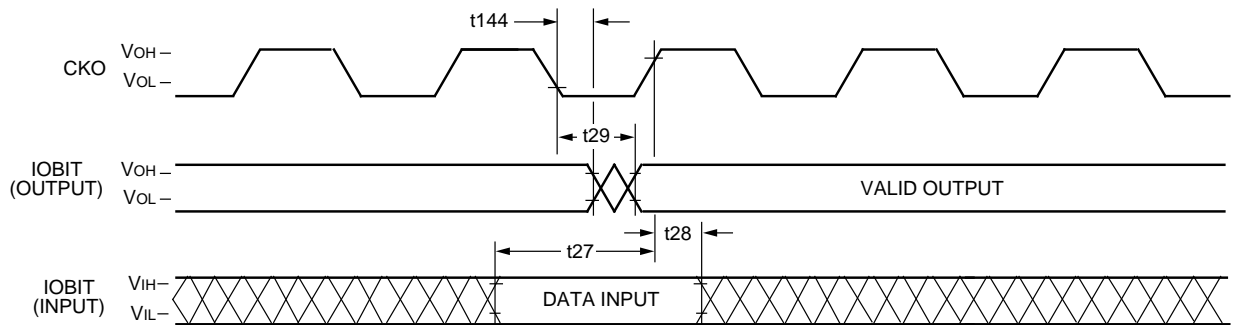
Table 74. Timing Characteristics for Interrupt

Note: Interrupt is asserted during an interruptible instruction and no other pending interrupts.

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|-----------------------------------|-----|----------|------|
| t23 | IACK Assertion Time (low to high) | — | T/2 + 10 | ns |
| t24 | VEC Assertion Time (low to high) | — | 12.5 | ns |
| t25 | IACK Invalid Time (low to low) | — | 10 | ns |
| t26 | VEC Invalid Time (low to low) | — | 12.5 | ns |

10 Timing Characteristics for 2.7 V Operation (continued)

10.6 Bit Input/Output (BIO)



5-4019 (C).a

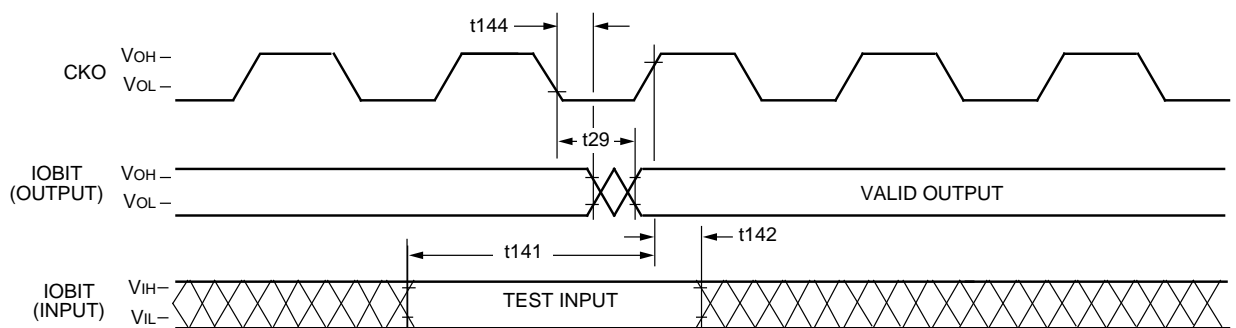
Figure 16. Write Outputs Followed by Read Inputs (cbit = Immediate; a1 = sbit)

Table 75. Timing Requirements for BIO Input Read

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t27 | IOBIT Input Setup Time (valid to high) | 15 | — | ns |
| t28 | IOBIT Input Hold Time (high to invalid) | 0 | — | ns |

Table 76. Timing Characteristics for BIO Output

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t29 | IOBIT Output Valid Time (low to valid) | — | 9 | ns |
| t144 | IOBIT Output Hold Time (low to invalid) | 1 | — | ns |



5-4019 (C).b

Figure 17. Write Outputs and Test Inputs (cbit = Immediate)

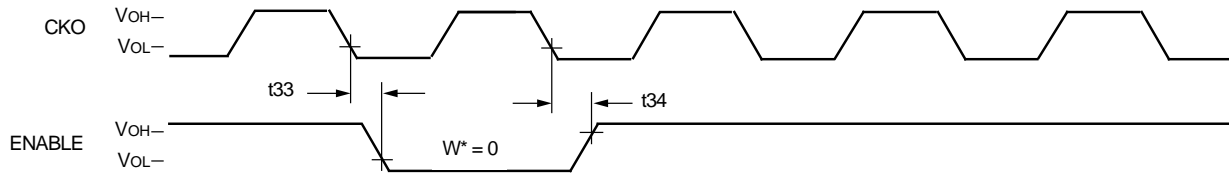
Table 77. Timing Requirements for BIO Input Test

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--|-----|-----|------|
| t141 | IOBIT Input Setup Time (valid to low) | 15 | — | ns |
| t142 | IOBIT Input Hold Time (low to invalid) | 0 | — | ns |

10 Timing Characteristics for 2.7 V Operation (continued)

10.7 External Memory Interface

The following timing diagrams, characteristics, and requirements do not apply to interactions with delayed external memory enables unless so stated. See the *DSP1611/17/18/27 Digital Signal Processor Information Manual* for a detailed description of the external memory interface including other functional diagrams.



5-4020 (C).b

* W = number of wait-states.

Figure 18. Enable Transition Timing

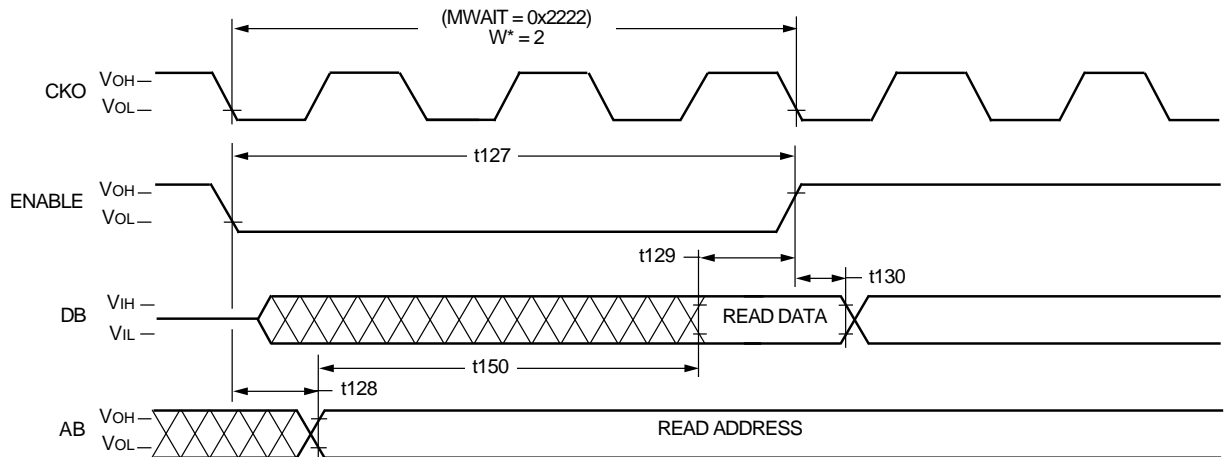
Table 78. Timing Characteristics for External Memory Enables (EROM, ERAMHI, IO, ERAMLO)

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--------------------------------------|-----|-----|------|
| t33 | CKO to ENABLE Active (low to low) | 0 | 5 | ns |
| t34 | CKO to ENABLE Inactive (low to high) | -1 | 4.5 | ns |

Table 79. Timing Characteristics for Delayed External Memory Enables (ioc = 0x000F)

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|---------|------|
| t33 | CKO to Delayed ENABLE Active (low to low) | T/2 | T/2 + 7 | ns |

10 Timing Characteristics for 2.7 V Operation (continued)



5-4021 (C).a

* W = number of wait-states.

