

**9600**

**Data Compression  
Processor**

**Data Sheet**

**Hifn**

Intelligent Secure Networking

DS-0002-01, © 2003, Hi/fn<sup>®</sup>, Inc. All rights reserved. 2/03

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means without the written permission of Hi/fn, Inc. (“Hifn”)

### **Licensing and Government Use**

Any Hifn software (“Licensed Programs”) described in this document is furnished under a license and may be used and copied only in accordance with the terms of such license and with the inclusion of this copyright notice. Distribution of this document or any copies thereof and the ability to transfer title or ownership of this document’s contents are subject to the terms of such license.

Such Licensed Programs and their documentation have been developed at private expense and no part of such Licensed Programs is in the public domain. Use, duplication, disclosure, and acquisition by the U.S. Government of such Licensed Programs is subject to the terms and definitions of their applicable license.

### **Disclaimer**

Hifn reserves the right to make changes to its products, including the contents of this document, or to discontinue any product or service without notice. Hifn advises its customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied upon is current. Every effort has been made to keep the information in this document current and accurate as of the date of this document’s publication or revision.

Hifn warrants performance of its products to the specifications applicable at the time of sale in accordance with Hifn’s standard warranty or the warranty provisions specified in any applicable license. Testing and other quality control techniques are utilized to the extent Hifn deems necessary to support such warranty. Specific testing of all parameters, with the exception of those mandated by government requirements, of each product is not necessarily performed.

Certain applications using Hifn products may involve potential risks of death, personal injury, or severe property or environmental damage (“Critical Applications”). Hifn products are not designed, intended, authorized, or warranted to be suitable for use in life saving, or life support applications, devices or systems or other critical applications. Inclusion of Hifn products in such critical applications is understood to be fully at the risk of the customer. Questions concerning potential risk applications should be directed to Hifn through a local sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards. “Typical” parameters can and do vary in different applications. All operating parameters, including “Typicals,” should be validated for each customer application by the customer’s technical experts.

Hifn does not warrant that its products are free from infringement of any patents, copyrights or other proprietary rights of third parties. In no event shall Hifn be liable for any special, incidental or consequential damages arising from infringement or alleged infringement of any patents, copyrights or other third party intellectual property rights.

The use of this product in stateful compression protocols (for example, PPP or multi-history applications) with certain configurations may require a license from Motorola. In such cases, a license agreement for the right to use Motorola patents may be obtained through Hifn or directly from Motorola.

### **Patents**

May include one or more of the following United States patents: 4,701,745; 5,003,307; 5,016,009; 5,126,739; 5,146,221; 5,414,425; 5,463,390; 5,506,580; and 5,5532,694. Other patents pending.

### **Trademarks**

Hi/fn<sup>®</sup>, MeterFlow<sup>®</sup>, MeterWorks<sup>®</sup>, and LZS<sup>®</sup>, are registered trademarks of Hi/fn, Inc. Hifn<sup>™</sup> and the Hifn logo are trademarks of Hi/fn, Inc. All other trademarks and trade names are the property of their respective holders.

### **Exporting**

This product may only be exported from the United States in accordance with applicable Export Administration Regulations. Diversion contrary to United States laws is prohibited.

# Contents

**List of Figures ..... vii**

**List of Table ..... ix**

**Preface ..... xi**

About This Document ..... xi  
Audience ..... xi  
Document Conventions ..... xi  
Customer Support ..... xi  
Web Site ..... xi

**List of Abbreviations and Acronyms ..... xiii**

**Chapter 1 Overview ..... 1**

1.1 9600 Description ..... 1  
1.2 9600 Features ..... 1  
1.3 Product Overview ..... 2  
1.4 Detailed Block Diagram ..... 4  
1.5 Signal Summary ..... 5  
1.6 Register Summary ..... 6

**Chapter 2 Operation ..... 7**

2.1 Records ..... 7  
2.2 Commands ..... 7  
2.3 Command and Record Termination ..... 8  
    2.3.1 Command Termination ..... 8  
    2.3.2 Record Termination ..... 8  
    2.3.3 Results ..... 9  
2.4 Compression ..... 10

.....

- 2.4.1 Compression, Step by Step..... 12
- 2.5 Compression History .....13
- 2.6 Decompression .....13
  - 2.6.1 Decompression, Step by Step..... 14
- 2.7 Error Handling .....15
- 2.8 CRC/LCB Calculation.....16
  - 2.8.1 CRC/LCB Algorithms..... 16
  - 2.8.2 Protected Records..... 17
- 2.9 Feedback Mode .....18
  - 2.9.1 Errors in Feedback Mode..... 19
  - 2.9.2 Feedback Mode Compression, Step by Step ..... 19
- 2.10 Data and Register Transfer .....20
  - 2.10.1 DMA ..... 20
  - 2.10.2 Programmed I/O ..... 22
  - 2.10.3 Mixing Programmed I/O and DMA transfers ..... 23
  - 2.10.4 Data Alignment Requirements ..... 23
  - 2.10.5 Requirements for Writing to the Source FIFO ..... 24
  - 2.10.6 Characteristics When Reading From the Destination FIFO .. 25
- 2.11 FIFO Data Flow .....25
  - 2.11.1 Source FIFO Data Flow ..... 25
  - 2.11.2 Destination FIFO Data Flow ..... 25
- 2.12 FIFO Thresholds.....26
  - 2.12.1 Programmed I/O and the FIFOs ..... 27
- 2.13 Clock .....28

**Chapter 3 Register Descriptions ..... 29**

- 3.1 Register Overview.....29
- 3.2 Data (0) .....30
- 3.3 Command (1).....31
  - 3.3.1 Commands ..... 32
  - 3.3.2 Command Fields..... 32
- 3.4 Result (2,10) .....33
  - 3.4.1 CRC Error..... 34
  - 3.4.2 CRC Value..... 35
  - 3.4.3 Record Count ..... 35
  - 3.4.4 Consumed Byte Count ..... 35
  - 3.4.5 Destination Count ..... 36
- 3.5 Configuration (3).....36
  - 3.5.1 Feedback Mode ..... 36

- 3.5.2 Bus Width ..... 36
- 3.5.3 Big Endian ..... 36
- 3.5.4 Reset ..... 37
- 3.6 Interrupt Enable (4,12)..... 37
  - 3.6.1 Parity Error Interrupt Enable ..... 37
  - 3.6.2 Command Ready Interrupt Enable ..... 38
  - 3.6.3 SDREQ Interrupt Enable..... 38
  - 3.6.4 DDREQ Interrupt Enable ..... 38
  - 3.6.5 Command/Result Overrun Interrupt Enable ..... 38
  - 3.6.6 Data Error Interrupt Enable ..... 38
  - 3.6.7 CRC Error Interrupt Enable ..... 38
- 3.7 Status (5,13) ..... 38
  - 3.7.1 Parity Error..... 39
  - 3.7.2 Command Ready..... 39
  - 3.7.3 SDREQ ..... 39
  - 3.7.4 DDREQ ..... 39
  - 3.7.5 Command/Result Overrun ..... 40
  - 3.7.6 Data Error..... 40
  - 3.7.7 CRC Error ..... 40
  - 3.7.8 Command In Progress ..... 40
  - 3.7.9 Result In Progress ..... 40
  - 3.7.10 Byte Enable..... 40
  - 3.7.11 EOR ..... 41
- 3.8 FIFO Configuration (7)..... 42
  - 3.8.1 U/L..... 42
  - 3.8.2 Source FIFO Threshold ..... 42
  - 3.8.3 Destination FIFO Threshold ..... 43
- 3.9 Chip ID (11)..... 43

**Chapter 4 Signal Descriptions ..... 45**

- 4.1 CPU Interface ..... 45
  - 4.1.1 System Data Bus (D[31-0])..... 45
  - 4.1.2 Parity ([3-0]) ..... 45
  - 4.1.3 Address (A[3-0]) ..... 45
  - 4.1.4 Read/Write Command (R/W#) ..... 45
  - 4.1.5 Command Strobe (CS#) ..... 46
  - 4.1.6 Interrupt (IRQ#)..... 46
- 4.2 DMA Interface..... 46
  - 4.2.1 Byte Enable (B[3-0])..... 46

- 4.2.2 DMA Ready (RDY#) ..... 46
- 4.2.3 End of Command (EOC#)..... 47
- 4.2.4 End of Record (EOR#) ..... 47
- 4.2.5 Source FIFO DMA Request (SDREQ1#)..... 47
- 4.2.6 Source DMA Acknowledge (SDACK1#)..... 47
- 4.2.7 Destination FIFO DMA Request (DDREQ1#) ..... 47
- 4.2.8 Destination DMA Acknowledge (DDACK1#) ..... 48
- 4.2.9 Command Ready..... 48
- 4.3 Miscellaneous Signals .....48
  - 4.3.1 Reset (RESET#) ..... 48
  - 4.3.2 Clock (CLK)..... 48
  - 4.3.3 DIV1, DIV0 ..... 48

**Chapter 5 Timing Descriptions ..... 49**

- 5.1 CPU Interface .....49
- 5.2 DMA Interface .....49
- 5.3 DMA Examples .....50
  - 5.3.1 Timing Diagram Terminology ..... 50
  - 5.3.2 Burst Transfer ..... 51
- 5.4 Bus Turn-Around Time .....52
- 5.5 Fixed-Length Burst Transfers .....53
  - 5.5.1 Reducing Dead Time ..... 54
- 5.6 Variable-Length Burst Transfers.....55
- 5.7 Bursting with Wait States .....56
- 5.8 Bursting with Dummy Bytes .....57
  - 5.8.1 Write Transfers..... 57
  - 5.8.2 Read Transfers ..... 58

**Chapter 6 Specifications ..... 59**

- 6.1 DC Specifications .....59
- 6.2 AC Specifications .....61
  - 6.2.1 Reset and Clock Timing ..... 61
- 6.3 Pin Description .....64
- 6.4 Package Dimensions .....67

## List of Figures

Figure 1. System block diagram .....	2
Figure 2. Internal block diagram.....	3
Figure 3. Channel architecture .....	3
Figure 4. Detailed block diagram.....	4
Figure 5. Compression example.....	11
Figure 6. Compression example, fixed-length records .....	13
Figure 7. Decompression example.....	14
Figure 8. CRC/LCB calculation in compression and decompression. ....	16
Figure 9. CRC calculation in Feedback Mode.....	18
Figure 10. Data (0) 16-bit bus mode .....	30
Figure 11. Data (0) 32-bit bus mode .....	30
Figure 12. Command (1) 32-bit bus mode.....	31
Figure 13. Command (1) 16-bit bus mode.....	31
Figure 14. Result (2,10) 32-bit bus mode.....	33
Figure 15. Result (2,10) 16-bit bus mode.....	34
Figure 16. Big Endian 32-bit bus mode.....	36
Figure 17. Big Endian 16-bit bus mode.....	37
Figure 18. Four-transfer DMA burst.....	51
Figure 19. DMA, bus turn around time.....	52
Figure 20. DMA, fixed length transfer .....	53
Figure 21. DMA, variable length transfer.....	55
Figure 22. DMA, three-transfer burst with wait states .....	56
Figure 23. DMA, four-transfer burst.....	57
Figure 24. Pinout diagram .....	66
Figure 25. 100-pin PQFP package dimensions .....	67

THIS PAGE INTENTIONALLY LEFT BLANK





## List of Table

Table 1: Signal summary .....	5
Table 2: Register summary .....	6
Table 3: Writing to the source FIFO .....	24
Table 4: Reading from the destination FIFO .....	25
Table 5: Clock multiplier selection .....	28
Table 6: Clock multiplier example .....	28
Table 7: Register List .....	29
Table 8: Commands .....	32
Table 9: Byte enable .....	41
Table 10: Absolute maximum ratings .....	59
Table 11: Recommended operating conditions .....	59
Table 12: AC specification definition .....	59
Table 13: DC electrical characteristics .....	60
Table 14: Reset and power-up timing .....	61
Table 15: CLK timing .....	61
Table 16: CPU timing .....	62
Table 17: DMA timing .....	63
Table 18: Pin Description, numerical order .....	64
Table 19: Pin Description, alphabetical order .....	65

THIS PAGE INTENTIONALLY LEFT BLANK



## Preface

### About This Document

Welcome to the 9600 Data Sheet for the Hifn 9600 Data Compression Processor. This document assumes you are already familiar with the chip technology and terminology.

### Audience

This document is for integrators and application developers responsible for and familiar with software and hardware architecture of a target system.

### Document Conventions

The following conventions may be used within this document:

- Small `Courier` typeface indicates code, **functions**, and *variables*.
- **Bold Courier** typeface indicates **items the user types**.
- *Italic* typeface indicates *file names* and *book titles*.
- **Registers** appear in **Bold** typeface.
- **SIGNAL** names appear in **BOLD CAPS**.
- Path names are written relative to the path `//filename`.

### Customer Support

For technical support about this product, please contact your local Hifn sales office, representative, distributor.

### Web Site

For general information about Hifn and Hifn products refer to: [www.hifn.com](http://www.hifn.com)

This page left intentionally blank.



## List of Abbreviations and Acronyms

Term	Definition
Command	A single instruction to the 9600.
Compressed Data Stream	A series of compressed bytes, which presumably consists of one or more compressed records.
Compressed Protected Record	A compressed record which, when decompressed, contains as its last four bytes the CRC of the preceding bytes of the record.
Compressed Record	A part of a compressed data stream ending with an embedded end marker that indicates end-of-record.
CPU	Central Processing Unit
CRC	cyclic redundancy calculation
Data Stream	Any Series of input or output bytes.
DMA	Direct Memory Access - Transfers using the 9600's request/acknowledge/ready protocol. DMA transfers send data into and out of the FIFOs; they cannot program internal registers.
Dummy Bytes	Data that is transferred but is marked invalid through the byte enables.
End Marker	A token at the end of a compressed record that indicates the end of the record. This allows the end of the record to be detected during decompression.
EOC	End-of-Command condition. This can be indicated in a variety of ways, including with the <b>EOC#</b> signal and by the use of the record count fields to specify the amount of data to process.
EOR	The end-of-record condition, which can be indicated by the <b>EOR#</b> signal or, for fixed-length records, by the use of the Source Count field in the <b>Command</b> register. On compression, an end marker is embedded into the compressed data stream to indicate the end of the record. On decompression, this end marker is detected by the 9600 automatically.
FIFO	First-In First-Out
LCB	longitudinal check byte
LZS	Lempel-Ziv-Stac compression
Padding	Data that rounds out a compressed record to an 8-bit boundary. Unlike dummy bytes, padding consists of valid data, and is a required part of the compressed data stream.
PLL	Phase-Locked Loop

<b>Term</b>	<b>Definition</b>
Programmed I/O	Transfers using the register access protocol. This method can access registers and the FIFOs.
RAM	Random Access Memory
Raw Data Stream	A series of uncompressed bytes
Raw protected record	A raw record containing, as its last four bytes, the CRC of all of the preceding bytes of the record. The CRC bytes are provided by the Host as part of the record.
Raw record	A series of uncompressed bytes with the EOR condition indicated externally.
Record	A portion of a data stream with a definite start and end. Compression and decompression operate on a record-by-record basis.
SDRAM	Synchronous Dynamic Random Access Memory



## 1 Overview

### 1.1 9600 Description

The Hifn 9600 is a lossless Data Compression Processor. It uses the industry-standard Lempel-Ziv-Stac compression (LZS®) algorithm. The 9600 can compress or decompress data at up to \*80 MB/sec. The 9600 contains a second decompress-only engine that may be used in feedback mode to optionally verify successful compression operations. The second engine is not available for general decompression operations.