Figure 19. External Memory Data Read Timing Diagram

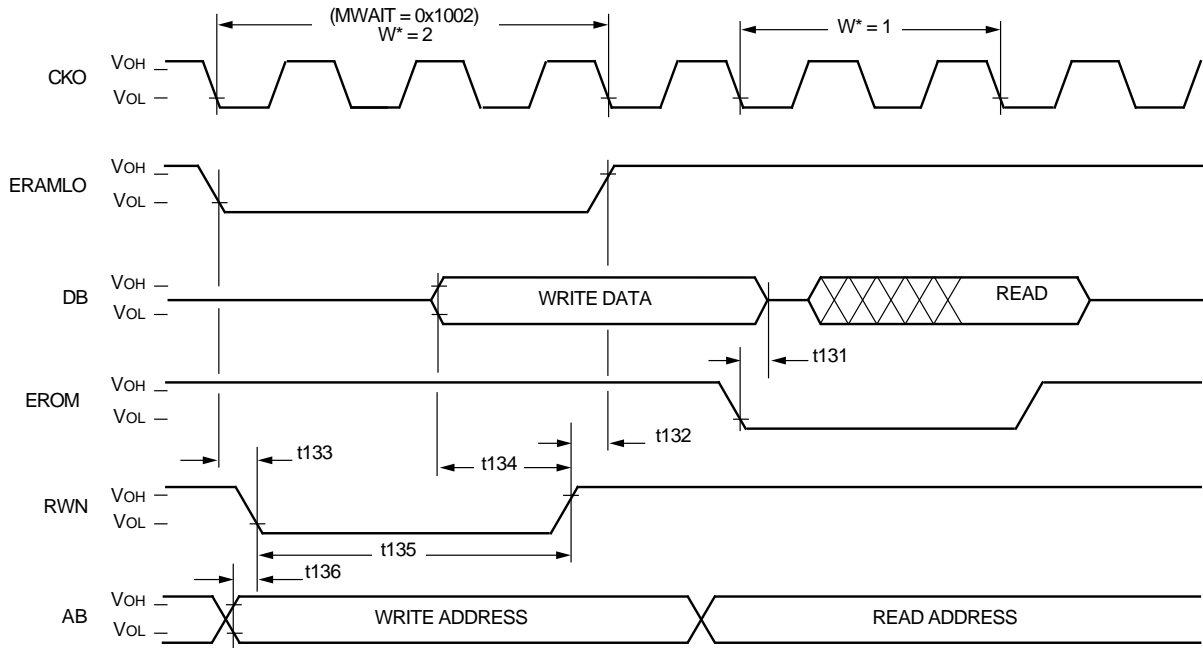
Table 80. Timing Characteristics for External Memory Access

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|-------------------------------------|------------------|-----|------|
| t127 | Enable Width (low to high) | $T(1 + W) - 1.5$ | — | ns |
| t128 | Address Valid (enable low to valid) | — | 2.5 | ns |

Table 81. Timing Requirements for External Memory Read (EROM, ERAMHI, IO, ERAMLO)

| Abbreviated Reference | Parameter | 19.2 ns | | 12.5 ns | | Unit |
|-----------------------|--|---------|-----------------|---------|-----------------|------|
| | | Min | Max | Min | Max | |
| t129 | Read Data Setup (valid to enable high) | 15 | — | 13 | — | ns |
| t130 | Read Data Hold (enable high to hold) | 0 | — | 0 | — | ns |
| t150 | External Memory Access Time (valid to valid) | — | $T(1 + W) - 15$ | — | $T(1 + W) - 14$ | ns |

10 Timing Characteristics for 2.7 V Operation (continued)



5-4022 (C).a

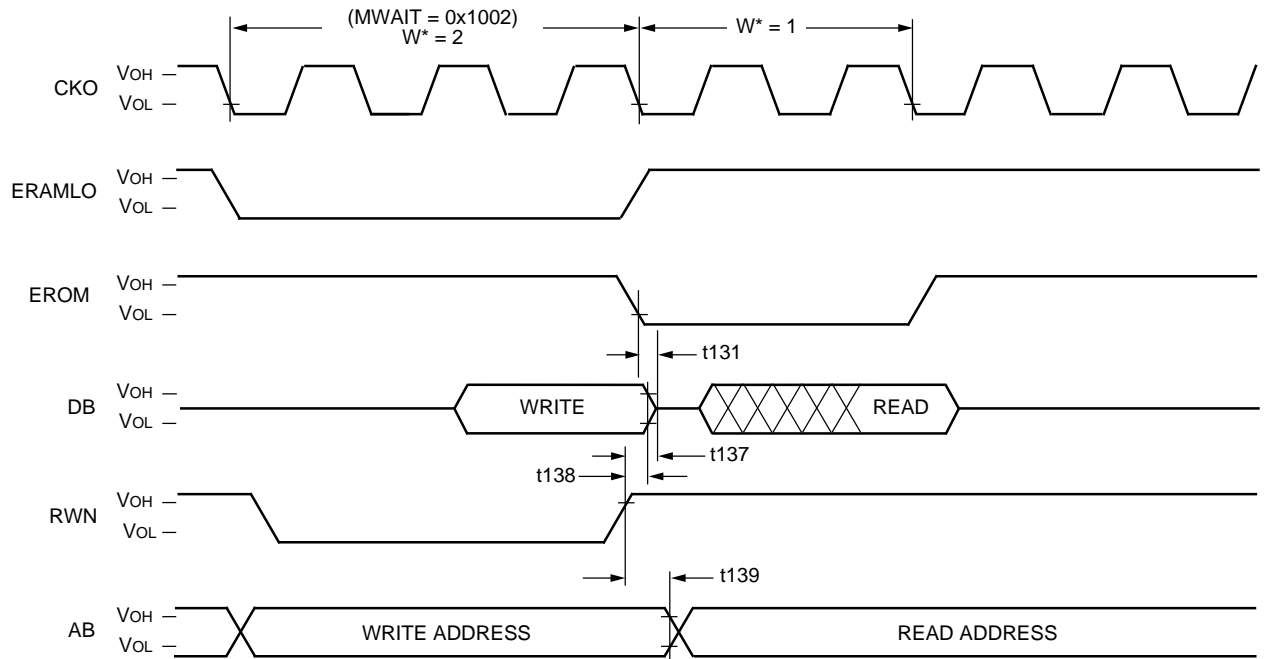
* W = number of wait-states.

Figure 20. External Memory Data Write Timing Diagram

Table 82. Timing Characteristics for External Memory Data Write (All Enables)

| Abbreviated Reference | Parameter | 19.2 ns | | 12.5 ns | | Unit |
|-----------------------|--|------------------|-----|------------------|-----|------|
| | | Min | Max | Min | Max | |
| t131 | Write Overlap (enable low to 3-state) | — | 0 | — | 0 | ns |
| t132 | RWN Advance (RWN high to enable high) | 0 | — | 0 | — | ns |
| t133 | RWN Delay (enable low to RWN low) | 0 | — | 0 | — | ns |
| t134 | Write Data Setup (data valid to RWN high) | $T(1 + W)/2 - 4$ | — | $T(1 + W)/2 - 3$ | — | ns |
| t135 | RWN Width (low to high) | $T(1 + W) - 5$ | — | $T(1 + W) - 4$ | — | ns |
| t136 | Write Address Setup (address valid to RWN low) | 0 | — | 0 | — | ns |

10 Timing Characteristics for 2.7 V Operation (continued)



* W = number of wait-states.

5-4023 (C).a

Figure 21. Write Cycle Followed by Read Cycle

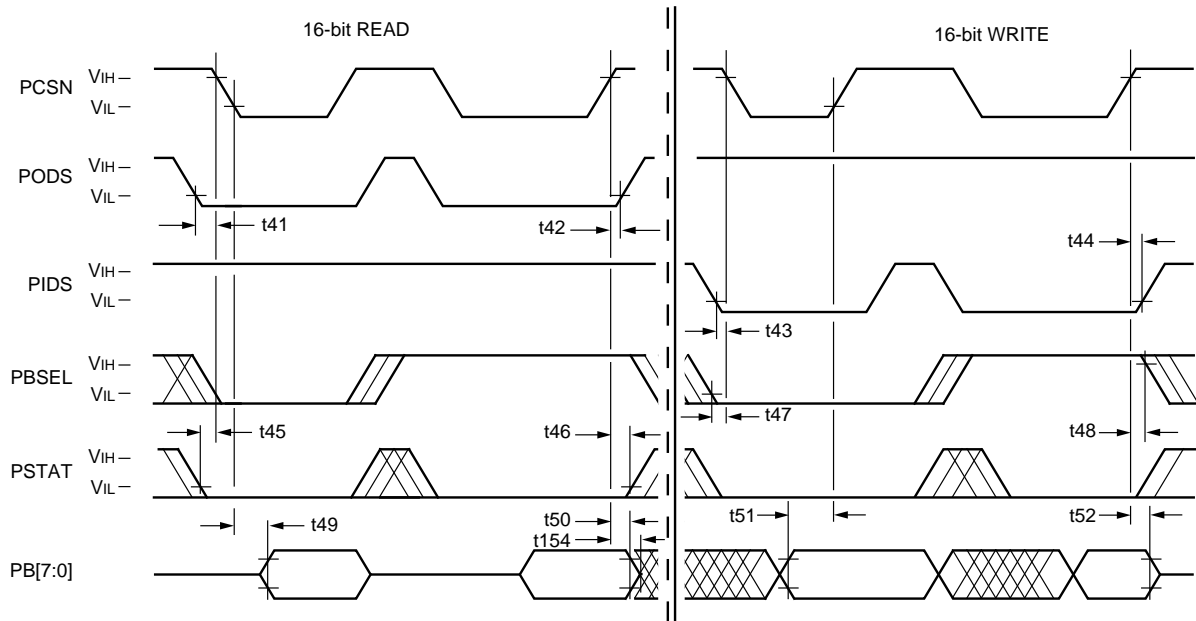
Table 83. Timing Characteristics for Write Cycle Followed by Read Cycle

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t131 | Write Overlap (enable low to 3-state) | — | 0 | ns |
| t137 | Write Data 3-state (RWN high to 3-state) | — | 2 | ns |
| t138 | Write Data Hold (RWN high to data hold) | 0 | — | ns |
| t139 | Write Address Hold (RWN high to address hold) | 0 | — | ns |

10 Timing Characteristics for 2.7 V Operation (continued)

10.8 PHIF Specifications

For the PHIF, read means read by the external user (output by the DSP); write is similarly defined. The 8-bit reads/writes are identical to one-half of a 16-bit access.