The 9600 system bus interface supports both register I/O and Direct Memory Access (DMA) transfers at bus widths of 16 and 32 bits. The DMA interface supports single-cycle bus transfers at a bus clock of up to 50 MHz. The bus interface is straightforward and easy to interface to.

Hifn's LZS compression algorithm has been standardized by many organizations, including ANSI (X.3.241), QIX (122), IETF (RFC 1967, RFC 1974), TIA/EIA (655), and the Frame Relay Forum (FRF.9).

\*Note: Can be achieved by using 40 MHz Bus Clock, a clock multiple of 2, which provides 80 MHz Core Logic Clock. See [Section 2.13](#), for more details.

### 1.2 9600 Features

- Compress or decompress at 80 MB/sec
- Industry-standard LZS algorithm
- Optional feedback mode to verify compression operation
- On-chip memory eliminates the need for external compression Random Access Memory (RAM)
- Processes multiple data records per command
- High-speed, single-cycle burst interface

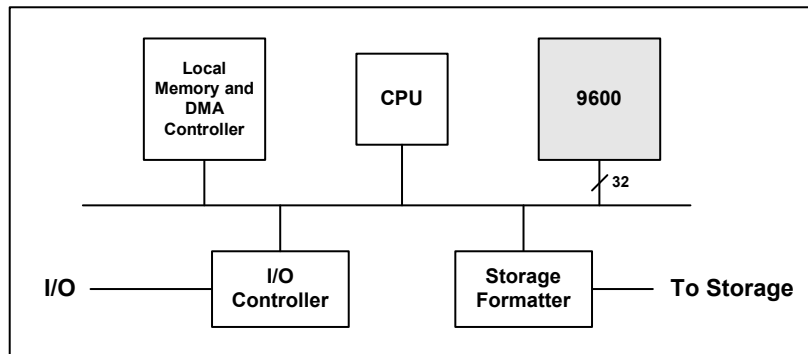


Figure 1. System block diagram

### 1.3 Product Overview

The 9600 is a dual-channel device with a single system bus interface. Each channel contains input and output First-In First-Outs (FIFOs), a compression/decompression engine, dedicated internal compression memory, command, status, and result registers. Channel 1 performs both compression and decompression; Channel 2 performs decompression only and is only used to verify the compression operation in feedback mode.

Feedback mode is a fault-detection feature that verifies the results of the compression engine in real time. In feedback mode, Channel 1 compresses the input data. Channel 2 decompresses the output of Channel 1 and verifies the resulting CRC.

Figure 2 shows an internal block diagram of the 9600. Channel 1 has a pair of 64-byte FIFOs that buffer data to and from the system data bus. These are the *Source FIFOs* and *Destination FIFOs*. These FIFOs allow Channel 1 to operate at full speed over the I/O bus. Channel 1 also has its own DMA handshaking signal to control the movement of data into and out of the FIFOs.

The 9600 system bus interface supports both register I/O and DMA transfers at bus widths of 16 and 32 bits. The DMA interface supports single-cycle bursting at a 50 MHz bus speed.



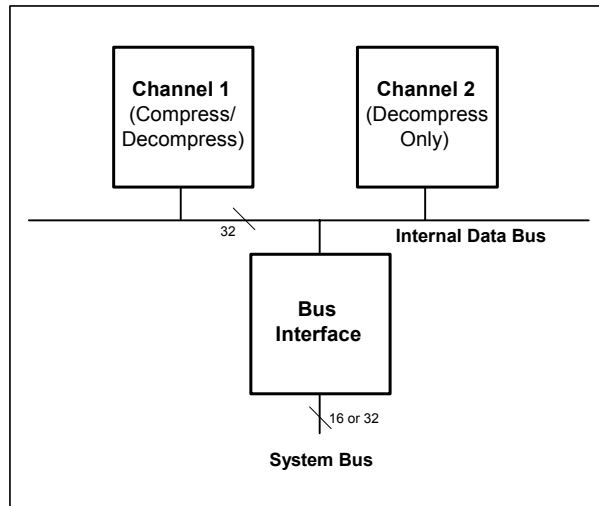


Figure 2. Internal block diagram

Figure 3 shows command and data flow in one of the Channels. Unlike data flow, command flow is not buffered by the FIFOs. The Host must wait until the previous command has completed before starting the next command.

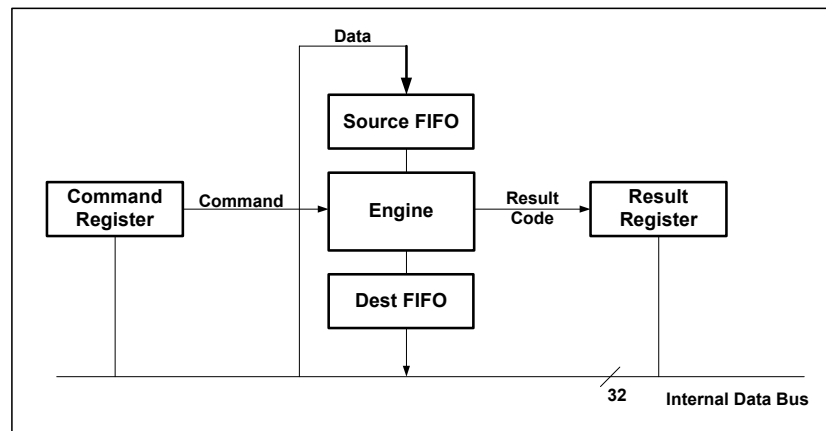


Figure 3. Channel architecture

An external DMA controller uses the 9600 to perform compression or decompression by following these steps:

1. The **Command** register is loaded with the command parameters.
2. Source data is written to the Source FIFO; output data is read from the Destination FIFO. (This step continues until all source data has been written and all output has been read.)
3. Results are checked through polling or interrupts to verify that the command completed without errors.

While the FIFOs are flushed between commands, this has little impact on performance, since a single command can process any number of data records.

## 1.4 Detailed Block Diagram

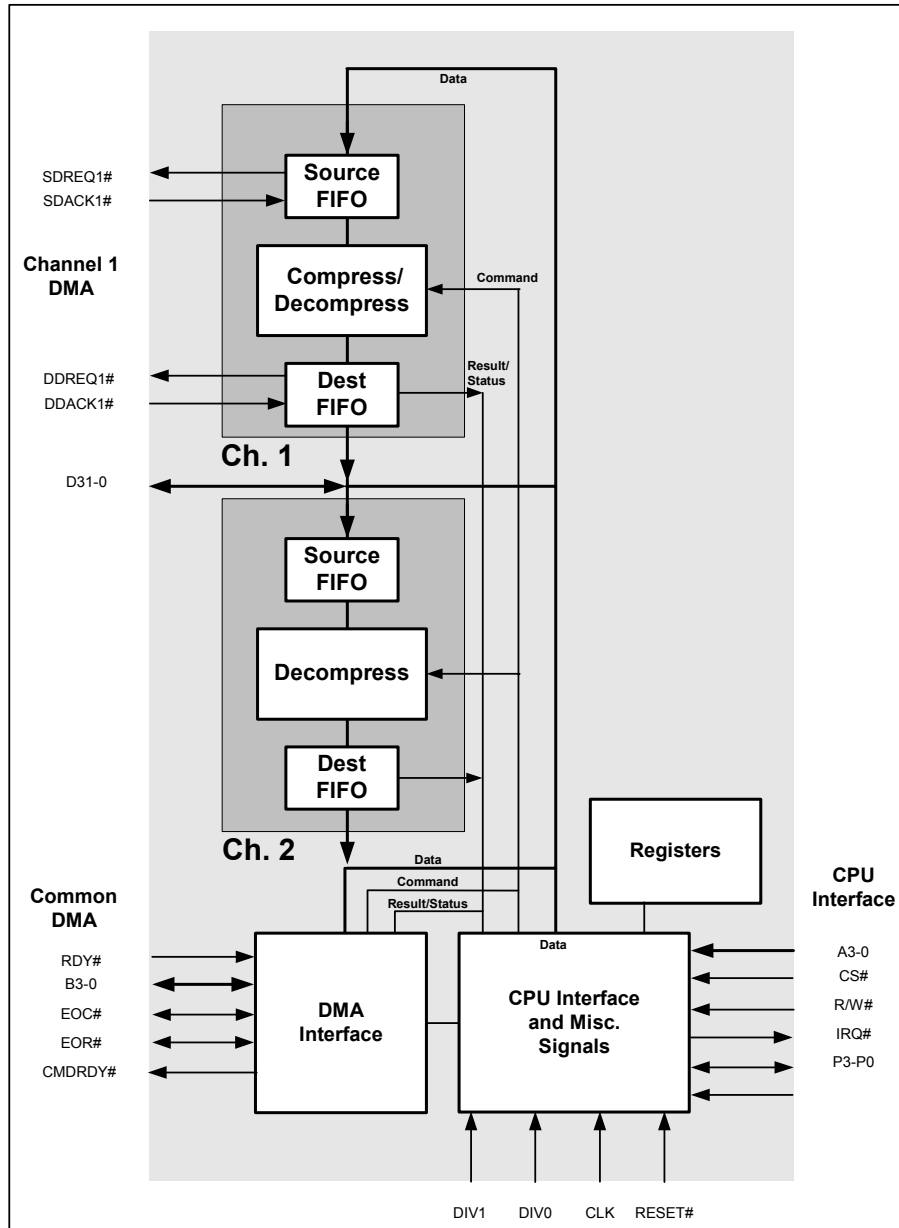


Figure 4. Detailed block diagram

## 1.5 Signal Summary

The following table summarizes the 9600 signals. For a complete signal description, see [Chapter 4](#). For a list of pins sorted by pin number, or sorted by signal name see [Chapter 6](#).

**Table 1: Signal summary**

CPU Interface	Signal	Type	Description
Common with DMA Interface	<b>D[31-0]</b>	<i>I/O</i>	32-bit System Data Bus
	<b>P[3-0]</b>	<i>I/O</i>	Parity bits
CPU/Register Interface Only	<b>A[3-0]</b>	<i>Input</i>	Register Select
	<b>R/W#</b>	<i>Input</i>	Read/Write
	<b>CS#</b>	<i>Input</i>	Command Strobe
	<b>IRQ#</b>	<i>Output</i>	Interrupt Request (open-collector)

DMA Interface	Signal	Type	Description
Common to both channels	<b>B[3-0]</b>	<i>I/O</i>	Byte Enable
	<b>RDY#</b>	<i>Input</i>	DMA Ready
	<b>EOC#</b>	<i>I/O</i>	End of Command
	<b>EOR#</b>	<i>I/O</i>	End of Record
Channel 1	<b>SDREQ1#</b>	<i>Output*</i>	Channel 1 DMA Input Request (Source FIFO)
	<b>SDACK1#</b>	<i>Input</i>	Channel 1 DMA Input Acknowledge
	<b>DDREQ1#</b>	<i>Output*</i>	Channel 1 DMA Output Request (Destination FIFO)
	<b>DDACK1#</b>	<i>Input</i>	Channel 1 DMA Output Acknowledge

	Signal	Type	Description
Miscellaneous Signals	<b>RESET#</b>	<i>Input</i>	Hardware Reset (Schmitt input)
	<b>CLK</b>	<i>Input</i>	Clock Input
	<b>DIV1, DIV0</b>	<i>Input</i>	Clock Divisor Input
	<b>VCC</b>	<i>Input</i>	+3.3 Volt supply
	<b>GND</b>	<i>Input</i>	Ground

\* External pull-up/pull-down resistors on these signals set the Phase-Lock Loop (**PLL**) clock multiple on reset. See [Section 2.13](#).

## 1.6 Register Summary

Table 2: Register summary

Name	Address		Description
	Channel 1	Channel 2	
<b>Data</b>	0	<i>Reserved</i>	32-bit data register. Writing to this register adds data to the Source FIFO. Reading removes data from the Destination FIFO.
<b>Command</b>	1	<i>Reserved</i>	3x32-bit <b>Command</b> register. Contains Command word, 32-bit Source Count, and 32-bit Record Count. Successive reads or writes access the three words in sequence.
<b>Result</b>	2	10	5x32-bit <b>Result</b> register. Contains result status word, 32-bit CRC, 32-bit Record Count, 32-bit Consumed Byte Count, and 32-bit Destination Byte Count. Successive reads or writes access the five words in sequence.
<b>Configuration</b>	3	<i>Reserved</i>	16-bit <b>Configuration</b> register. Sets device configuration, including Device Mode, Bus Width, Big-Endian, and Software Reset fields.
<b>Interrupt Enable</b>	4	12	16-bit <b>Interrupt Enable</b> register. Allows interrupts to be enabled on a variety of conditions
<b>Status</b>	5	13	16-bit <b>R/W Status</b> register. Reflects the state of each channel.
<i>Reserved</i>	6	14	<i>Reserved</i> for future expansion. Do not write to this register.
<b>FIFO Configuration</b>	7	<i>Reserved</i>	16-bit <b>FIFO Configuration</b> register. Sets the upper and lower FIFO thresholds for each channel. This controls the point at which each channel's DMA logic will request (or cease to request) data.
<b>Chip ID</b>	11	<i>Reserved</i>	16-bit read-only Chip ID.

See [Chapter 3](#) for complete register descriptions.

## 2 Operation

### 2.1 Records

The record is the fundamental unit of the compressed data stream. The number of records that can be processed by a single command is specified in the **record count** field in the **Command** register.

A data stream can consist of any series of data bytes. A record is a series of data bytes terminated by an end-of-record (EOR) indicator. During compression, the 9600 divides the raw data stream into records, then compresses the records, appends an end marker to the end of each record, and emits the records as a compressed output data stream. During decompression, the 9600 takes a compressed data stream and produces a series of raw records, reconstructing the original raw data stream.

### 2.2 Commands

A command is a single 9600 instruction. A command can process any number of records. A command is launched when a command structure is written to the **Command** register. (There is also a **Configuration** register, shared by both channels that contains state information that will not change on a command-by-command basis, such as bus width.) A command consists of three 32-bit words (or six 16-bit words) written in sequence to the **Command** register. These words are the Command word, the Source Count, and the Record Count. Writing the last word launches the command. See [Section 3.3](#) for a description of the **Command** register and its bit fields.

There are three valid commands: Compress, Decompress, and Passthrough. The Passthrough command is used mainly in diagnostics.

The concept of a command is specific to the 9600; it has no precise parallel to any feature of the compression standard. It is simply a way of processing one or more records with a single instruction.

## 2.3 Command and Record Termination

### 2.3.1 Command Termination

Commands can be terminated during compression or decompression as follows:

- By asserting the **EOC#** signal during a DMA write transfer.
- By setting the **EOC** bit in the **Status** register before a programmed I/O data write.
- When the number of records indicated by the record count field has been processed (if the **ignore record count** bit is not set).
- On a CRC error (if the CRC check is enabled).

The end-of-command (EOC) condition is associated with a specific byte of data, and flows through the channel along with that byte. Thus, when the EOC exits the Source FIFO, the Source FIFO is flushed and goes idle. No new data will be clocked into it until a new command is launched. When the EOC reaches the Engine, it updates its **Status** and **Results** registers and goes idle. When the EOC exits the Destination FIFO, the 9600 asserts the **EOC#** signal on the last output transfer containing valid data, or on a dummy transfer following the last data transfer. The Destination FIFO then goes idle. After the burst containing EOC, the Destination FIFO interface is disabled. Only the register I/O is active between commands.

Regardless of the source of the EOC, the condition is treated identically by the Channel. Whichever enabled condition is asserted first will terminate the command. For example, asserting the **EOC#** signal will terminate a command if asserted before the Record Counter reaches 0, and vice versa.

After each command, the **Result** register is updated, the FIFOs are flushed, and history is cleared.

See [Section 2.10.4](#) for details on requirements for data alignment, padding, and dummy bytes during EOC.

### 2.3.2 Record Termination

Like EOC, the EOR condition is associated with a specific byte of data, and flows through the channel along with that byte. Regardless of the source of the EOR, the condition is treated identically by the channel. Any enabled EOR condition will end a record.

#### 2.3.2.1 Compression

During compression, the raw data stream can be divided into records either externally or internally. External termination occurs when the DMA controller asserts the **EOR#** signal on the last bus transfer of each record or on a dummy transfer following the last valid transfer. The last valid byte, as determined by the

byte enables, becomes the last byte of the record. (When using the programmed I/O mechanism to transfer data, the **eor** bit in the **Status** register needs to be set to mimic the assertion of **EOR#**.)

Records can also be terminated internally, with the raw data stream divided into records of equal length before compression. This length is set by the Source Count, which is a command field.

On output, the compressed record is always padded to a 32-bit boundary. Full bytes of padding are marked as invalid by the byte enables (that is, they are transferred as dummy bytes). In 16-bit bus mode, this padding to 32-bit boundaries may cause dummy transfers.

The **EOR#** signal is asserted by the 9600 on the transfer, from the Destination FIFO, of the last byte of data in the record.

### 2.3.2.2 Decompression

On decompression, all record termination is internal, and is the result of embedded EOR tokens in the compressed data stream.

On input, the compressed data stream must consist of a multiple of four valid bytes. The 9600 assumes that padding exists after EOR and strips out the number of bytes necessary to pad the record to a multiple of four bytes. This operation is done after dummy bytes are stripped from the input stream, so if the record is padded with dummy bytes instead of valid bytes, the Channel will strip bytes from the beginning of the next record. This behavior is true only on input, and only during decompression.

On output, the Channel asserts the **EOR#** signal on a dummy transfer after the last word of the record has been transferred from the Destination FIFO.

Before each record, the CRC is initialized, the Source Counter is reset (on compression) to the maximum record length, the Record Counter is decremented, and history is reset if the **clear history command** bit is set.

### 2.3.2.3 Alignment and Padding

See [Section 2.10.4](#) for details on requirements for data alignment, padding, and dummy bytes during EOR.

## 2.3.3 Results

The **Result** register is valid at the end of each command. It is cleared at the beginning of each new command, and its contents are undefined when a command is in progress.

Each command updates the **Results** register, a structure of five 32-bit words with the following fields:

- A status word.
- The 32-bit CRC of the last record processed. The CRC is calculated on the raw (uncompressed) data stream. In other words, the CRC is calculated on the input stream during compression and the output stream during decompression. The CRC value is usually of little interest by itself, but it can be useful in debugging.
- Record Count - This starts with the Record Count given in the command, and decrements with each record. This is true even if the **ignore record count** bit is set. This allows you to determine how many records were processed, regardless of the source of EOR.
- Consumed Byte Count - This counter is cleared at the beginning of the command, then updated at each EOR with the number of valid bytes processed so far in the command (dummy bytes are not counted). If the command doesn't end on EOR, the partial record at the end of the command is not counted. This counter is used mostly in decompression, where it gives the offset of the end of the successfully decompressed portion of the compressed data stream.
- Destination Count - This value increments with each valid destination byte produced in either compression or decompression. The value is cumulative across all records in the command. It is reset to 0 at the beginning of each command.

The **Results** register is valid when the chip is idle. Attempting to read it during a command will cause Command/Result Overrun error. The value returned by such a read is undefined.

A Channel is idle when:

- The **command ready** bit in the Channel **Status** register is set, or
- **EOC#** has been asserted by the 9600 during an output transfer from the Channel Destination FIFO. In Feedback Mode, **EOC#** is used to signal that Channel 1 is idle, **CMDRDY** signals that Channel 2 is idle.

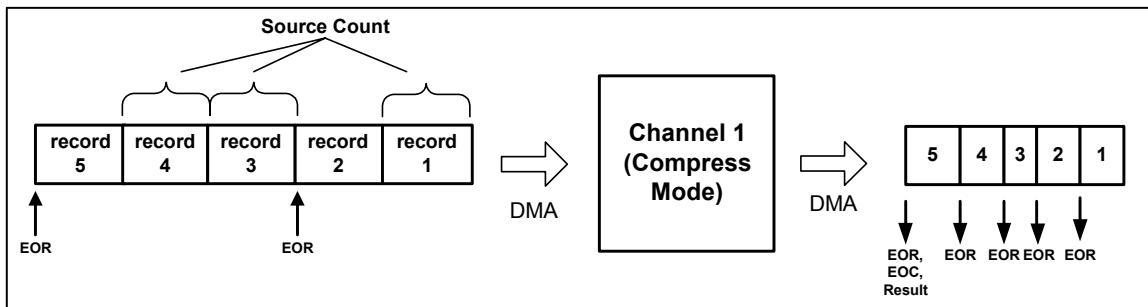
## 2.4 Compression

While both channels perform decompression, only Channel 1 can compress data. The DMA controller sends raw data to Channel 1 through the Channel 1 DMA interface. Incoming data is buffered in the Channel 1 Source FIFO, which generates a request signal (**SDREQ1#**) based on programmable thresholds. The Host DMA controller initiates transfers to Channel 1 with the **SDACK1#** signal and inserts wait states with **RDY#**. See [Section 2.11.1](#) for further details. Data can also be transferred through the programmed I/O mechanism. See [Section 2.11.2](#) for further details.



Raw data must be divided into records. This can be done internally, through the Source Count field in the command, or externally, with the **EOR#** signal. Data must also be divided into commands in a similar manner.

These external and internal methods of ending records can be mixed. [Figure 5](#) shows a command that consists of five input records. Records 1, 3, and 4 are terminated when the Source Count reaches 0. Records 2 and 5 end when **EOR#** is asserted. The channel creates five compressed records, making no distinction between records that were terminated by **EOR#** or by the Source Count. These records are then compressed to varying degrees, depending on their contents.



**Figure 5. Compression example**

On output, EOR markers are embedded in the compressed output data stream. The output data stream is buffered by the Channel 1 Destination FIFO, which generates a request signal (**DDREQ1#**) based on programmable thresholds. The Host DMA controller initiates transfers from the Channel with the **DDACK1#** signal and inserts wait states with **RDY#**. During output transfers, **EOR#** and **EOC#** become outputs. **EOR#** is asserted when the last bytes of a record are transferred. **EOC#** is asserted when the last bytes of a command are transferred across the bus. The EOR and EOC bits in the **Status** register also follow the EOR and EOC status.

Output is 32-bit aligned; the output of a new record will begin with a new bus cycle, and will end with a bus cycle padded with dummy bytes, if necessary. Output padding guarantees that the output stream will never contain bytes from two different records. The requirements for alignment and padding are listed in [Section 2.10.4](#).

The record count field decrements with each record. In the example in [Figure 6](#), after the first record is processed it will be set to 4, and it will have counted down to zero at the end of the command.

A CRC is calculated automatically on a per-record basis on the raw data. On compression, this is the input data.

## 2.4.1 Compression, Step by Step

The steps below are for the case where the **EOR#** and **EOC#** signals are used instead of the Source Count and Record Count features, as shown in [Figure 6](#). Compression is only available on Channel 1. (Details of register usage, signal definitions, and system bus timing are given in [Chapter 3](#), [Chapter 5](#), and [Section 6.2](#).)

1. Deassert the **EOR#** and **EOC#** pins.
2. Test the **command ready** bit in the **Status** register. This can be done through polling or interrupts. (This step is not strictly necessary, as the 9600 is always ready to accept a new command after the previous command's EOC.)
3. Write three 32-bit words to the **Command** register: the Command Word (0x00003000), the Source Count (0), and the Record Count (0). (In this example, the **ignore source count** and **ignore record count** bits are set in the control word, so the values written to the Source and Record Count are don't-cares. These words must be written, however, to launch the command, even though the values are ignored.)
4. Start writing data to Channel 1. The handshaking is done entirely in hardware, using the Channel 1 source DMA control signals (**SDREQ1#**, **SDACK1#**, and the common signal **RDY#**). Assert **EOR#** along with the transfer containing the last byte of each record. This does not have to occur on the last transfer of a burst.
5. Start reading data from the Channel 1 destination DMA interface (using the handshaking signals **DDREQ1#**, **DDACK1#**, and **RDY#**). This is the compressed data stream. Unlike the input stream, we don't know how long this data stream is going to be. **EOR#** and **EOC#** will be asserted by the chip as we transfer the last words of each record and of the command.
6. At the end of the input data stream, assert **EOR#** and **EOC#** along with the last data transfer. Again, this does not have to be at the end of a burst, but all data transferred after **EOC#** will be discarded.
7. Once all the output data has been transferred (**EOC#** will be asserted by the chip on the last transfer), read the **Result** register to check the error status. Reading the **Result** register is optional (interrupts can be set for all error conditions). It is also acceptable to read only part of the five-word Result structure.

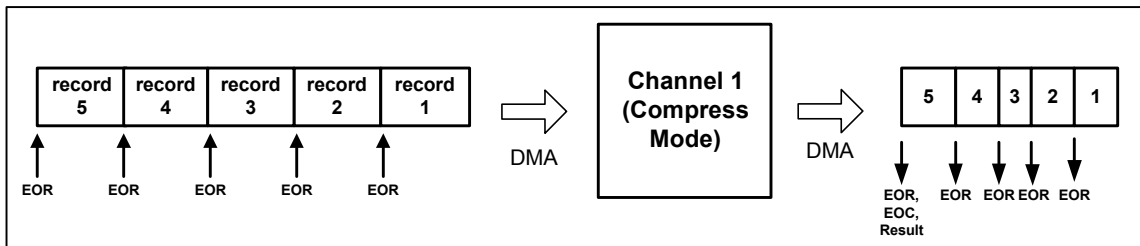


Figure 6. Compression example, fixed-length records

## 2.5 Compression History

Compression algorithms maintain data-dependent state information. This information is called the compression history. The compression history is stored in an internal compression RAM. The compression history is always cleared between commands. It can be retained across records within the same command, or cleared before each record, as determined by the **clear history** bit in the **Command** register.

## 2.6 Decompression

Compressed data is presented to the Channel as an undifferentiated data stream. As the data is decompressed, the record boundaries embedded in the data stream are detected. As in compression, the Command includes the Source Count and Record Count. For example, setting the Record Count to five will terminate the command after five records have been decompressed. The command will also terminate if the number of input bytes specified in Source Count has been consumed.

DMA data flow is almost identical to the case of compression, though in decompression **EOR#** will be ignored if asserted along with the source data stream. In the destination data stream, the chip asserts **EOR#** or **EOC#** in a dummy transfer at the end of each decompressed record or command. The start of each output record is aligned to a 32-bit boundary. Dummy bytes are inserted between output records to achieve this alignment.

See [Section 2.10.4](#) for details of alignment and padding requirements for decompression.

### 2.6.1 Decompression, Step by Step

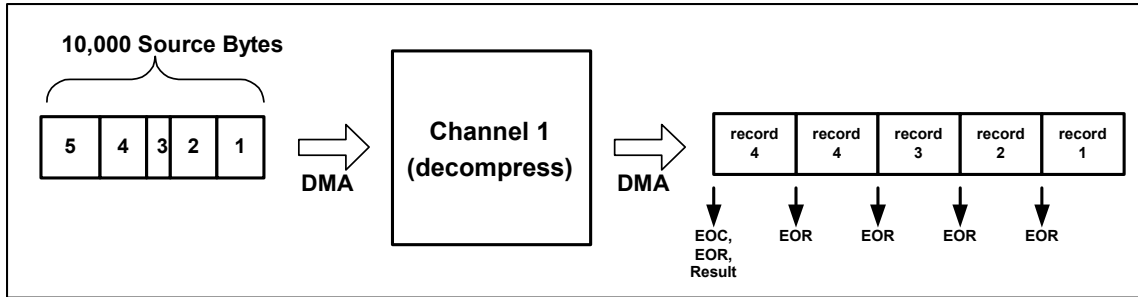


Figure 7. Decompression example

The example in [Figure 7](#) assumes that we have chosen to decompress a block of 10,000 compressed bytes.