5-4036 (C)

Figure 22. PHIF Intel Mode Signaling (Read and Write) Timing Diagram

Table 84. Timing Requirements for PHIF Intel Mode Signaling

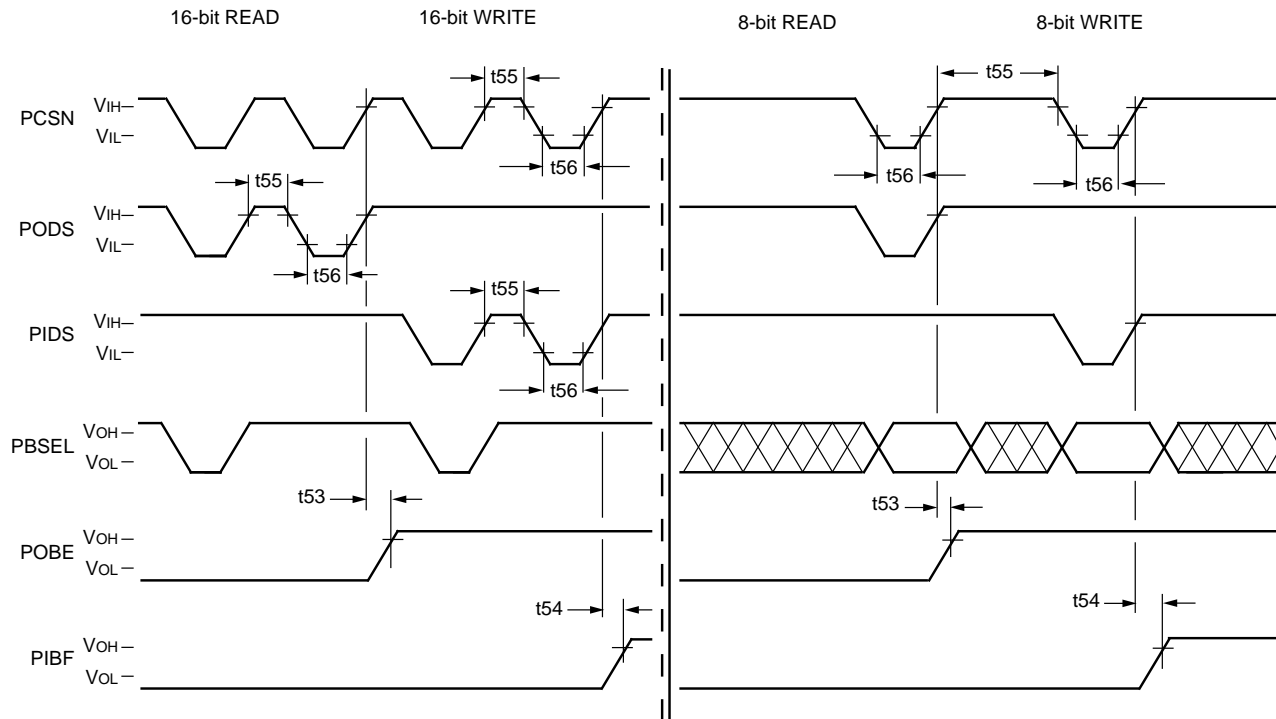
| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t41 | PODS to PCSN Setup (low to low) | 0 | — | ns |
| t42 | PCSN to PODS Hold (high to high) | 0 | — | ns |
| t43 | PIDS to PCSN Setup (low to low) | 0 | — | ns |
| t44 | PCSN to PIDS Hold (high to high) | 0 | — | ns |
| t45* | PSTAT to PCSN Setup (valid to low) | 4 | — | ns |
| t46* | PCSN to PSTAT Hold (high to invalid) | 0 | — | ns |
| t47* | PBSEL to PCSN Setup (valid to low) | 6 | — | ns |
| t48* | PCSN to PBSEL Hold (high to invalid) | 0 | — | ns |
| t51* | PB Write to PCSN Setup (valid to high) | 10 | — | ns |
| t52* | PCSN to PB Write Hold (high to invalid) | 4 | — | ns |

Table 85. Timing Characteristics for PHIF Intel Mode Signaling

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t49* | PCSN to PB Read (low to valid) | — | 12 | ns |
| t50* | PCSN to PB Read Hold (high to invalid) | 0 | — | ns |
| t154 | PCSN to PB Read 3-state (high to 3-state) | — | 8 | ns |

* This timing diagram for the PHIF port shows accesses using the PCSN signal to initiate and complete a transaction. The transactions can also be initiated and completed with the PIDS and PODS signals. An output transaction (read) is initiated by PCSN or PODS going low, whichever comes last. For example, the timing requirements referenced to PCSN going low, t45 and t49, should be referenced to PODS going low, if PODS goes low after PCSN. An output transaction is completed by PCSN or PODS going high, whichever comes first. An input transaction is initiated by PCSN or PIDS going low, whichever comes last. An input transaction is completed by PCSN or PIDS going high, whichever comes first. All requirements referenced to PCSN apply to PIDS or PODS, if PIDS or PODS is the controlling signal.

10 Timing Characteristics for 2.7 V Operation (continued)



5-4037 (C).a

Figure 23. PHIF Intel Mode Signaling (Pulse Period and Flags) Timing Diagram

Table 86. Timing Requirements for PHIF Intel Mode Signaling

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--|------|-----|------|
| t55 | PCSN/PODS/PIDS Pulse Width (high to low) | 20.5 | — | ns |
| t56 | PCSN/PODS/PIDS Pulse Width (low to high) | 20.5 | — | ns |

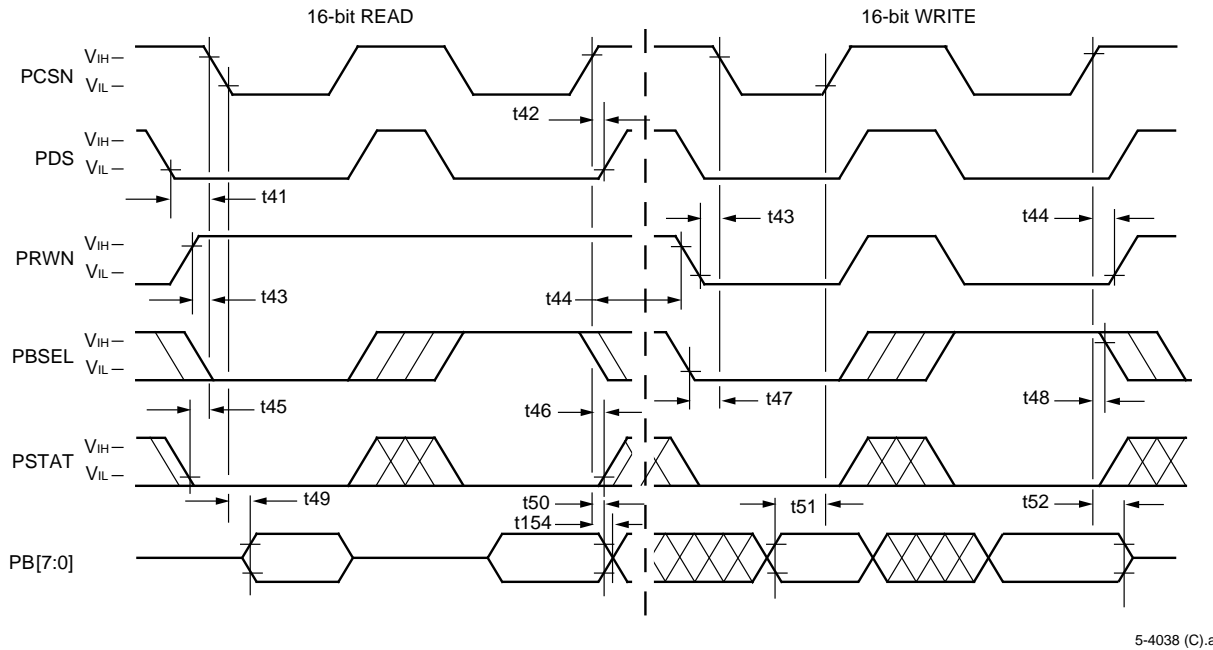
Table 87. Timing Characteristics for PHIF Intel Mode Signaling

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|-----------------------------------|-----|-----|------|
| t53* | PCSN/PODS to POBE† (high to high) | — | 17 | ns |
| t54* | PCSN/PIDS to PIBF† (high to high) | — | 17 | ns |

* t53 should be referenced to the rising edge of PCSN or PODS, whichever comes first. t54 should be referenced to the rising edge of PCSN or PIDS, whichever comes first.

† POBE and PIBF may be programmed to be the opposite logic levels shown in the diagram (positive assertion levels shown). t53 and t54 apply to the inverted levels as well as those shown.