1. Deassert the **EOC#** pin.
2. Wait, through interrupts or polling, until **Status** register bit 14 (**command ready**) is set. (As in compression, this is not strictly necessary.)
3. Write three 32-bit words to the **Command** register: the Command (0x10001000), the Source Count (10,000), and the Record Count (0). (With the **ignore word count** bit set in the Command Word, the Record Count value will be ignored, but a write must still take place.)
4. Start writing data to Channel 1. This is the same as the previous example, except that **EOR#** should not be asserted.
5. Start reading data from Channel 1. This is the decompressed data stream. As in example 1, we don't know how long it will be. **EOR#** or **EOC#** will be asserted by the chip as we transfer the data at the ends of records or commands.
6. Stop sending data after you've transferred your 10,000 bytes. Alternatively, you can assert **EOC#** if you want to stop before 10,000 bytes have been transferred.
7. Wait until the last byte of decompressed data has been transferred from the channel (**EOC#** will be asserted after the last transfer of valid data.)
8. Read the **Result** register, which contains a data structure of five 32-bit words. Reading the Result is optional. Reading only part of the five-word Result structure is also acceptable.
9. Note that the command ended in the middle of record 5. The CRC in the **Result** register will be that of record 4, not for the incomplete record 5. The consumed byte count will also reflect the value at the EOR 4. If we wished to decompress record 5, the consumed byte count field would give the offset of the first byte of record 5.

## 2.7 Error Handling

The **Status** register contains error bits for command/result overrun, data error, and CRC error. These error bits are set under the following conditions:

**Command/Result Overrun:** The **Command** register was written or the **Result** register was read when there was a command in progress.

**Data Error:** The Source FIFO was overflowed or the Destination FIFO was underflowed.

**CRC Error:** A raw protected record with an invalid CRC was detected. The **crc enable** bit must be set for this test to take place.

**Parity Error:** A parity error was detected on the **P[3:0]** bus during a write to the Source FIFO.

When a CRC error is detected, the current command is terminated. The other errors simply cause the associated bits in the **Status** register to be set, and, if the associated **interrupt enable** bit is set, the **IRQ#** signal to be asserted.

These error bits can only be cleared by a hardware or software reset of the chip. A software reset is performed by writing a 1 to the **reset** bit of the **Configuration** register.

## 2.8 CRC/LCB Calculation

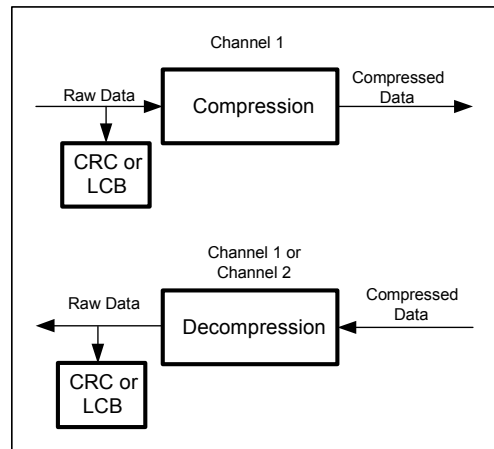


Figure 8. CRC/LCB calculation in compression and decompression.

In [Figure 8](#), the 9600 calculates the CRC (cyclic redundancy calculation) or LCB (longitudinal check byte) of the raw (uncompressed) data stream automatically. In compression, it calculates the CRC or LCB of the input data. In decompression, it calculates the CRC or LCB of the output data. These calculations are done on a per-record basis. The CRC field in the **Result** register is the CRC or LCB of the last complete record. The CRC field is not updated on partial records.

### 2.8.1 CRC/LCB Algorithms

In all three algorithms, the initial value is all ones (0xFFFFFFFF for a 32-bit CRC, 0xFFFF for a 16-bit CRC, and 0xFF for an 8-bit LCB).

The CRC can be either 16 or 32 bits in length. The LCB is eight bits long. The 32-bit CRC equation is:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1,$$

where x is the current data byte. The 16-bit CRC equation is:

$$x^{16} + x^{12} + x^5 + 1$$

The LCB is the exclusive-OR sum of each byte in the raw record. That is, with each data byte, the data is XORed with the partial LCB, and the result becomes the new LCB.

In addition, both CRC32 and CRC16 should be logically bit-wise inverted:

$$\text{CRC32} = \sim(x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1);$$

$$\text{CRC16} = \sim(x^{16} + x^{12} + x^5 + 1);$$

for a record to pass through the 9600 correctly (i.e., No CRC error).

## 2.8.2 Protected Records

The 9600 may be configured to test the integrity of each record being processed. To function in this way, the data being processed must be formatted as protected records. A protected record is a record with a CRC-32 appended to the end.

Before records may be compressed, the system must append a CRC-32 to the end of each record. During compression, the 9600 will verify the integrity of the protected record by scanning the entire record, including the appended CRC-32. If the CRC-32 at the end of a protected record is valid, device operation will continue. If the CRC-32 is invalid, the device will terminate at the end of the record. The entire protected record (including the appended CRC-32) will be compressed by the device and becomes part of the compressed record.

During decompression, the protected record will be decompressed, resulting in the original uncompressed protected record (with the original appended CRC-32). The entire record, including the CRC-32 is scanned for integrity after decompression. If the CRC at the end of a protected record is valid, device operation will continue. If the CRC-32 is invalid, the device will terminate at the end of the record.

As described in the CRC/LCB Calculation section, the CRC-32 will always be calculated over the uncompressed data in a record. However, testing the integrity of protected records may be configured by the **crc enable** bit in the Command Word. The result of the integrity test is part of the **Result** and **Status** registers.

The 9600 updates the CRC Error flag only at the end of the record. A CRC error will not be flagged if a command terminates before the end of a record is processed. This prevents spurious CRC errors from being reported every time a command terminates in the middle of a record.

When a CRC error is detected at the end of a record, the command will terminate. A reset must be issued to clear the CRC Error flag.

### 2.8.2.1 Notes on Protected Records

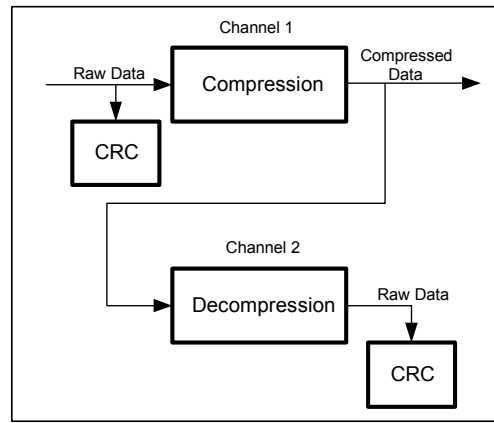
If the CRC-32 is calculated over an entire protected record (including the embedded CRC-32), the result will always be 0xDEBB20E3. This information may be useful when designing the external logic used to scan protected records decompressed by the 9600.

The CRC embedded in a protected record is part of the record. It is compressed and decompressed along with the rest of the record, and the 9600 treats it like any other data. The CRC is embedded by the Host; the 9600 neither adds it to the data

stream nor removes it from the data stream. If the **crc enable** bit is set in the Command Word, four CRC bytes must be appended to the end of every raw record, or a CRC Error will be generated.

For example, if a 256-byte record has a 32-bit CRC of 0x1A9BFE29, appending the 32-bit constant 0x1A9BFE29 to the end of the record will create a new 260-byte protected record. When this record is processed by the 9600, it will return a CRC value of 0xDEBB20E3. This value is constant for every protected record with valid 32-bit CRC bytes.

### 2.9 Feedback Mode



**Figure 9. CRC calculation in Feedback Mode**

The 9600 can operate with Feedback Mode enabled or disabled. In Feedback Mode, Channel 1 is used for both compression and decompression, while Channel 2 is used only to verify proper operation of the device in compression. The figure above shows feedback-mode operation.

Feedback Mode is enabled by setting the **feedback mode** bit in the **Configuration** register.

In Feedback Mode, Channel 1 is used for both compression and decompression; Channel 2 is dedicated to per-record CRC checking during compression. Channel 1 compresses records, while Channel 2 decompresses them to check their CRCs. Provided that the input data consists of valid raw protected records, this error will only occur if there has been some kind of hardware failure.

When the device is put into Feedback Mode, Channel 2 registers are programmed automatically for proper feedback operation. The only Channel 2 registers which can be accessed in Feedback mode are the **Result** register, the **Interrupt Enable** register, and the **Status** register. Writes to the other registers will be ignored; reads are undefined.



Feedback Mode ties the Destination FIFO of Channel 1 to the Source FIFO of Channel 2. Since both channels operate at the same speed, the bandwidth is the same whether compression uses Feedback Mode or not, but Feedback Mode adds a small amount of latency.

The **CMDRDY** pin is asserted when the Channel 2 command is complete, informing the Host system that the Channel 2 **Result** register is ready and that a new command can be started.

Checking the result requires that the Host read both **Results** registers. The Channel 2 result will not become ready at the same time as the Channel 1 result. For Channel 1, EOC takes place as usual, with the **EOC#** pin being asserted on the last valid data transfer from the Destination FIFO. Channel 2 has no opportunity to assert **EOC#**, as there are no data transfers from Channel 2. Channel 2 does, however, assert the **CMDRDY** signal at the end of the command. The 9600 is not ready for a new command until the **CMDRDY** signal has been asserted.

### 2.9.1 Errors in Feedback Mode

If the 9600 is in Feedback Mode with the CRC test enabled, any record that fails the CRC test in either Channel will cause the command to abort. The CRC error status will be given in the **Result** register for each channel. A Channel 1 CRC failure indicates a data corruption problem. A Channel 2 CRC failure indicates a device failure.

The CRC error is only asserted on EOR. If a command ends partway through a record, no error is generated.

When the CRC test is enabled, both Channels check the CRC with every record, in both compression and decompression. The **crc error** bit is not only in the **Result** register but is also replicated in the **Status** register. The **Interrupt Enable** register determines whether a CRC error will cause an interrupt.

If a CRC error occurs in Channel 1, the command will terminate in Channel 1, but Channel 2 will keep going until any valid data produced by Channel 1 before the error has been processed. In this case, **EOC#** will be asserted first, followed by **CMDRDY** some number of cycles later.

If a CRC error occurs in Channel 2, the command is terminated immediately in both channels. In this case, **CMDRDY** may be asserted alone, and **EOC#** will not be asserted at all. The Host DMA state machine needs to take this into account.

If a CRC error occurs in both Channels on the same cycle, both **EOC#** and **CMDRDY** may be asserted at the same time.

### 2.9.2 Feedback Mode Compression, Step by Step

This example gives the procedure for compression in Feedback Mode.

1. Enable Feedback Mode by setting the **feedback mode** bit in the **Configuration** register.

2. If desired, set the interrupt enable for the CRC Error in the Channel 2 **Interrupt Enable** register. This will catch device failures. This step is not absolutely necessary, since CRC errors will cause the command to terminate, and the **crc error** bit will be set in the **Status** and **Result** registers.
3. If desired, also set the interrupt enable for the CRC Error in the Channel 1 **Interrupt Enable** register. This will detect data corruption errors on the raw protected records. Again, the interrupt is not necessary to catch CRC errors, because the command will terminate on CRC errors.
4. Compress the data as usual.
5. When the last byte of output data has been transferred from the Channel 1 Destination FIFO, **EOC#** will be asserted as usual, unless there has been an error in Channel 2. The Channel 1 result can be checked after EOC if desired.
6. When Channel 2 has reached the end of the command, it will assert **CMDRDY**. This marks the true end of the command.
7. Test Channel 2 for errors as usual, through programmed I/O or interrupts. The CRC Error has special significance in Feedback Mode, of course, and should always be checked, or there is no point to operating the chip in Feedback Mode.
8. Begin a new command. The 9600 will be ready for a new command when **CMDRDY** is asserted.

## 2.10 Data and Register Transfer

Once a command has started, data can be transferred to the Channel. Input data is written to the Source FIFO. Output is read from the Destination FIFO. There are two mechanisms for transferring data: programmed I/O and DMA. DMA provides higher performance.

Both methods use FIFOs in an identical manner. Since the programmed I/O method does not use the EOR, EOC, or byte-enable signals, there are a set of **read/write status** register bits that provide the same function. Writing these bits has the same effect as asserting the signals; reading them gives the same information as sampling the signals.

DMA and programmed I/O data transfers should not be mixed in the same command.

The examples in this document will all use DMA unless it is explicitly stated that programmed I/O is being used.

### 2.10.1 DMA

The most efficient way to use the 9600 is through its DMA interface. The DMA interface allows data to move directly between the Host DMA controller and the 9600 FIFOs. Channel 1 has two FIFOs: a Source FIFO and a Destination FIFO.

Each FIFO has its own request and acknowledge signals. Thus, Channel 1 has **SDREQ1#**, **SDACK1#**, **DDREQ1#**, and **DDACK1#**. The request signals are outputs; the acknowledge signals are inputs.

In addition to the FIFO-specific signals, there are a number of shared signals: **RDY#** input and the bidirectional **EOC#**, **EOR#**, **D [31:0]** data bus, and **B[3:0]** byte enables.

Once a command is issued, DMA transfers can begin. The Source FIFO is always empty at the beginning of a command, so the **SDREQ1#** signal will be asserted immediately. Data to be processed is written by the Host to the Source FIFO. The Host initiates a burst by asserting the **SDACK#** signal.

The **SDREQ1#** and **DDREQ1#** signals are controlled by programmable FIFO thresholds. These thresholds determine the points at which the signals are asserted and deasserted, based on how full or empty the FIFOs are. FIFO thresholds are discussed in [Section 2.12](#).

The EOR and EOC conditions are indicated by the **EOR#** and **EOC#** signals. **EOR#** and **EOC#** are transferred with the data, or immediately after it as part of a dummy transfer. When the Host is writing data, it writes **EOR#** and **EOC#** as part of the last data write. When the Host reads data, it reads **EOR#** and **EOC#** from the 9600.

Data sent to the Channel after EOR is considered to be part of the next record (see the discussion of data alignment requirements in [Section 2.10.4](#)).

Any data sent to the Channel after EOC is ignored. Data read from the channel after EOC is invalid (these statements are also true of programmed I/O). In other words, the Host can continue to send data after EOC, and there will be no adverse effects. This is useful, for example, in decompression, where the Host often has no idea of the exact position of the end of a compressed record. The Host can send a too-large block of data, telling the 9600 to decompress the correct number of records. Leftover data at the end will be discarded automatically. The consumed byte count field in the **Result** register will report the number of valid bytes processed during the command (See [Section 3.4.4](#) for details of the consumed byte count.)

DMA timing is covered in detail in [Section 5.2](#).

### 2.10.1.1 Byte-Enable Signals

The byte-enable signals, **BE[3:0]**, are written at the same time as data on the data bus. They indicate which bytes in the data word are valid. When writing data to the Source FIFO, the byte-enables can be asserted in any pattern, and the 9600 will discard invalid (dummy) bytes and process those marked valid. When reading from the Destination FIFO, the Host should discard any bytes for which the byte-enables are not asserted. The Host DMA logic must monitor the byte-enables on every read transfer from the 9600.