10 Timing Characteristics for 2.7 V Operation (continued)



5-4038 (C).a

Figure 24. PHIF Motorola Mode Signaling (Read and Write) Timing Diagram

Table 88. Timing Requirements for PHIF Motorola Mode Signaling

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t41 | PDS [†] to PCSN Setup (valid to low) | 0 | — | ns |
| t42 | PCSN to PDS [†] Hold (high to invalid) | 0 | — | ns |
| t43 | PRWN to PCSN Setup (valid to low) | 4 | — | ns |
| t44 | PCSN to PRWN Hold (high to invalid) | 0 | — | ns |
| t45* | PSTAT to PCSN Setup (valid to low) | 4 | — | ns |
| t46* | PCSN to PSTAT Hold (high to invalid) | 0 | — | ns |
| t47* | PBSEL to PCSN Setup (valid to low) | 6 | — | ns |
| t48* | PCSN to PBSEL Hold (high to invalid) | 0 | — | ns |
| t51* | PB Write to PCSN Setup (valid to high) | 10 | — | ns |
| t52* | PCSN to PB Write Hold (high to invalid) | 4 | — | ns |

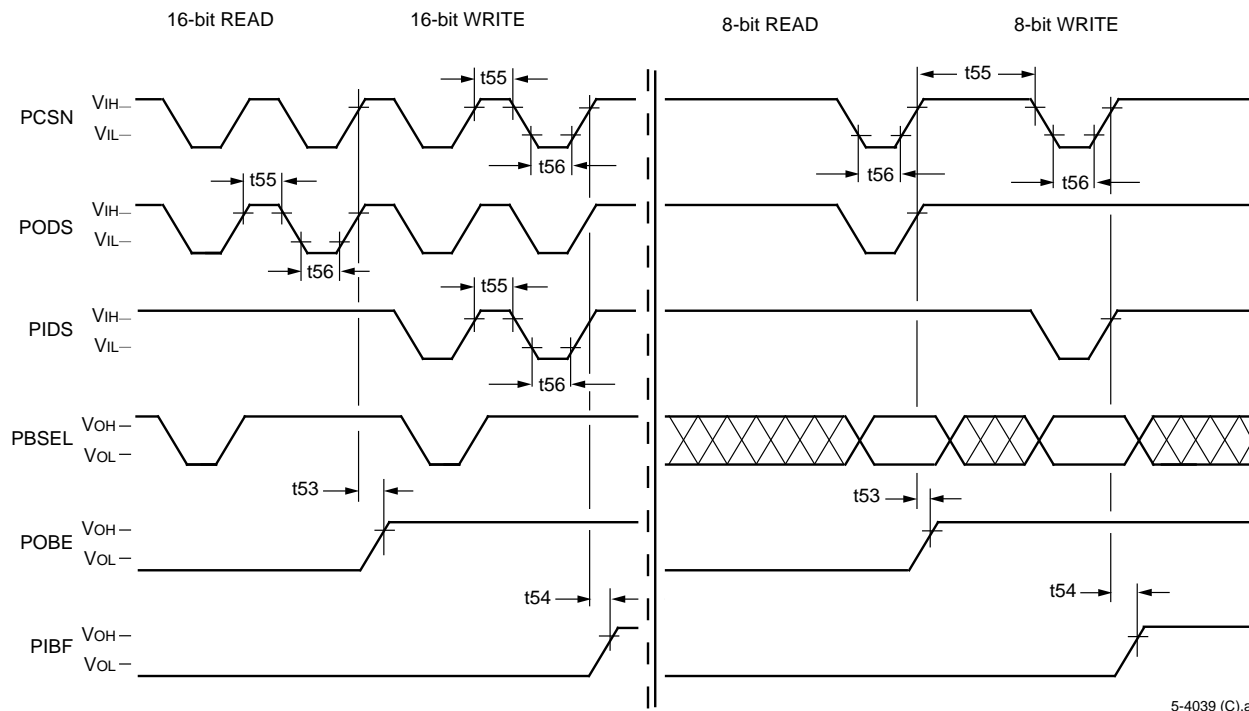
Table 89. Timing Characteristics for PHIF Motorola Mode Signaling

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t49* | PCSN to PB Read (low to valid) | — | 12 | ns |
| t50* | PCSN to PB Read Hold (high to invalid) | 0 | — | ns |
| t154 | PCSN to PB Read 3-state (high to 3-state) | — | 8 | ns |

* This timing diagram for the PHIF port shows accesses using the PCSN signal to initiate and complete a transaction. The transactions can also be initiated and completed with the PDS signal. An input/output transaction is initiated by PCSN or PDS going low, whichever comes last. For example, the timing requirements referenced to PCSN going low, t45 and t49, should be referenced to PDS going low, if PDS goes low after PCSN. An input/output transaction is completed by PCSN or PDS going high, whichever comes first. All requirements referenced to PCSN should be referenced to PDS, if PDS is the controlling signal. PRWN should never be used to initiate or complete a transaction.

† PDS is programmable to be active-high or active-low. It is shown active-low in Figures 24 and 25. POBE and PIBF may be programmed to be the opposite logic levels shown in the diagram. t53 and t54 apply to the inverted levels as well as those shown.

10 Timing Characteristics for 2.7 V Operation (continued)



5-4039 (C).a

Figure 25. PHIF Motorola Mode Signaling (Pulse Period and Flags) Timing Diagram

Table 90. Timing Characteristics for PHIF Motorola Mode Signaling

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t53* | PCSN/PDS [†] to POBE [†] (high to high) | — | 17 | ns |
| t54* | PCSN/PDS [†] to PIBF [†] (high to high) | — | 17 | ns |

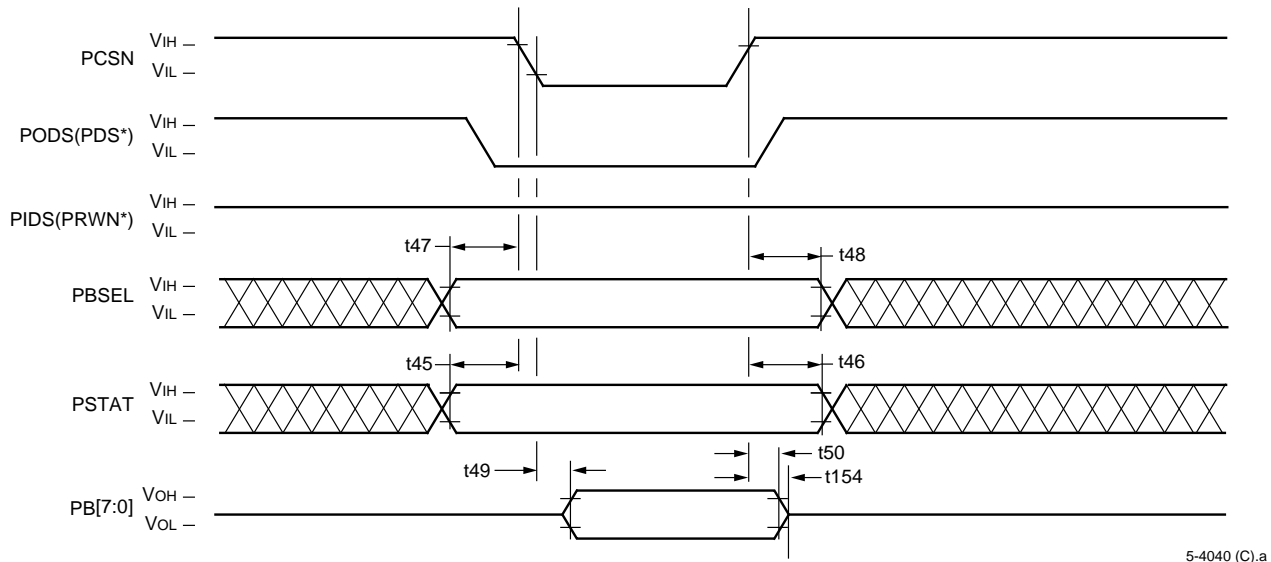
* An input/output transaction is initiated by PCSN or PDS going low, whichever comes last. For example, t53 and t54 should be referenced to PDS going low, if PDS goes low after PCSN. An input/output transaction is completed by PCSN or PDS going high, whichever comes first. All requirements referenced to PCSN should be referenced to PDS, if PDS is the controlling signal. PRWN should never be used to initiate or complete a transaction.

† PDS is programmable to be active-high or active-low. It is shown active-low in Figures 24 and 25. POBE and PIBF may be programmed to be the opposite logic levels shown in the diagram. t53 and t54 apply to the inverted levels as well as those shown.

Table 91. Timing Requirements for PHIF Motorola Mode Signaling

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t55 | PCSN/PDS/PRWN Pulse Width (high to low) | 20 | — | ns |
| t56 | PCSN/PDS/PRWN Pulse Width (low to high) | 20 | — | ns |

10 Timing Characteristics for 2.7 V Operation (continued)



* Motorola mode signal name.