To indicate the end of a command or record without writing data, a dummy word may be written to the Source FIFO. When this dummy word is written, the **EOC#** or **EOR#** signals are asserted, but the byte-enables are not. This signals the EOR or EOC status without transferring data.

Dummy bytes and dummy transfers to the Source FIFO occupy space in the FIFO as if they were valid data. They are not discarded until they exit the FIFO. Thus, the design of the Host DMA controller must count dummy bytes as if they were valid for purposes of setting Source FIFO thresholds.

When using programmed I/O, the byte-enable bits in the **Status** register must be used to indicate which bytes are valid. This is described further in [Section 2.12.1](#).

When reading from the Destination FIFO, the byte-enable signals are written by the 9600 along with the data. The first transfer of every record is always aligned to the bus boundary.

In some situations, there may be a destination transfer with no valid bytes to mark the EOC, record or both. This would be indicated by a transfer which has no valid data bytes, but for which the **EOC#** or **EOR#** signals are asserted. Since the word has no valid data, the **BE[3:0]** signals (or the **BE** bits in the **Status** register for programmed I/O) would be deasserted.

See [Section 5.2](#) for a detailed description of the DMA interface protocol.

### 2.10.2 Programmed I/O

Programmed I/O is a conventional interface using the **CS#** and **R/W#** signals for control, the **A[3:0]** bus for register addressing, and the **D[31:0]** bus for data transfers. See [Section 2.10.2](#) for a detailed description of the programmed I/O bus protocol.

Programmed I/O is the only mechanism for reading and writing registers, including the **Command** register. The chip registers are inaccessible to the DMA interface. Thus, chip configuration, commands, results, and status are all transferred exclusively through programmed I/O.

The basic operation of programmed I/O consists of reading and writing registers. Most registers are either 16 or 32 bits long. The **Command** and **Results** structures contain multiple 32-bit words, which are read or written sequentially through consecutive accesses. For example, the **Command** structure consists of three 32-bit words (or six 16-bit words in 16-bit bus mode). In 32-bit bus mode, a command is launched by writing three words to the **Command** register: first the Command Word, then the Source Count, then the Record Count. The command is launched when the third word is written.

The **Result** structure is five 32-bit words long (or ten 16-bit words long in 16-bit bus mode), and is read through five (or ten) reads to the **Result** register. The **Command** can also be read back.

Programmed I/O can be used as an alternative to DMA for transferring data into and out of the 9600. Writes to the **Data** register send data into the Source FIFO. Reads from the **Data** register return data from the Destination FIFO.

Because crucial state information is carried by the DMA control signals, these signals must be emulated when using programmed I/O in place of DMA. The signals that must be emulated are: **B[3-0]**, **DDREQ1#**, **EOC#**, **EOR#**, and **SDREQ1#**. Bits to emulate these signals exist in the **Status** register. These signals are used to indicate valid bytes, EORs and commands, and to provide FIFO handshaking. On writes, the **Status** register is first written with the correct handshaking bits, then the **Data** register is written. However, if the transfer matches the chip default conditions (all bytes valid, no EOC, no EOR), the **Status** register does not have to be written. On reads, the **Status** register is read, then the **Data** register is read. The **Status** register must be examined before every data read to note the status of the 9600. Otherwise, errors, EOC and EOR status, and FIFO flags will be ignored.

See [Section 2.12.1](#) for an example of programmed I/O transfers.

### 2.10.3 Mixing Programmed I/O and DMA transfers

The input and output data streams should be provided either by programmed I/O or by DMA. The two should not be mixed in the same command.

Other than optionally providing the input and output data stream, the only programmed I/O that should take place during a command is the reading of the **Status** register. All other register access should take place only when the chip is idle (after reset or EOC).

### 2.10.4 Data Alignment Requirements

The 9600 has different alignment requirements depending on what operation is being performed. When using the **EOR#** and **EOC#** signals to indicate EOR and EOC, there is the problem of determining which of the bytes in the transfer is the last byte. This issue can be solved either requiring a particular data alignment (such as requiring that every record be 32-bit aligned) or by using the byte-enables to mark bytes past the EOR or EOC as invalid. On output the byte-enables are used to indicate the last byte. On input, there are cases where the system cannot mark the last input byte in the transfer:

- During compression with fixed-length records, where the 9600, not the Host, is dividing the raw data stream into records using the source count field, and
- During decompression, where the Host does not know exactly where the EORs are in the data stream.

The chip handles all these cases. The following two tables show the alignment and padding requirements for different circumstances:

## 2.10.5 Requirements for Writing to the Source FIFO

Table 3: Writing to the source FIFO

<b>Start of Record Alignment</b>	Compression and Decompression	Data can have any alignment when sent to the Source FIFO. Dummy bytes (that is, bytes marked invalid by the byte-enable signals) can be sent at any point in the record. (But see 'Decompression' under 'EOR Alignment,' below.)
<b>EOR Alignment</b>	Compression	If the <b>EOR#</b> input signal is used, the last transfer of a record must not contain bytes from two different records. Thus, the last transfer should be padded with dummy bytes, if necessary, to round out the transfer to the full bus width. If the <b>EOR#</b> input signal is not used (that is, if the source count field is used to divide data into records), the data stream can be continuous: the last byte of one record and the first byte of the next record can be in the same bus transfer.
	Decompression	The number of valid bytes in a record must be a multiple of four bytes. To enforce this, the Source FIFO discards 0-3 bytes after EOR. This happens after dummy bytes are stripped from the input. Thus, the end of the record must be padded with 0-3 bytes of data after EOR. These bytes must be valid bytes. In practice, this is generally achieved by sending 32-bit-aligned records to the 9600 and not using the byte enables at all when writing to the Source FIFO. The <b>EOR#</b> signal is ignored on input during decompression.
<b>Start of Command Alignment</b>	Compression and Decompression	No restrictions.
<b>EOC Alignment</b>	Compression and Decompression	No restrictions. Data after EOC is discarded.
<b>EOR/EOC Location</b>	Compression and Decompression	<b>EOR#</b> and <b>EOC#</b> can be asserted by the Host during either the write containing the last byte of the record/command, or to a dummy transfer that follows the write of the last valid data transfer.

## 2.10.6 Characteristics When Reading From the Destination FIFO

The 9600 never produces dummy bytes except at the EOR or command. All bytes between the start of a record and EOR are valid.

**Table 4: Reading from the destination FIFO**

<b>Start of Record Characteristics</b>	Compression and Decompression	Records are 32-bit aligned on output. They always start on a new bus transfer. Thus, a transfer never contains bytes from two records.
<b>EOR Characteristics:</b>	Compression	To achieve 32-bit alignment on output, dummy bytes are added after the last byte of a record to round it out to a 32-bit boundary. If the Host does not conclude the burst immediately, dummy transfers will be sent for the rest of the burst.
<b>Start of Command Characteristics</b>	Compression and Decompression	Commands are 32-bit aligned on output. They always start on a new bus transfer.
<b>EOC Characteristics</b>	Compression and Decompression	To achieve 32-bit alignment on output, dummy bytes are added after the last byte of a command to round it out to a 32-bit boundary. If the Host does not conclude the burst immediately, dummy transfers will be sent for the rest of the burst.
<b>EOR/EOC Location</b>	Compression	<b>EOR#</b> and <b>EOC#</b> are attached to the transfer containing the last valid data byte
	Decompression	<b>EOR#</b> and <b>EOC#</b> are part of a dummy transfer that occurs after the chip writes that last valid data.

## 2.11 FIFO Data Flow

### 2.11.1 Source FIFO Data Flow

Each Channel has its own Source FIFO. The Source FIFO obtains source data from the bus interface using either DMA or programmed I/O transfers.

The Source FIFO requests data if there is an active command and there is room in the Source FIFO for additional data. The Source FIFO will stop requesting data when there is no more room, or when the command terminates. The thresholds at which the FIFO starts and stops requesting data are programmable.

All bytes in the Source FIFO are counted when determining thresholds, even dummy bytes (marked as invalid by the byte-enable signals). For example, three 32-bit data transfers count as 12 bytes, regardless of how many of the bytes were valid.

### 2.11.2 Destination FIFO Data Flow

Each channel also has its own Destination FIFO. The Destination FIFO delivers destination data to the bus interface using either DMA or programmed I/O transfers.

The Destination FIFO requests output when there is enough data available. The request will remain active until there is too little data remaining. These thresholds are programmable.

After the last byte of data from a record (or command) has entered the Destination FIFO, the Destination FIFO requests data transfers, regardless of the FIFO thresholds, until the last byte of the record (or command) exits the FIFO.

If the **EOR#** or **EOC#** signal is asserted in the middle of a data burst, the burst will continue with dummy bytes until the burst is concluded.

The **EOR#** and **EOC#** signals are independent of one another. If **EOC#** is asserted without **EOR#**, the last record was incomplete.

### 2.12 FIFO Thresholds

There are four FIFO Threshold values for each engine: Source FIFO Upper Threshold, Source FIFO Lower Threshold, Destination FIFO Upper Threshold, and Destination FIFO Lower Threshold. All four of these values are programmed by the user into the **FIFO Configuration** register.

The upper threshold values determine when data transfers will be requested. The lower threshold values set when the request will be deasserted.

The **SDREQ1#** signal (and the **sdreq** bit in the **Status** register) will be asserted when the number of bytes of empty space in the Source FIFO is greater than the source FIFO upper threshold.

The **SDREQ1#** signal (and the **sdreq** bit in the **Status** register) will be deasserted when the number of bytes of empty space in the Source FIFO is less than or equal to the Source FIFO Lower Threshold.

The **DDREQ1#** signal (and the **ddreq** bit in the **Status** register) will be asserted when the number of bytes in the Destination FIFO is greater than the destination FIFO upper threshold.

The **DDREQ1#** signal (and the **ddreq** bit in the **Status** register) will be deasserted when the number of bytes in the Destination FIFO is less than or equal to the destination FIFO lower threshold.

If the thresholds are set below 1 for a 16-bit bus, or 3 for a 32-bit bus, the 9600 will ignore the settings and use the values of 1 and 3, respectively. (These are not recommended values, but exist for testing purposes.)

The FIFO thresholds allow flexibility in how data is written to the Source FIFO and read from the Destination FIFO. For example, some DMA controllers operate in a fixed-length burst mode, where each DMA operation involves a fixed number of transfers. Setting the lower FIFO thresholds properly will guarantee a minimum number of bytes that will be able to be transferred in a burst. If this is done, the **DREQ#** signals need only be checked at the beginning of each burst.



For example, if a burst size is 16 bytes, setting Source FIFO upper threshold to 15 will guarantee that at least 16 bytes of empty space are in the Source FIFO when **SDREQ1#** is asserted. Setting the Spource FIFO lower threshold to 15 will guarantee that the **SDREQ1#** signal will be deasserted when the number of bytes of empty space in the Source FIFO is 15 or less.

Similarly, setting the Destination FIFO upper threshold to 15 will guarantee that at least 16 bytes of data are in the Destination FIFO before **DDREQ1#** will be asserted. Setting the Destination FIFO lower threshold to 15 will ensure that the **DDREQ1#** signal will be deasserted if there are 15 or less bytes in the Destination FIFO.

The FIFO Lower Thresholds can also be used to offset the effects of DMA controllers that cannot respond immediately to the deassertion of **DREQ#**. If a DMA controller will continue to transfer several bytes of data after the **DREQ#** signal has been deasserted, then setting the FIFO Lower Threshold correctly will guarantee that there will be room in the FIFOs for these additional bytes. For example, setting the Source FIFO lower Threshold to seven will guarantee that the Source FIFO will be able to accept two additional 32-bit transfers after **SDREQ1#** is deasserted. Similarly, setting the Destination FIFO lower threshold to seven will guarantee that the Destination FIFO will be able to provide two additional 32-bit transfers after the **DDREQ1#** signal is deasserted.

With fixed-length bursts, the FIFO Lower Threshold can be used to give the DMA controller advance warning of the FIFO readiness to accept a burst after the end of the current burst. For example, a controller with a 32-bit bus width and a burst size of 16 bytes could sample **DREQ#** one bus cycle early if the FIFO Lower Threshold were set to 19 rather than 15. This can reduce dead time between back-to-back bursts.

## 2.12.1 Programmed I/O and the FIFOs

Data passed into the Channels using programmed I/O access rather than DMA must still take the FIFOs into consideration. Overflowing the Source FIFO or underflowing the Destination FIFO will cause a fatal error. The following pseudocode demonstrates how the FIFO handshaking is emulated in software. (It shows the transfer of a single record to and from the FIFOs, and assumes that the command has already been issued. If performance were an issue, one would write the code to transfer multiple records instead of just one.)

```
repeat
    Status = Read_Status_Register;
    if (Status[SDREQ] & (input_data > 0)) then
        Write_Status(input_data); /* Set byte
            enables (all four are-enabled except when there are fewer than four
            input bytes remaining (input_data < 4), and possibly on the first
            data transfer). Set the EOR bit on the last transfer of the record
            (input_data <= 4), and EOC on the last transfer of the command.*/
        Write_Data[input_data]; /* Send data word (32 bits
            except on last transfer)*/
        input_data = input_data - 4;
    endif
    if (Status[DDREQ]) then
```

```

Read_Data; /* get a 32-bit word from the FIFO, discard any bytes for
           which the byte enables aren't asserted, and store the valid bytes
           in an output buffer. Also test the Status word for EOR and EOC. */
endif
until (Status[EOR] or Status[EOC]); /* errors cause EOC, so all bases are covered */

```

## 2.13 Clock

The 9600 internal logic is based on a single clock input, **CLK**. An internal **PLL** locks the system bus to **CLK** and provides a frequency-multiplied clock for the core logic. Because the **PLL** must lock to **CLK** after power-up, the chip should not be accessed for some time (see the AC Specifications in [Chapter 6](#)). The frequency multiple is controlled by 10 kW pull-up or pull-down resistors on four pins, which are sampled after each hardware reset. Their values are shown below:

**Table 5: Clock multiplier selection**

SDREQ1#	DDREQ1#	DIV1	DIV0	Frequency Multiple
0	0	1	1	1
0	1	0	1	1.5
0	1	1	1	2
1	0	0	1	2.5
1	0	1	1	3
1	1	1	1	4
All other combinations				<i>Reserved</i>

A "1" in the table indicates a pull-up resistor. A "0" indicates a pull-down.

The **SDREQ1#** and **DDREQ1#** signals are tri-stated on reset to avoid contention with the pull-ups and pull-downs

Because pull-downs will look to the Host as if **SDREQ1#** or **DDREQ1#** are being asserted, the Host should not sample these outputs for three cycles after **RESET#** is deasserted. See [Section 6.2.1](#).

### Clock Multiplier Examples

**Table 6: Clock multiplier example**

Bus Clock (CLK)	Logic Clock	Clock Multiple
25 MHz	75 Mhz	3
33 MHz	66 MHz	2
40 MHz	80 MHz	2
50 MHz	75 MHz	1.5
50 MHz	50 MHz	1

## 3 Register Descriptions

### 3.1 Register Overview

The device register list is shown below. Each Channel has its own set of registers, excluding the shared **Configuration** and **Chip ID** registers. Some of the Channel 2 registers are inaccessible when the device is in Feedback Mode. These registers are shown with bolded red text in the [Table 7](#).

Many registers are 32 bits wide. If the device is operating in 16-bit bus mode, accessing these registers will require two transfers. The first transfer writes to the lower 16 bits of the register, and the second writes to the upper 16 bits. Only the **Data** register is effected by the **big-endian** bit in the **Configuration** register.

In the register descriptions that follow, some bits will be marked "*Reserved*." **Reserved** bits must be written as zeros and ignored when read.

**Table 7: Register List**

Name	Address	
	Channel 1	Channel 2
<b>Data</b>	0	<i>8 (reserved)</i>
<b>Command</b>	1	<i>9 (reserved)</i>
<b>Result</b>	2	10
<b>Configuration</b>	3	
<b>Interrupt Enable</b>	4	12
<b>Status</b>	5	13
<i>Reserved</i>	6	<i>14 (reserved)</i>
<b>FIFO Configuration</b>	7	<i>15 (reserved)</i>
<b>Chip ID</b>	11	

## 3.2 Data (0)

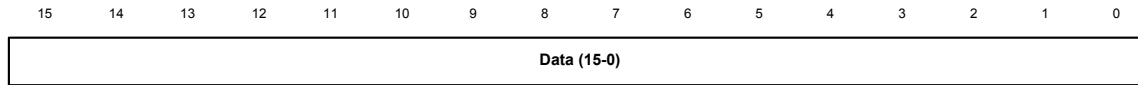


Figure 10. Data (0) 16-bit bus mode

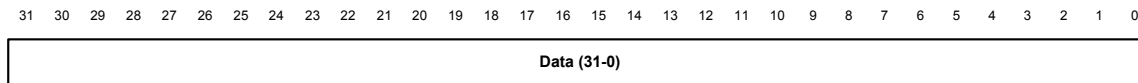


Figure 11. Data (0) 32-bit bus mode

Register 0 is the **Data** register for Channel 1.

This register can be read or written. A write to the **Data** Register writes data to the Source FIFO. A read reads data from the Destination FIFO.

The **Data** register provides access to the FIFOs through the programmed I/O mechanism. The DMA mechanism is more efficient.

Transfers to or from the **Data** register can only take place under the same conditions that a DMA transfer can take place. That is, the state of the FIFO must be taken into consideration. The **Status** register **SDREQ** and **DDREQ** bits reflect the state of the **SDREQ1#** and **DDREQ1#** signals. The **Data** register should only be read when the **DDREQ** is set, and should only be written when the **sdreq** bit is set. If these conditions are not observed, the Channel may terminate the command when an attempt is made to write to a full FIFO or read from an empty one. The **data error** bit in the **Status** register will be set to indicate the error. This is a fatal error which requires a hardware or software reset before a new command can be processed.

For programmed I/O transfers, the behavior of the **EOC#**, **EOR#**, and **B[3-0]** signals must be duplicated in software. This is done through read/write bits in the **Status** register. When setting these bits, the **Status** register is written first, then the **Data** register. See [Section 2.10.2](#) and [Section 2.12.1](#) for further details.

For example, to signal the EOR in programmed I/O compression, the Host would set the **eor** bit in the **Status** register, then write the final data word of the record to the **Data** register.

### 3.3 Command (1)

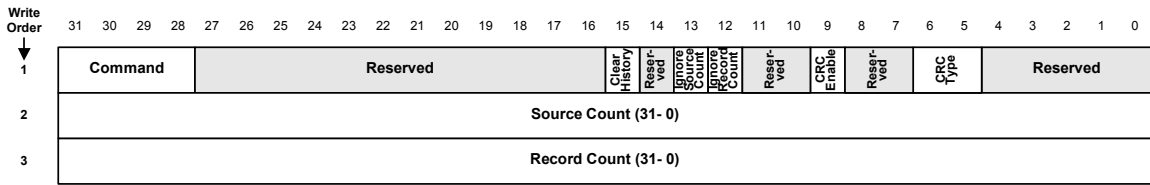


Figure 12. Command (1) 32-bit bus mode

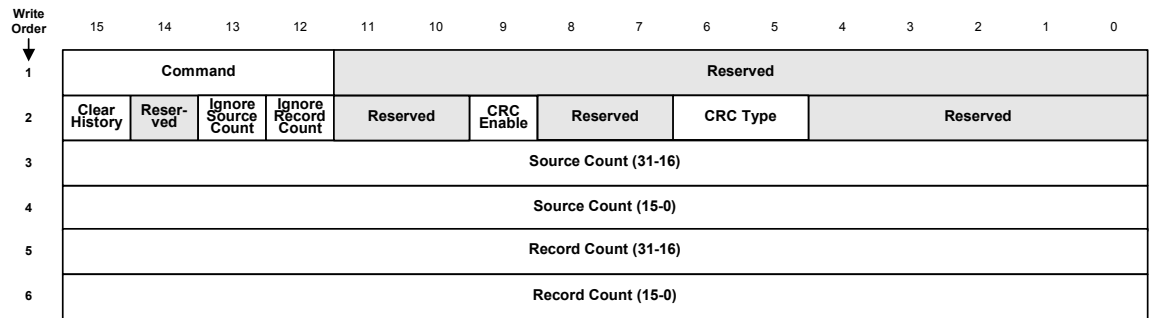


Figure 13. Command (1) 16-bit bus mode

Register 1 is the **Command** register for Channel 1.

The **Command** register can be read or written. It specifies the command to be executed and sets the command parameters.

Three 32-bit write operations (six in 16-bit mode) are required to launch a command. All writes must take place. The **command in progress** bit in the **Status** register may be used to verify the synchronization of write operations. This bit will be set after the first write to the **Command** register, and cleared after the last write. A new command can be issued when the **command ready** bit in the **Status** register is set.

The command may also be read by the CPU. This would not generally be done except for testing purposes. Reading the **Command** register is similar to writing to it. The **command in progress** bit in the **Status** register must be clear before the read transfer is initiated. The entire Command (three 32-bit words) must be read.

### 3.3.1 Commands

The 4-bit command field specifies the function to be executed. These are listed below:

**Table 8: Commands**

Command	Command field
Compress	0
Decompress	1
Passthrough	2
Reserved	3-15

**Compress:** This channel will compress data. See [Section 2.4](#) for details on how the Compress command operates.

**Decompress:** The channel will decompress data. See [Section 2.6](#) for details on how the Decompress command operates.

**Passthrough:** This channel passes data unmodified from input to output. Compression history is not affected. This command is mostly used for diagnostics.

### 3.3.2 Command Fields

**Clear History:** This bit is significant for both compression and decompression, and determines whether compression history will be cleared between records of the same command. History is always cleared before the first record of a command, regardless of the setting of this bit.

On compression, the compression history is only cleared before the first record of the command, if this bit is set to 0. On decompression, the compression history is cleared between all records if this bit is set to 1.

On decompression, this bit tells the Engine whether to clear history between records. If set, history is cleared. Otherwise, it is not. Decompression requires that the state of the **clear history** bit match the compression mode used to compress the data.

**Ignore Source Count:** This bit is significant for the Compress command. If this bit is set, the value of the source count command field is ignored. If it is 0, the source count is significant.

**Ignore Record Count:** This bit is significant for both the Compress and Decompress commands. If set, the record count command field is ignored. If 0, the record count field is significant.

**CRC Enable:** This bit is significant for the compress and decompress commands, but only in 32-bit CRC mode. If this bit is set to 0, the CRC will not be verified against the constant value for a protected record (0xDEBB20E3). If this bit is set to 1, the CRC will be checked. The result of the test is written to the **CRC error**

bits in the **Status** and **Results** registers. Setting this bit enables the CRC test, requires source data to be protected records, and causes CRC errors to become fatal errors. See [Section 2.8.2](#).

**CRC Type:** Selects the checksum algorithm for the current command. This 2-bit field is decoded as follows:

Value	Check Type
00	32-bit CRC
01	16-bit CRC
10	8-bit LCB
11	<i>Reserved</i>

See [Section 2.8](#) for a discussion of CRC/LCB generation.

**Source Count:** This field is the initial value used for the Source Counter. The Source Counter is used only for the Compress command.

For a Compress Command, the Source Count specifies the size of each record to be compressed. Each time the Source Count reaches 0 (unless the **ignore source count** bit in the **Command** register is set to 1), an end marker will be appended and a new compression operation will begin on the next record.

If the Source Count is set to 0, the record length is set to 232 bytes on compression.

**Record Count:** This field is the initial value used for the Record Counter. The Record Count specifies the number of records that will be compressed or decompressed during the command. When the Record Count reaches 0 (unless the **ignore record count** bit in the **Command** register is set to 1), the command will terminate.

If the Record Count is set to 0, 232 records will be processed before the command terminates.

## 3.4 Result (2,10)

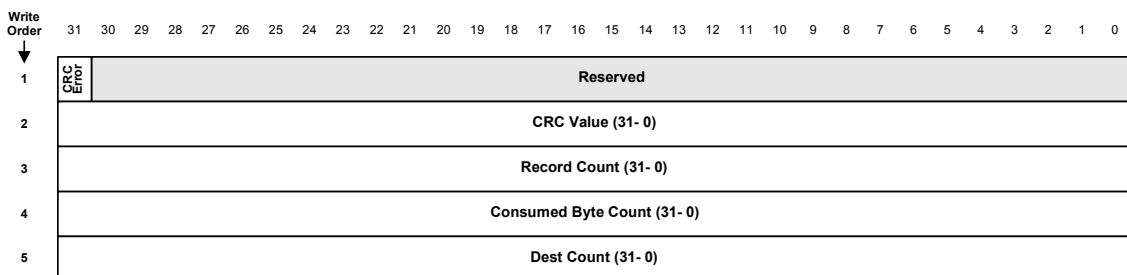


Figure 14. Result (2,10) 32-bit bus mode

Write Order ↓	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	CRC Error	Reserved														
2	Reserved															
3	CRC Value (31-16)															
4	CRC Value (15-0)															
5	Record Count (31-16)															
6	Record Count (15-0)															
7	Consumed Byte Count (31-16)															
8	Consumed Byte Count (15-0)															
9	Dest Count (31-16)															
10	Dest Count (15-0)															

**Figure 15. Result (2,10) 16-bit bus mode**

Register 2 is the **Result** register for Channel 1. Register 10 is the **Result** register for Channel 2.

This read-only register reports the result of a completed command.

Reading the **Result** register is optional. Five read operations (ten in 16-bit mode) are required to properly read the result. The **result in progress** bit in the **Status** register may be used to verify the synchronization of read operations. All reads must take place before the **result in progress** bit will be cleared. This normally has no effect on the operation of the chip, however, as the bit will be cleared in any event when a new command is launched.

The **Results** register are only updated at the end of every command. These registers are not affected by hardware or software reset; its contents will be unknown on power up and will only be known after a command has passed through the 9600. This applies to all blocks in the result register:

- CRC Error,
- CRC Value
- Record Count
- Consumed Byte Count
- Dest Count

### 3.4.1 CRC Error

This bit is significant for the Compress and Decompress commands, but only if the **crc error** bit in the **Command** register is set to 1 and the 32-bit CRC algorithm is selected. It is used to check the integrity of protected records.



If this bit is set to 0, the CRC in the CRC value field was correct, or the test was not enabled. If this bit is set to 1, the CRC in the CRC value field failed the CRC test. Unless the record is a valid protected record, it will fail the test. When the test is enabled, a CRC error will cause the current command to terminate.

See [Section 2.8](#) for details on when this bit will be set.

### 3.4.2 CRC Value

This field is significant for the Compress and Decompress commands. This field represents the calculated 32-bit CRC, 16-bit CRC or 8-bit LCB value for the most recently completed record. The CRC algorithm is selected in the Command word. These calculations are performed on the raw data (input for compression, output for decompression). Values are aligned to the least-significant bit, so a 16-bit CRC occupies bits 15:0, while a longitudinal check byte occupies bits 7:0.

This field will contain results of the selected checksum algorithm on every operation, regardless of the state of the **crc enable** bit.

Due to the way they are generated, protected records always have a 32-bit CRC value of 0xDEBB20E3.

### 3.4.3 Record Count

This field is significant for all commands. It represents the number of records remaining to be processed for the command.

The value of the Record Count result field is calculated as follows: An internal record counter is initialized to one less than the Record Count Command field. The internal counter is decremented with each record processed. When the command terminates, the value of the internal counter is written into the Record Count Result field. (If the command terminated because the number of records specified in the Command word had been processed successfully, this field will be 0.)

### 3.4.4 Consumed Byte Count

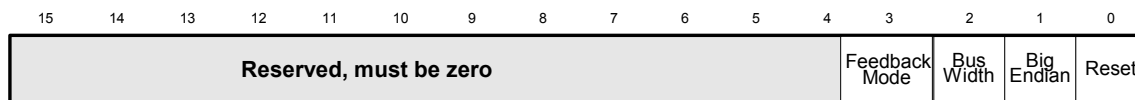
This field is significant for all commands. It contains the number of input bytes processed by the Channel since the start of the command. Neither dummy bytes nor partial records at the end of the command are counted. The primary use of the Consumed Byte Count is to identify the point in the compressed data stream at which decompression can be recommenced. For example, if we issued a command to decompress ten records, the Consumed Byte Count would contain the offset in the compressed data stream from which decompression of the eleventh record could begin with a new command.

The Consumed Byte Count is cleared at the beginning of the command. A temporary register at the input of the Engine (below the Source FIFO) counts the valid bytes in the input data stream. Valid bytes are those bytes whose address enables were asserted. At each EOR, the count in the temporary register is added to the Consumed Byte Count. Since the Consumed Byte Count is only updated at EOR, partial records at the end of a command do not contribute to the count.

## 3.4.5 Destination Count

This field is significant for all commands. The Destination Count starts each command at 0 and increments with each destination byte produced by the Channel. Thus, the Destination Count gives the number of valid output bytes produced by the command.