5-4040 (C).a

Figure 26. PHIF Intel or Motorola Mode Signaling (Status Register Read) Timing Diagram

Table 92. Timing Requirements for Intel and Motorola Mode Signaling (Status Register Read)

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--------------------------------------|-----|-----|------|
| t45† | PSTAT to PCSN Setup (valid to low) | 4 | — | ns |
| t46‡ | PCSN to PSTAT Hold (high to invalid) | 0 | — | ns |
| t47† | PBSEL to PCSN Setup (valid to low) | 6 | — | ns |
| t48‡ | PCSN to PBSEL Hold (high to invalid) | 0 | — | ns |

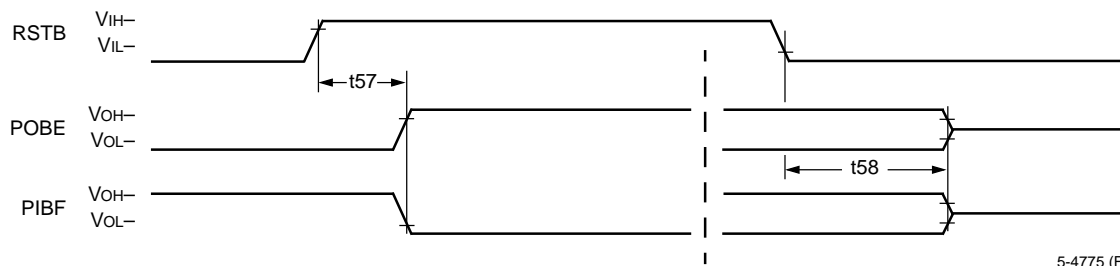
Table 93. Timing Characteristics for Intel and Motorola Mode Signaling (Status Register Read)

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--|-----|-----|------|
| t49† | PCSN to PB Read (low to valid) | — | 12 | ns |
| t50‡ | PCSN to PB Read Hold (high to invalid) | 0 | — | ns |
| t154 | PCSN to PB 3-state (high to 3-state) | — | 8 | ns |

† t45, t47, and t49 are referenced to the falling edge of PCSN or PODS(PDS), whichever occurs last.

‡ t46, t48, t154, and t50 are referenced to the rising edge of PCSN or PODS(PDS), whichever occurs first.

10 Timing Characteristics for 2.7 V Operation (continued)



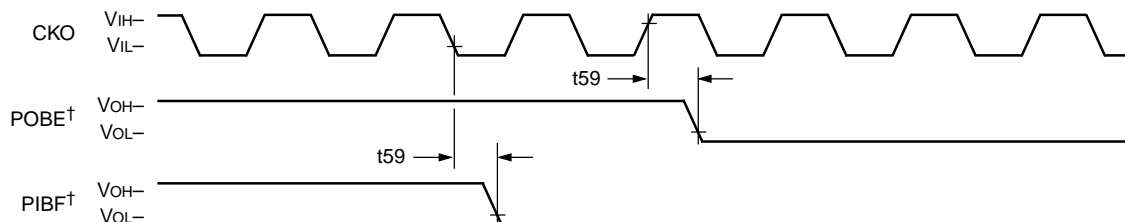
5-4775 (F)

Figure 27. PHIF, PIBF, and POBE Reset Timing Diagram

Table 94. PHIF Timing Characteristics for PHIF, PIBF, and POBE Reset

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--|-----|-----|------|
| t57 | RSTB Disable to POBE/PIBF* (high to valid) | — | 25 | ns |
| t58 | RSTB Enable to POBE/PIBF* (low to invalid) | 3 | 25 | ns |

* After reset, POBE and PIBF always go to the levels shown, indicating output buffer empty and input buffer empty. The DSP program, however, may later invert the definition of the logic levels for POBE and PIBF. t57 and t58 continue to apply.



5-4776 (F)

† POBE and PIBF can be programmed to be active-high or active-low. They are shown active-high. The timing characteristic for active-low is the same as for active-high.

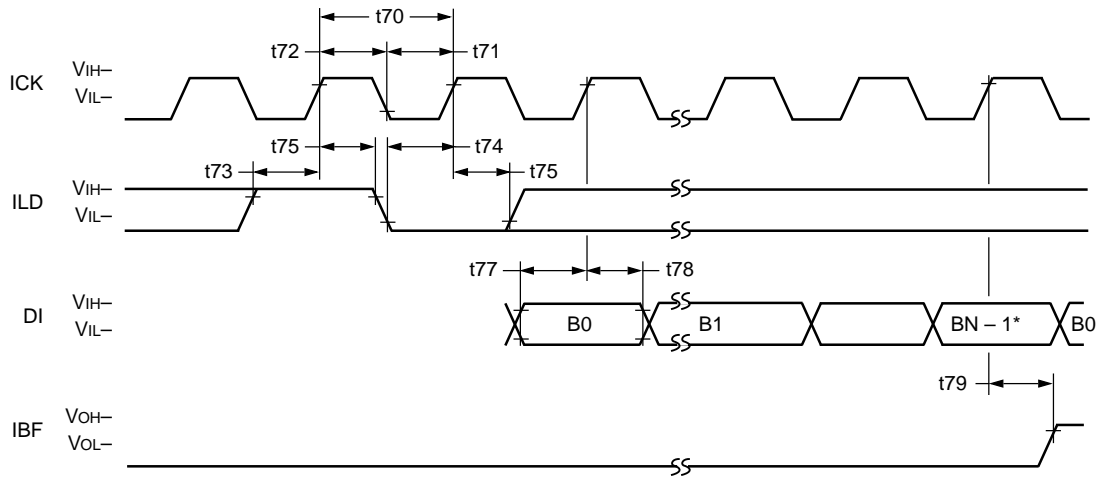
Figure 28. PHIF, PIBF, and POBE Disable Timing Diagram

Table 95. PHIF Timing Characteristics for POBE and PIBF Disable

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|---|-----|-----|------|
| t59 | CKO to POBE/PIBF* Disable (high/low to disable) | — | 20 | ns |

10 Timing Characteristics for 2.7 V Operation (continued)

10.9 Serial I/O Specifications



5-4777 (F)

* N = 16 or 8 bits.

Figure 29. SIO Passive Mode Input Timing Diagram

Table 96. Timing Requirements for Serial Inputs

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--|-----|----------------|------|
| t70 | Clock Period (high to high) [†] | 40 | — [‡] | ns |
| t71 | Clock Low Time (low to high) | 18 | — | ns |
| t72 | Clock High Time (high to low) | 18 | — | ns |
| t73 | Load High Setup (high to high) | 8 | — | ns |
| t74 | Load Low Setup (low to high) | 8 | — | ns |
| t75 | Load High Hold (high to invalid) | 0 | — | ns |
| t77 | Data Setup (valid to high) | 7 | — | ns |
| t78 | Data Hold (high to invalid) | 0 | — | ns |

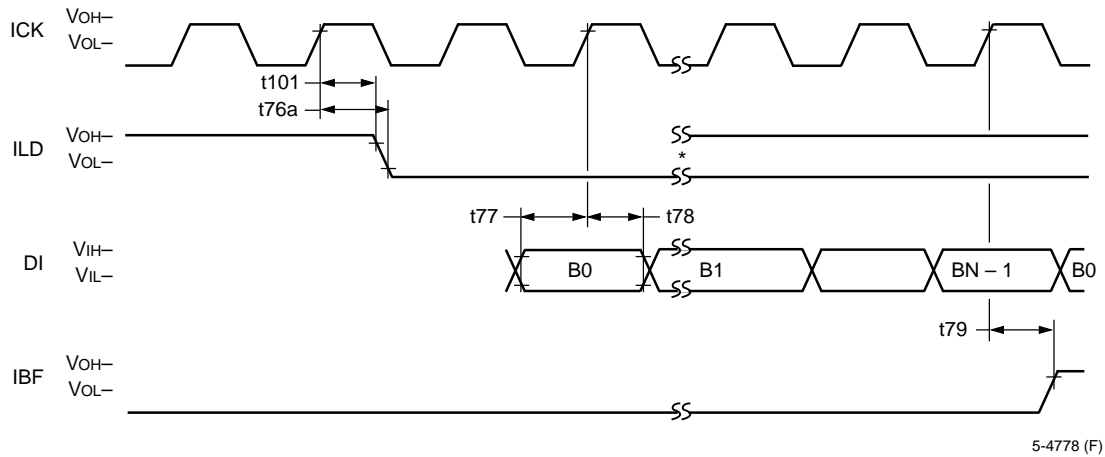
[†] For multiprocessor mode, see note in Section 10.10.

[‡] Device is fully static; t70 is tested at 200 ns.

Table 97. Timing Characteristics for Serial Outputs

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--------------------------|-----|-----|------|
| t79 | IBF Delay (high to high) | — | 35 | ns |

10 Timing Characteristics for 2.7 V Operation (continued)



* ILD goes high during bit 6 (of 0:15), N = 8 or 16.