## 3.5 Configuration (3)



There is only one **Configuration** register, which configures both Channels. This register may be read or written. It is used to configure chip options. The default value of all the fields in this register after a hardware reset is 0.

### 3.5.1 Feedback Mode

Feedback Mode selects the mode of operation. If set, the device operates in feedback mode.

Value	Mode
0	Normal
1	Feedback

### 3.5.2 Bus Width

This bit selects the width of the system bus. If this bit is set to 1, the bus width will be 32 bits. If this bit is set to 0, the bus width will be 16 bits. This bit is significant only for the data bus and three registers: the **Data** register, the **Command** register, and the **Result** register. All other registers are 16 bits.

### 3.5.3 Big Endian

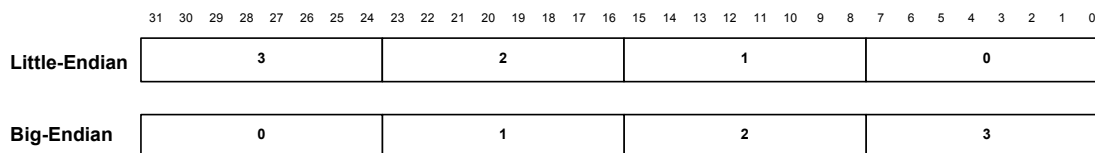
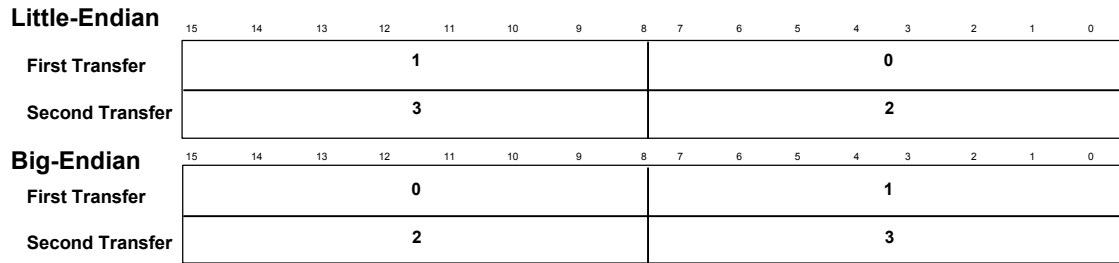


Figure 16. Big Endian 32-bit bus mode



**Figure 17. Big Endian 16-bit bus mode**

The **big-endian** bit selects the byte order of data transferred via the DMA interfaces or the **Data** register. Byte ordering of registers other than the **Data** register will not be affected. Bytes are processed in the order shown (that is, byte 0 first, then bytes 1, 2, and 3).

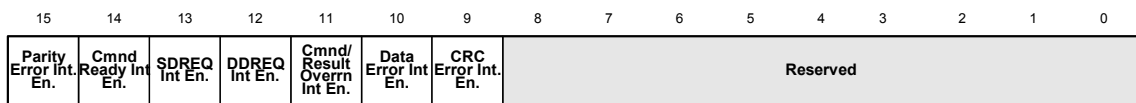
If this bit is set to 0, this chip operates as if it were attached to a Little Endian processor. Bits 7-0 of the data bus are processed first. If this bit is set to 1, this chip operates as if it were attached to a Big Endian processor. In 32-bit bus mode, data bus bits 32-24 will be processed first. In 16-bit mode, data bus bits 15-8 will be processed first.

### 3.5.4 Reset

Setting this bit is similar to asserting and deasserting the hardware **RESET#** signal. The chip will immediately stop any current activity and go into a known state. The difference between this bit and the **RESET#** signal is that this bit will not clear the Configuration or FIFO Configuration registers.

During a hardware reset, this bit is set in hardware until the device is ready for normal operation. Do not access the device (except to read the **Configuration** register) after a reset until after the **reset** bit returns to 0. Writing a 0 to this bit will not clear the **reset**, it will clear automatically.

## 3.6 Interrupt Enable (4,12)



This register may be read or written. The default value of all the bits in this register after a reset is 0.

### 3.6.1 Parity Error Interrupt Enable

While this bit is set to 1, the **IRQ#** signal will be asserted while the **parity error** bit in the **Status** register is set to 1.

## 3.6.2 Command Ready Interrupt Enable

While this bit is set to 1, the **IRQ#** signal will be asserted while the **command ready** bit in the **Status** register is set to 1.

## 3.6.3 SDREQ Interrupt Enable

While this bit is set to 1, the **IRQ#** signal will be asserted while the **sdreq** bit in the **Status** register is set to 1.

## 3.6.4 DDREQ Interrupt Enable

While this bit is set to 1, the **IRQ#** signal will be asserted while the **ddreq** bit in the **Status** register is set to 1.

## 3.6.5 Command/Result Overrun Interrupt Enable

While this bit is set to 1, the **IRQ#** signal will be asserted while the **command/result overrun** bit in the **Status** register is set to 1.

## 3.6.6 Data Error Interrupt Enable

While this bit is set to 1, the **IRQ#** signal will be asserted while the **data error** bit in the **Status** register is set to 1.

## 3.6.7 CRC Error Interrupt Enable

While this bit is set to 1, the **IRQ#** signal will be asserted while the **crc error** bit in the **Status** register is set to 1.

## 3.7 Status (5,13)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Parity Error	Cmnd Ready	SDREQ	DDREQ	Cmnd/Result Overrun	Data Error	CRC Error	Reserved	Cmnd In Prog.	Result In Prog.	Byte Enable (3-0)			EOC	EOR	

The **Status** register reports the status of the Channel. The default value of the bits in this register are listed under each bit description.

Only the Byte Enable, EOC, and EOR fields are affected by writes. Writing to these bytes affects the Source FIFO logic as if the corresponding signals had been asserted. Reading them reads the state of the signals as set by the Destination FIFO logic.

The error bits - **Parity Error**, **Command/Result Overrun**, **Data Error**, and **CRC Error** - are persistent across commands. Writing to them has no effect. They can only be cleared by a hardware or software reset.

### 3.7.1 Parity Error

The read-only **parity error** bit is set when the Channel detects a parity error on incoming data. The chip supports byte-wise parity on **P[3:0]**.

This error bit is persistent across commands. It can only be cleared through a hardware or software reset.

### 3.7.2 Command Ready

The read-only **command ready** bit is set to 1 when the Channel is ready to accept a new command. It also indicates that the Channel is idle and that the results of the previous command can be read from the **Result** register.

This bit is cleared by writing the first word to the **Command** register. When cleared, this bit will not be set to 1 again until the chip is ready to accept another new command (after the current command has completed).

### 3.7.3 SDREQ

This read-only bit is set to 1 when the number of bytes of empty space in the Source FIFO is greater than the Source FIFO Upper Threshold value programmed in the **FIFO Configuration** register.

This bit is set to 0 when the number of bytes of empty space in the Source FIFO is less than or equal to the Source FIFO Lower Threshold value programmed in the **FIFO Configuration** register.

The **sdreq** bit will operate identically to the **SDREQ1#** signal. That is, this bit will be 0 when the **SDREQ1#** signal is inactive (high), and 1 when the **SDREQ1#** signal is active (low).

If CPU I/O is used, the **SDREQ1#** signal is not used. However, the operation of the **sdreq** bit remains the same as the **SDREQ1#** signal.

### 3.7.4 DDREQ

This read-only bit is set to 1 when the number of bytes in the Destination FIFO is greater than the Destination FIFO Upper Threshold value programmed in the **FIFO Configuration** register.

This bit is set to 0 when the number of bytes in the Destination FIFO is less than or equal to the Destination FIFO Lower Threshold value programmed in the **FIFO Configuration** register.

The **ddreq** bit will operate the same as the **DDREQ1#** signal. That is, this bit will be 0 when the **DDREQ1#** signal is inactive (high), and 1 when the **DDREQ1#** signal is active (low).

If CPU I/O is used, the **DDREQ1#** signal is not used. However, the operation of the **ddreq** bit remains the same as the **DDREQ1#** signal.

## 3.7.5 Command/Result Overrun

This read-only bit is set to 1 if the **Command** register was written or the **Result** register was read while a command was in progress.

When set, this bit is persistent across commands and can only be cleared by a hardware or software reset.

## 3.7.6 Data Error

This read-only bit is set to 1 if the Source FIFO is written when it is full (overflow), or the Destination FIFO is read when it is empty (underflow).

When set, this bit is persistent across commands and can only be cleared by a hardware or software reset.

### Note

**Data error** Bit will be set at the EOR when Destination FIFO send out EOR and follow with "DUMMY" bytes. Just ignore **data error** bit when use multiple record per command.

## 3.7.7 CRC Error

This read-only bit follows the state of the **crc error** bit in the **Result** register. See [Section 3.4.1](#) for further details.

When set, this bit is persistent across commands and can only be cleared by a hardware or software reset.

## 3.7.8 Command In Progress

This read-only bit indicates that a command is currently in the middle of being written or read. This bit becomes set to 1 after the first access to the **Command** register. This bit returns to 0 after the last access to the **Command** register.

## 3.7.9 Result In Progress

This read-only bit indicates that a result is currently in the middle of being read. This bit becomes set to 1 after the first read from the **Result** register. This bit returns to 0 after the last read from the **Result** register, or when a new command is launched.

## 3.7.10 Byte Enable

The read/write Byte Enable field allows the information provided by the **B[3-0]** signals during DMA to be transferred when the data stream is provided through programmed I/O.

### 3.7.10.1 Write Transfer

During a write transfer to the **Data** register, this field indicates which bytes of the next write to the **Data** register are valid. Invalid bytes are treated as dummy bytes. See [Table 9](#) to determine which bits correspond to which bytes.

The default value is 0b1111 (all bytes valid). The byte-enables are reinitialized to this value after every write to the **Data** register.

### 3.7.10.2 Read Transfer

During a read transfer from the **Data** register, this field will indicate the valid bytes for the next read transfer. Invalid bytes should be treated as dummy bytes. See [Table 9](#) to determine which bits correspond to which bytes.

**Table 9: Byte enable**

BYTE ENABLE 0	D[7-0]
BYTE ENABLE 1	D[15-8]
BYTE ENABLE 2	D[23-16]
BYTE ENABLE 3	D[32-24]

### 3.7.10.3 EOC

The **eoc** bit allows the information provided by the **EOC#** signals during DMA to be transferred when the data stream is provided through programmed I/O.

### 3.7.10.4 Write Transfer

During a write transfer to the **Data** register, this bit must be set to 1 before the last write transfer of a command. The next write to the **Data** register will be flagged as the last data of the current command. A new command must be started before any further data will be processed.

The default value for EOC is 0 (no EOC). The **eoc** bit is reset to 0 after each write to the **Data** register.

### 3.7.10.5 Read Transfer

During a read transfer from the **Data** register, this bit will be set to 1 if the next read transfer is the last transfer of a command. No new data will be processed until a new command is issued.

## 3.7.11 EOR

The **eor** bit allows the information provided by the **EOR#** signals during DMA to be transferred when the data stream is provided through programmed I/O.

## 3.7.11.1 Write Transfer

During a write transfer from the **Data** register, this bit must be set to 1 before the last write transfer of a record. The next write to the **Data** register will be flagged as the last data of the current record.

This bit is reset to its default value of 0 (indicating no EOR) after every write to the **Data** register.

## 3.7.11.2 Read Transfer

During a read transfer from the **Data** register, this bit will be set to 1 if the next read transfer is the last transfer of a record. Data following the next transfer will be from the next record.

## 3.8 FIFO Configuration (7)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U/L = 0		Rsrvd	Source FIFO Lower Threshold				Reserved		Dest FIFO Lower Threshold						
U/L = 1		Rsrvd	Source FIFO Upper Threshold				Reserved		Dest FIFO Upper Threshold						

Register 7 is the **FIFO Configuration** register for Channel 1.

This register may be read or written. It configures the thresholds at which a FIFO will request data transfers. These thresholds determine when the FIFO ready status bits in the **Status** register and the DMA request signals will be asserted. See [Section 2.12](#) for more information on FIFO programming.

FIFO threshold values can be in the range of 0-59.

### 3.8.1 U/L

This bit determines whether the upper or lower threshold pairs are accessed.

On writes, if this bit is 0, the lower FIFO thresholds for the Source and Destination FIFOs will be accessed. If this bit is set to 1, the upper FIFO thresholds will be accessed.

On reads, the **u/l** bit reports whether the upper or lower thresholds were read. When reading this register, the upper and lower halves will be returned alternately, but there is no way to request one or the other. By reading the register twice and examining the **u/l** bits, there is no ambiguity over which data is which.

### 3.8.2 Source FIFO Threshold

This field sets both the upper and lower source FIFO thresholds. If the **u/l** bit is set to 0, the lower FIFO threshold will be set. If the **upper/lower** bit is set to 1, the upper FIFO threshold will be set.



When the number of bytes of empty space in the source FIFO is greater than the value set in the Source FIFO Upper Threshold field, the **sdreq** bit in the **Status** register will be set to 1 and the **SDREQ1#** pin will become active. When the number of bytes of empty space in the source FIFO is less than or equal to the value set in the Source FIFO Lower Threshold field, the **sdreq** bit in the **Status** register will be set to zero and the **SDREQ1#** pin will become inactive. See [Section 2.11.1](#) and [Section 2.12](#) for more information regarding this field.

Allowable values for this field are 0 to 59. The default values for both upper and lower thresholds after reset are 0x00.

**SDREQ** will never be asserted unless there is at least one full bus transfer that can take place (except for the last transfer of the command, which can be less than full-width).

### 3.8.3 Destination FIFO Threshold

This field sets both the upper and lower Destination FIFO thresholds. If the **u/l** bit is set to zero, the lower FIFO threshold will be written. If the **u/l** bit is set to 1, the upper FIFO threshold will be set.

When the number of bytes in the Destination FIFO is greater than the value set in the Destination FIFO Upper Threshold field, the **ddreq** bit in the **Status** register will be set to 1 and the **DDREQ1#** pin will become active. When the number of bytes in the Destination FIFO is less than or equal to the value set in the Destination FIFO Lower Threshold field, the **ddreq** bit in the **Status** register will be set to zero and **DDREQ1#** pin will become inactive. See [Section 2.11.2](#) and [Section 2.12](#) for more information regarding this field.

Allowable values for this field are 0 to 59. The default values for both upper and lower thresholds after reset are 0x00.

**DDREQ** will never be asserted unless there is at least one full bus transfer that can take place (except for the last transfer of the command, which can be less than full-width).

This field defaults to zero on reset.