Figure 30. SIO Active Mode Input Timing Diagram

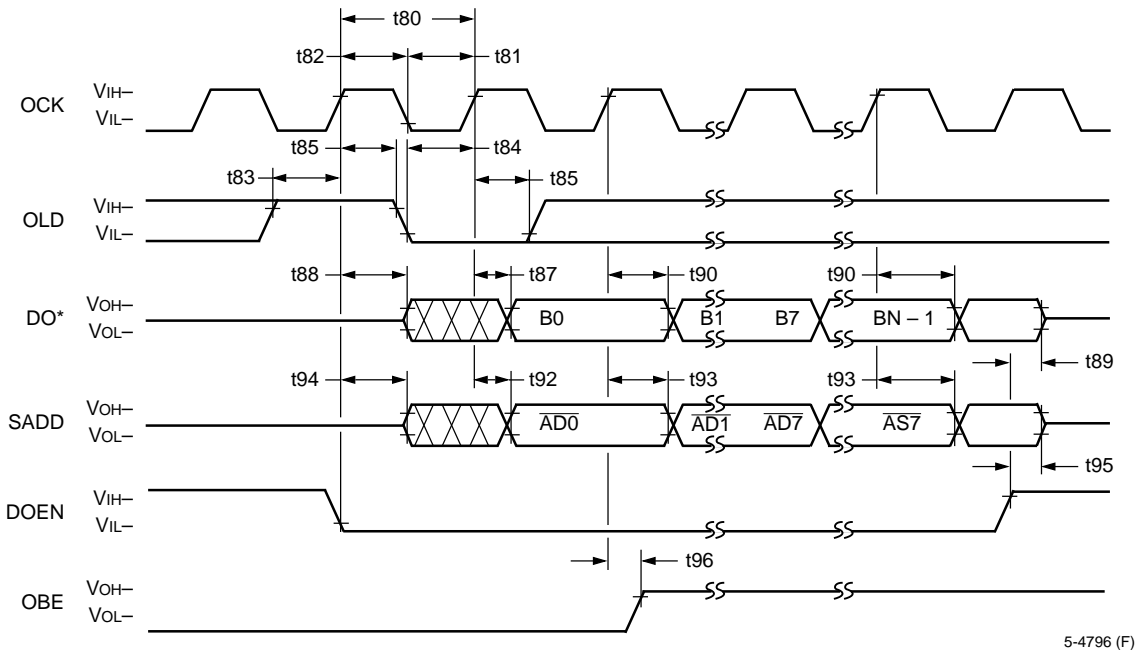
Table 98. Timing Requirements for Serial Inputs

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|-----------------------------|-----|-----|------|
| t77 | Data Setup (valid to high) | 7 | — | ns |
| t78 | Data Hold (high to invalid) | 0 | — | ns |

Table 99. Timing Characteristics for Serial Outputs

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|----------------------------|-----|-----|------|
| t76a | ILD Delay (high to low) | — | 35 | ns |
| t101 | ILD Hold (high to invalid) | 3 | — | ns |
| t79 | IBF Delay (high to high) | — | 35 | ns |

10 Timing Characteristics for 2.7 V Operation (continued)



* See **sioc** register, MSB field, to determine if B0 is the MSB or LSB. See **sioc** register, ILEN field, to determine if the DO word length is 8 bits or 16 bits.

Figure 31. SIO Passive Mode Output Timing Diagram

Table 100. Timing Requirements for Serial Inputs

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--|-----|----------------|------|
| t80 | Clock Period (high to high) [†] | 40 | — [‡] | ns |
| t81 | Clock Low Time (low to high) | 18 | — | ns |
| t82 | Clock High Time (high to low) | 18 | — | ns |
| t83 | Load High Setup (high to high) | 8 | — | ns |
| t84 | Load Low Setup (low to high) | 8 | — | ns |
| t85 | Load Hold (high to invalid) | 0 | — | ns |

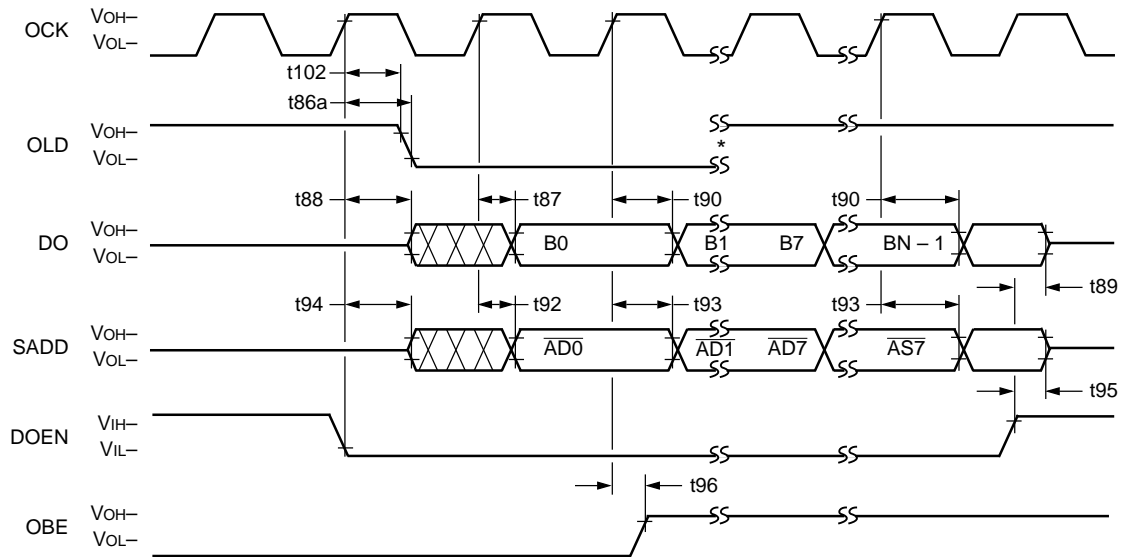
[†] For multiprocessor mode, see note in Section 10.10.

[‡] Device is fully static; t80 is tested at 200 ns.

Table 101. Timing Characteristics for Serial Outputs

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--------------------------------------|-----|-----|------|
| t87 | Data Delay (high to valid) | — | 35 | ns |
| t88 | Enable Data Delay (low to active) | — | 35 | ns |
| t89 | Disable Data Delay (high to 3-state) | — | 35 | ns |
| t90 | Data Hold (high to invalid) | 3 | — | ns |
| t92 | Address Delay (high to valid) | — | 35 | ns |
| t93 | Address Hold (high to invalid) | 3 | — | ns |
| t94 | Enable Delay (low to active) | — | 35 | ns |
| t95 | Disable Delay (high to 3-state) | — | 35 | ns |
| t96 | OBE Delay (high to high) | — | 35 | ns |

10 Timing Characteristics for 2.7 V Operation (continued)



5-4797 (F)

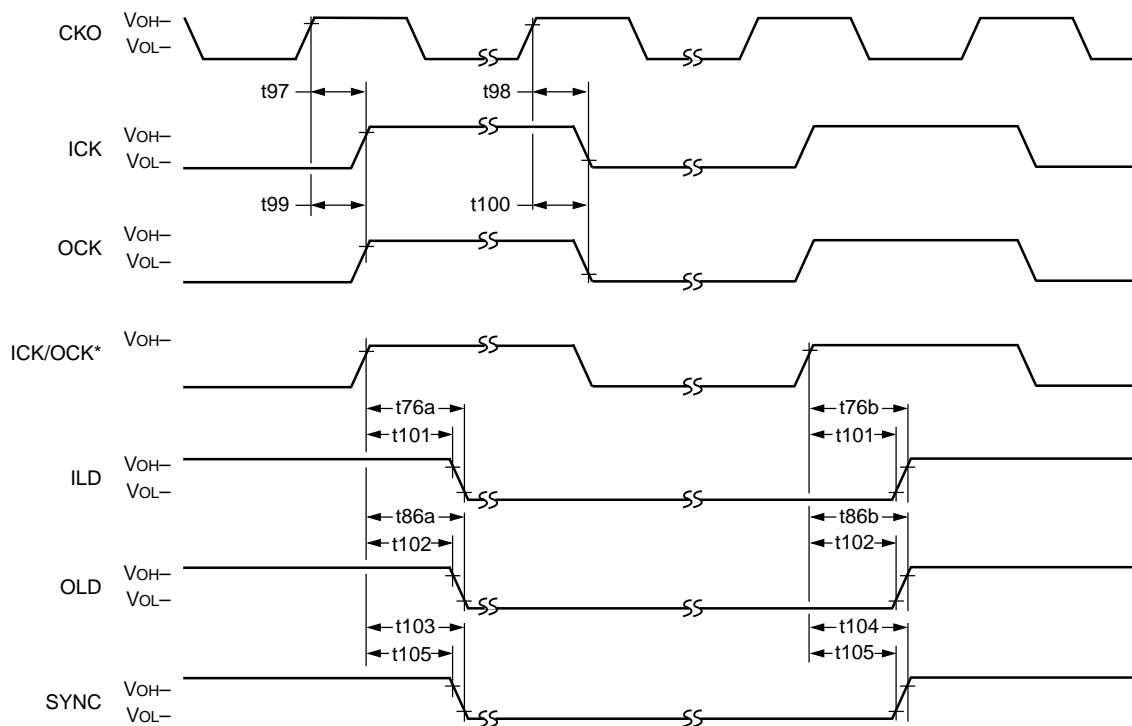
* OLD goes high at the end of bit 6 of 0:15.

Figure 32. SIO Active Mode Output Timing Diagram

Table 102. Timing Characteristics for Serial Output

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--------------------------------------|-----|-----|------|
| t86a | OLD Delay (high to low) | — | 35 | ns |
| t102 | OLD Hold (high to invalid) | 3 | — | ns |
| t87 | Data Delay (high to valid) | — | 35 | ns |
| t88 | Enable Data Delay (low to active) | — | 35 | ns |
| t89 | Disable Data Delay (high to 3-state) | — | 35 | ns |
| t90 | Data Hold (high to invalid) | 3 | — | ns |
| t92 | Address Delay (high to valid) | — | 35 | ns |
| t93 | Address Hold (high to invalid) | 3 | — | ns |
| t94 | Enable Delay (low to active) | — | 35 | ns |
| t95 | Disable Delay (high to 3-state) | — | 35 | ns |
| t96 | OBE Delay (high to high) | — | 35 | ns |

10 Timing Characteristics for 2.7 V Operation (continued)



5-4798 (F)

* See **sioc** register, LD field.

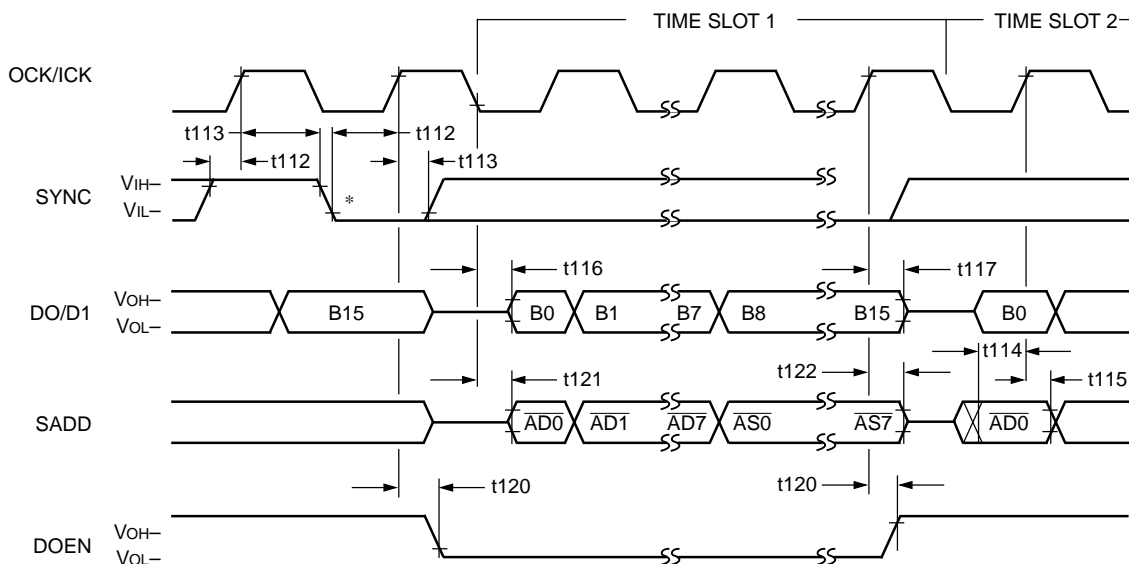
Figure 33. Serial I/O Active Mode Clock Timing

Table 103. Timing Characteristics for Signal Generation

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|-----------------------------|-----|-----|------|
| t97 | ICK Delay (high to high) | — | 18 | ns |
| t98 | ICK Delay (high to low) | — | 18 | ns |
| t99 | OCK Delay (high to high) | — | 18 | ns |
| t100 | OCK Delay (high to low) | — | 18 | ns |
| t76a | ILD Delay (high to low) | — | 35 | ns |
| t76b | ILD Delay (high to high) | — | 35 | ns |
| t101 | ILD Hold (high to invalid) | 3 | — | ns |
| t86a | OLD Delay (high to low) | — | 35 | ns |
| t86b | OLD Delay (high to high) | — | 35 | ns |
| t102 | OLD Hold (high to invalid) | 3 | — | ns |
| t103 | SYNC Delay (high to low) | — | 35 | ns |
| t104 | SYNC Delay (high to high) | — | 35 | ns |
| t105 | SYNC Hold (high to invalid) | 3 | — | ns |

10 Timing Characteristics for 2.7 V Operation (continued)

10.10 Multiprocessor Communication



* Negative edge initiates time slot 0.

5-4799 (F)

Figure 34. SIO Multiprocessor Timing Diagram

Note: All serial I/O timing requirements and characteristics still apply, but the minimum clock period in passive multiprocessor mode, assuming 50% duty cycle, is calculated as $(t_{77} + t_{116}) \times 2$.

Table 104. Timing Requirements for SIO Multiprocessor Communication

| Abbreviated Reference | Parameter | Min | Max | Unit |
|-----------------------|--------------------------------|-----|-----|------|
| t112 | Sync Setup (high/low to high) | 35 | — | ns |
| t113 | Sync Hold (high to high/low) | 0 | — | ns |
| t114 | Address Setup (valid to high) | 12 | — | ns |
| t115 | Address Hold (high to invalid) | 0 | — | ns |

Table 105. Timing Characteristics for SIO Multiprocessor Communication

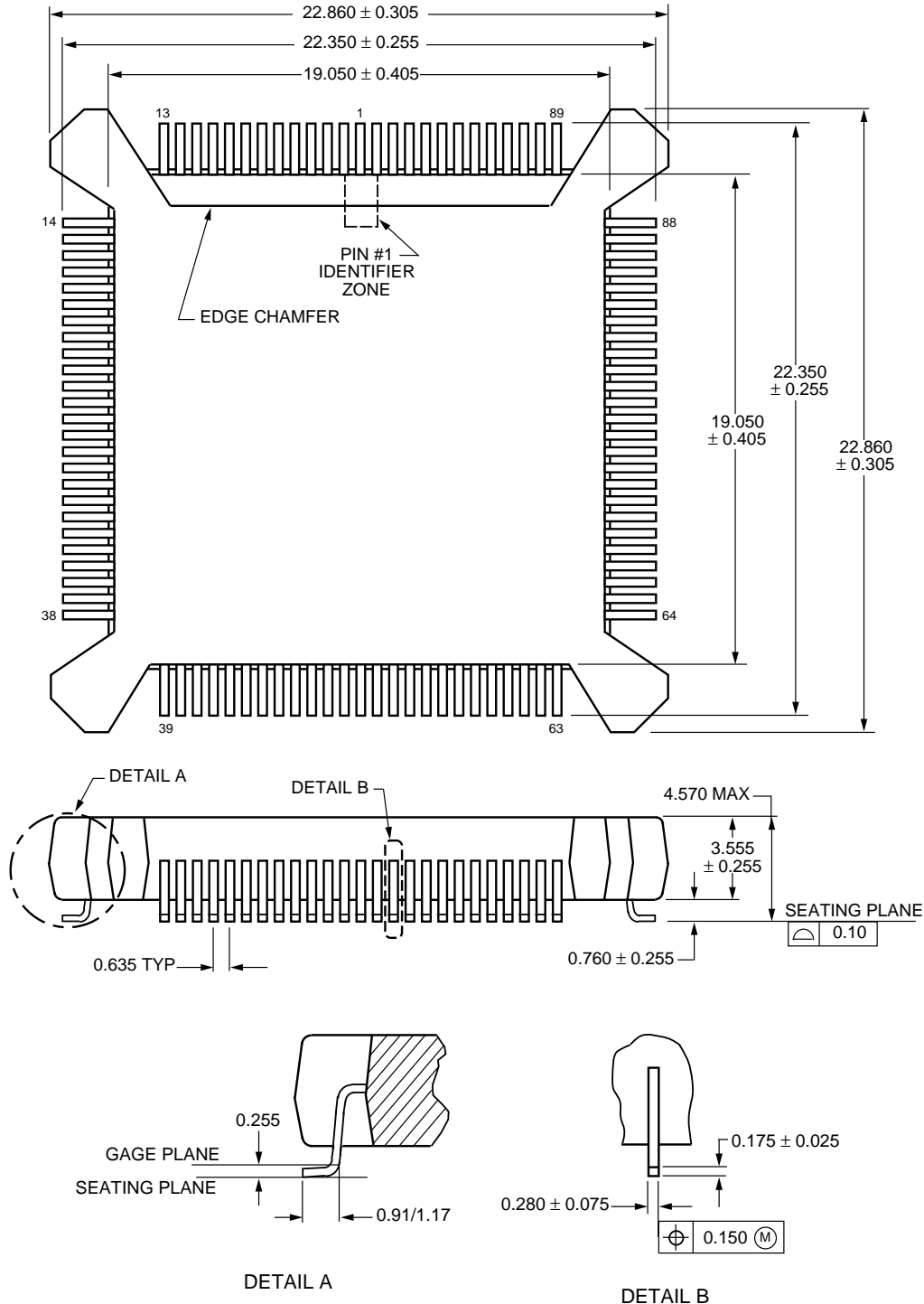
| Abbreviated Reference* | Parameter | Min | Max | Unit |
|------------------------|---|-----|-----|------|
| t116 | Data Delay (bit 0 only) (low to valid) | — | 35 | ns |
| t117 | Data Disable Delay (high to 3-state) | — | 30 | ns |
| t120 | DOEN Valid Delay (high to valid) | — | 25 | ns |
| t121 | Address Delay (bit 0 only) (low to valid) | — | 35 | ns |
| t122 | Address Disable Delay (high to 3-state) | — | 30 | ns |

* With capacitance load on ICK, OCK, DO, SYNC, and SADD = 100 pF, add 4 ns to t116—t122.

11 Outline Diagrams

11.1 100-Pin BQFP (Bumpered Quad Flat Pack)

All dimensions are in millimeters.

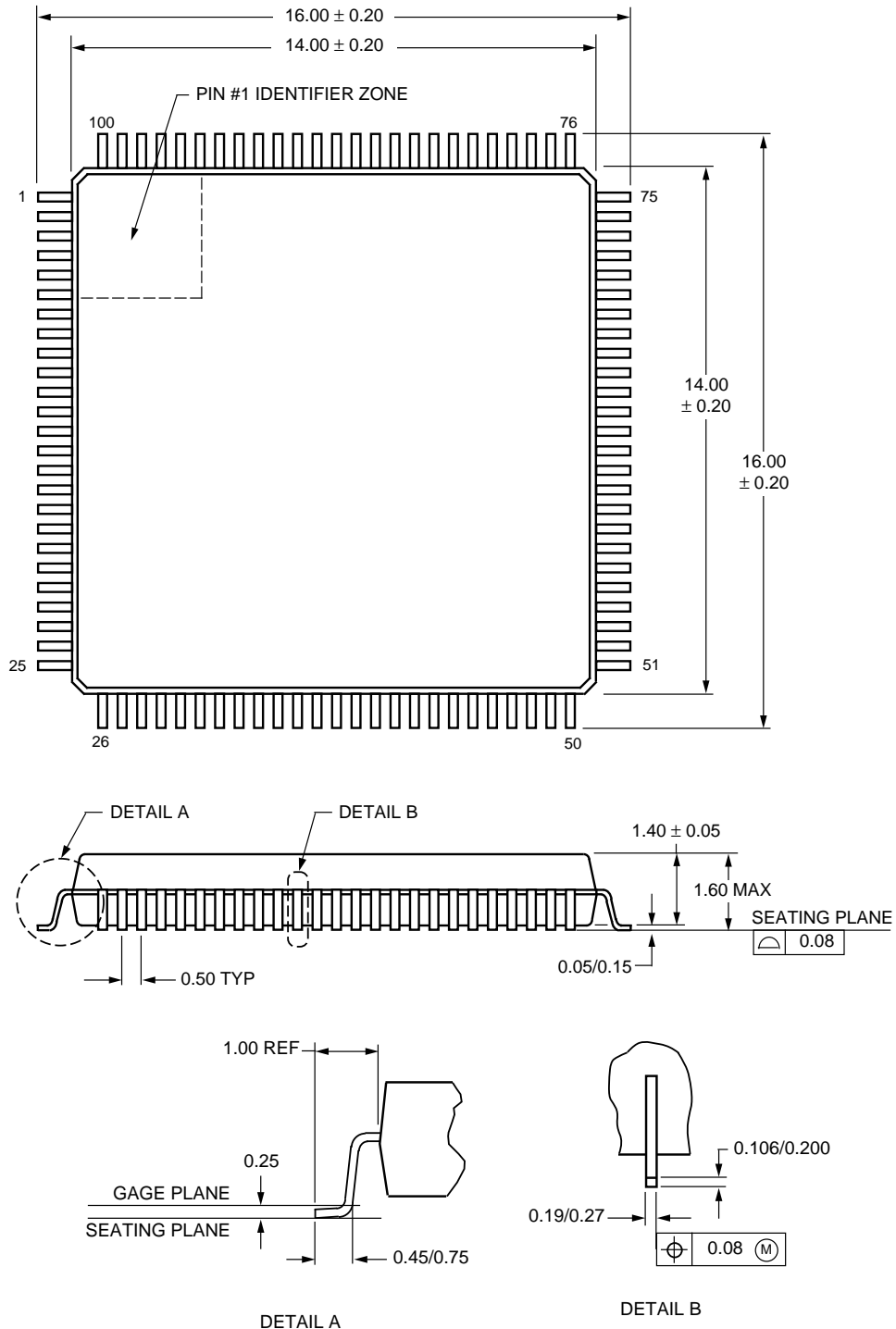


5-1970.r10

11 Outline Diagrams (continued)

11.2 100-Pin TQFP (Thin Quad Flat Pack)

All dimensions are in millimeters.

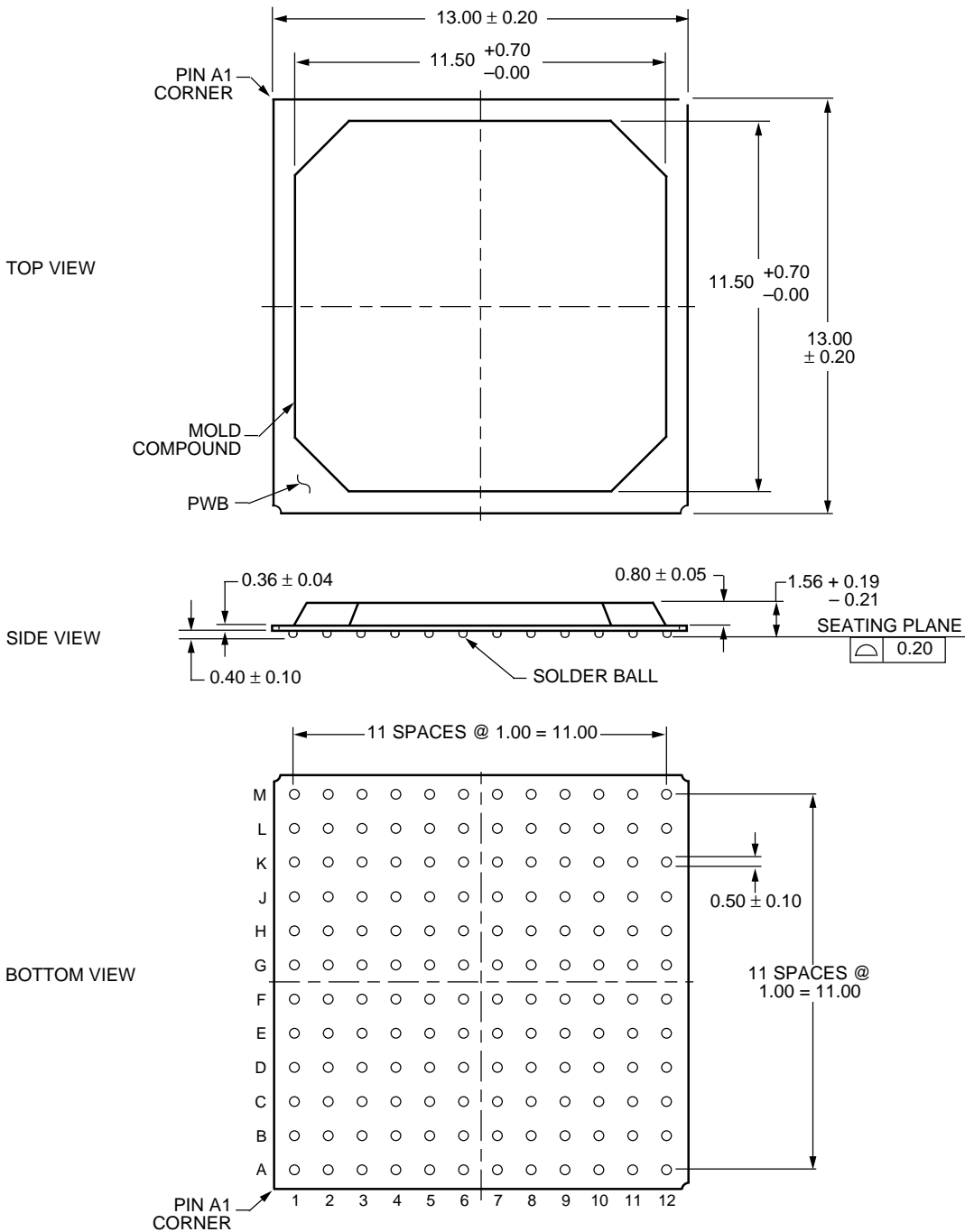


5-2146.r14

11 Outline Diagrams (continued)

11.3 144-Pin PBGA (Plastic Ball Grid Array)

All dimensions are in millimeters.



5-5205 (C)

For additional information, contact your Microelectronics Group Account Manager or the following:

INTERNET: <http://www.lucent.com/micro>

U.S.A.: Microelectronics Group, Lucent Technologies Inc., 555 Union Boulevard, Room 30L-15P-BA, Allentown, PA 18103

1-800-372-2447, FAX 610-712-4106 (In CANADA: **1-800-553-2448**, FAX 610-712-4106), e-mail docmaster@micro.lucent.com

ASIA PACIFIC: Microelectronics Group, Lucent Technologies Singapore Pte. Ltd., 77 Science Park Drive, #03-18 Cintech III, Singapore 118256

Tel. (65) 778 8833, FAX (65) 777 7495

JAPAN: Microelectronics Group, Lucent Technologies Japan Ltd., 7-18, Higashi-Gotanda 2-chome, Shinagawa-ku, Tokyo 141, Japan

Tel. (81) 3 5421 1600, FAX (81) 3 5421 1700

For data requests in Europe:

MICROELECTRONICS GROUP DATALINE: **Tel. (44) 1734 324 299**, FAX (44) 1734 328 148

For technical inquiries in Europe:

CENTRAL EUROPE: **(49) 89 95086 0** (Munich), NORTHERN EUROPE: **(44) 1344 865 900** (Bracknell UK),

FRANCE: **(33) 1 41 45 77 00** (Paris), SOUTHERN EUROPE: **(39) 2 6601 1800** (Milan) or **(34) 1 807 1700** (Madrid)

Lucent Technologies Inc. reserves the right to make changes to the product(s) or information contained herein without notice. No liability is assumed as a result of their use or application. No rights under any patent accompany the sale of any such product(s) or information.