## 3.9 Chip ID (11)



This register may only be read. The upper 8 bits are defined as the product ID code and the lower 8 bits are reserved. When this register is read, it returns the Chip ID value of 0x08XX.

THIS PAGE INTENTIONALLY LEFT BLANK



## 4 Signal Descriptions

### 4.1 CPU Interface

This section describes the CPU interface signals on the system bus. The CPU interface shares some signals with the DMA interface.

#### 4.1.1 System Data Bus (D[31-0])

Bi-directional system data bus. This bus may be configured for either 16-bit or 32-bit operation with the **bus width** bit in the **Configuration** register.

While configured for 16-bit mode, the upper 16 bits (**D[31-16]**) may be left unconnected, as they are terminated with internal pull-up resistors.

The system data bus is shared with the DMA interface.

#### 4.1.2 Parity ([3-0])

Bi-directional parity bus for the data bus. Each bit reflects 8-bits of the data bus. The least-significant bit corresponds to the least significant byte of the bus. If the bus is configured for 16-bit operation, only PB1-PB0 are significant. The other two parity signals may be left unconnected, as they have internal pull-up resistors.

The parity bus is shared with the DMA interface.

Odd parity is used. That is, a byte of 00000000b would result in a corresponding parity bit of '1'.

#### 4.1.3 Address (A[3-0])

Address input signals for the CPU Interface. These signals are significant only for register accesses, not for DMA accesses.

#### 4.1.4 Read/Write Command (R/W#)

This input signal determines the direction of the data bus during a register transfer. If this signal is high, a read transfer will take place. If this signal is low, a write transfer will take place.

## 4.1.5 Command Strobe (CS#)

Active low input. While this signal is active, a register access will take place. The data direction is determined by the **R/W#** signal.

## 4.1.6 Interrupt (IRQ#)

Active low output. This signal will become active when any event occurs that is enabled in either of the **Interrupt Enable** registers. Once asserted, **IRQ#** is persistent across commands, and will not be deasserted until the 9600 receives a hardware or software reset, or until the appropriate interrupt enable bits are cleared. See the **Interrupt Enable** register description for further information about when this signal is asserted and deasserted.

This signal is an open collector output requiring an external pull-up resistor to VCC.

## 4.2 DMA Interface

This section describes the DMA interface signals on the system bus. Inputs to the 9600 are sampled on the rising edge of **CLK**. Outputs are valid on the rising edge of **CLK**. The bus protocol is covered in Section 5.3.3. AC timing parameters are covered in [Section 6.2](#).

### 4.2.1 Byte Enable (B[3-0])

Bi-directional byte-enable for the data bus. Each bit reflects an enable for eight bits of the data bus. The least significant bit corresponds to the least significant byte of the bus. This signal is driven by the Host on writes to the Source FIFO and by the chip on reads from the Destination FIFO.

For 16-bit operation, **B[3:2]** can be left unconnected, as they have internal pull-up resistors.

### 4.2.2 DMA Ready (RDY#)

Active low input. This signal enables data transfers within the 9600. When asserted, a transfer on the current cycle is completed (written into the Source FIFO or read from the Destination FIFO). When deasserted, the transfer is deferred.

Deasserting **RDY#** on DMA reads from the Destination FIFOs causes output wait states. The 9600 will drive the same Destination FIFO data on every clock cycle until **RDY#** is asserted. Asserting **RDY#** completes the bus transfer and enables FIFO clocking so that new data will appear on the DMA transfer.

Deasserting **RDY#** on input transfers to the Source FIFOs causes input wait states. Data on the bus is ignored until **RDY#** is asserted. When **RDY#** is asserted, the data latched on the rising edge of the current clock cycle is written to the Source FIFO.

This signal is shared between both FIFOs.

### 4.2.3 End of Command (EOC#)

Bi-directional EOC signal. This signal indicates the current DMA transfer contains the last byte of the current command, or, if the current transfer is a dummy transfer (no byte-enables asserted), that the previous transfer contained the last byte of the current command. This signal is used for both Channel 1 and Channel 2.

During a DMA transfer to the Source FIFO, this signal operates as an input. During a DMA transfer from the Destination FIFO, this signal operates as an output.

### 4.2.4 End of Record (EOR#)

Bi-directional EOR signal. This signal is both an active low input and an active low output. This signal indicates the current DMA transfer contains the last byte of the current record, or, if the current transfer is a dummy transfer (no byte-enables asserted), that the previous transfer contained the last byte of the current record. This signal is used for both Channel 1 and Channel 2.

During a DMA transfer to the Source FIFO, this signal operates as an input. During a DMA transfer from the Destination FIFO, this signal operates as an output.

### 4.2.5 Source FIFO DMA Request (SDREQ1#)

Active low output. This signal becomes active when the Source FIFO contains enough free space to accept a DMA burst, as determined by the upper FIFO threshold. The signal will become inactive when the Source FIFO contains a number of data bytes greater than the source FIFO lower threshold. See [Section 2.11.1](#) and [Section 2.12](#) for more information on FIFO usage. Both **DREQ#** signals (**DDREQ1#**, **SDREQ1#**) may be active at the same time. It is the responsibility of the Host to choose which FIFO to service.

### 4.2.6 Source DMA Acknowledge (SDACK1#)

Active low input. The Host asserts the **SDACK#** signal to initiate DMA writes to the Source FIFO. Only one of the two **DACK#** signals (**DDACK1#**, **SDACK1#**) should be asserted at the same time. The **SDACK1#** signal causes the 9600 to accept data on the next clock cycle. The data will be written to the Source FIFO if **RDY#** is asserted. If **RDY#** is not asserted, the data is not written to the FIFO. See [Section 2.11.1](#) and [Section 2.12](#) for more information on FIFO usage.

### 4.2.7 Destination FIFO DMA Request (DDREQ1#)

Active low output. This signal becomes active when the Destination FIFO contains enough data to deliver a DMA burst, as determined by the upper FIFO threshold. The signal will become inactive when the Destination FIFO contains bytes of data

less than or equal to the value in the Lower FIFO threshold, or, at the end of a record or command, when the FIFO is empty. See [Section 2.11.2](#) and [Section 2.12](#) for more information on FIFO usage. Both **DREQ#** signals (**DDREQ1#**, **SDREQ1#**) may be active at the same time. It is the responsibility of the Host to choose which FIFO to service.

### 4.2.8 Destination DMA Acknowledge (DDACK1#)

Active low input. The Host asserts the **DDACK#** signal to initiate DMA reads from the Destination FIFO. Only one of the two **DACK#** signals (**DDACK1#**, **SDACK1#**) should be asserted at the same time. The **DDACK1#** signal causes the 9600 to drive the data bus with data after a one-cycle delay. **RDY#** is used to insert wait states. See [Section 2.11.2](#) and [Section 2.12](#) for more information on FIFO usage.

### 4.2.9 Command Ready

Active-low output. The **CMDRDY** signal is an active-low output that indicates that Channel 2 is ready to accept a new command. It is used for Feedback-Mode DMA handshaking. When the 9600 is not in feedback mode, only **EOC#** should be used to detect command termination.

## 4.3 Miscellaneous Signals

### 4.3.1 Reset (RESET#)

Active low input. While this signal is active, the chip will immediately stop any current activity and go into a known state. After a hardware reset, the Compression History will be cleared before its first use.

The chip requires several cycles to reset itself. The **reset** bit in the **Configuration** register will be set until this process is complete. Except for reading the **Configuration** register, the chip should not be accessed until the **reset** bit clears.

### 4.3.2 Clock (CLK)

**CLK** is the system bus clock. The 9600's internal Engines are driven at 1-4 times the rate of **CLK**. See [Section 2.13](#).

### 4.3.3 DIV1, DIV0

These 2 inputs are used in conjunction with the **SDREQ1#** and **DDREQ1#** pins at reset time to determine the internal **PLL** clock frequency, as shown in [Table 5](#) in the previous section. These two inputs may be pulled up or down by 10 k $\Omega$  external resistors.

## 5 Timing Descriptions

Both the CPU and DMA interfaces on the system bus operate in a synchronous mode. All bus signals are relative to the rising edge of the **CLK** signal.

### 5.1 CPU Interface

A typical CPU access consists of two clock cycles (T1 and T2), with any number of wait states (Tw) between the T1 and T2 cycles. A bus cycle with no CPU activity is identified as Ti. This bus mode supports a minimum bus cycle time of two clock cycles.

A T1 cycle is identified by the assertion of the **CS#** signal by the end of a clock. By the end of T1, the **R/W** and **ADDR** signals must also be valid. The following clock cycle will be either T2 or Tw based on the value of the **CS#** signal at the end of the following clock cycle. If **CS#** is active, then this cycle will be Tw. If **CS#** is inactive, then this cycle will be T2. There can be any number of Tw cycles (including zero).

During a Tw cycle, the data bus will be active. The **R/W** and **ADDR** signals no longer need to be valid. The following clock cycle will be either T2 or Tw based on the value of the **CS#** signal at the end of the following clock cycle. If **CS#** is active, then this cycle will be Tw. If **CS#** is inactive, then this cycle will be T2.

During T2, the data transfer remains active. The following clock cycle will be either T1 or Ti based on the value of the **CS#** signal at the end of the following cycle. If **CS#** is active then this cycle will be T1. If **CS#** is inactive, then this cycle will be Ti.

There can be any number of Ti cycles (including zero). During a Ti cycle, **CS#** and the data bus will be inactive.

### 5.2 DMA Interface

A DMA access is initiated when the Host asserts one of the **DACK#** signals (**DDACK1#**, **SDACK1#**), and ends one cycle after **DACK#** deassertion. The **DACK#** signal indicates that the Host will either read or write the bus on the next clock cycle.

Data transfers must only take place when there is space available in the Source FIFO or data available in the Destination FIFO, as indicated by the **SDREQ1#** and **DDREQ1#** signals. If a data overflow or underflow occurs during a command, the **data error** bit is set in the **Status** register. The device must be reset in hardware or software to recover from this error. The exception to this rule is that both reads and writes after EOC are ignored.

A transfer will take place on a rising clock edge if the **DACK#** signal was active on the previous rising clock edge, and the **RDY#** signal is active on the current rising clock edge. If both **DACK#** and **RDY#** are kept active, the device will burst one data transfer every clock.

When no **DACK#** is asserted, the bus is considered to be idle. When **DACK#** is asserted without **RDY#**, the bus is considered to be in a wait state. A burst ends one cycle after **DACK#** is deasserted.

Note that **DREQ#** may or may not be active following the first data transfer, based on the values set in the **FIFO Configuration** register and the amount of data in the FIFO.

## 5.3 DMA Examples

The programmable FIFO thresholds allow the Host DMA interface to be designed in several ways. Depending on the burst size and the FIFO thresholds, the Host may have to sample **DREQ#** once per cycle, once per burst, or at some intermediate value. Similarly, leaving more room in the FIFOs will allow the Host DMA controller to take extra cycles to start or stop bursting without overflowing or underflowing the FIFOs.

The DMA timing diagrams in the following sections are provided as an example of how to implement a DMA controller. The following sections analyze timing for the following conditions:

- Burst transfer
- Bus turn around
- Fixed length burst transfer
- Variable length burst transfer
- Burst with wait states
- Burst with dummy bytes

### 5.3.1 Timing Diagram Terminology

- Cycles start and end on a rising edges.
- The "beginning" of a cycle is the leading rising edge, and the "end" of a cycle is the trailing rising edge. This is shown in the diagrams by the cycle numbers located in between the rising edges.
- The term "clock 4" refers to the sample point at the end of cycle 4. The term "cycle 4" indicates the entire clock cycle.
- Signals are driven at the beginning of the cycle and sampled at end of the cycle.

Example: In [Figure 22](#), **DACK#** is driven low by the system at the beginning of cycle 3, and it is sampled low by the 9600 at the end of clock 3.



### 5.3.2 Burst Transfer

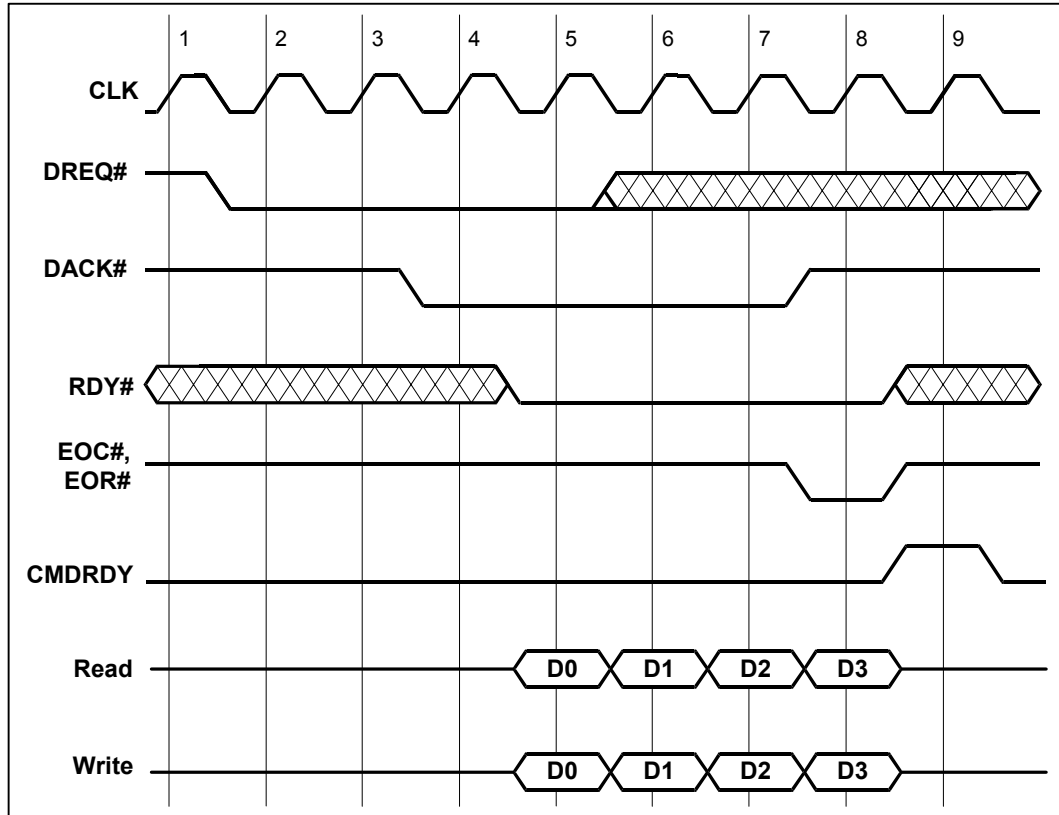


Figure 18. Four-transfer DMA burst

DMA transfers consists of data bursts. A burst may consist of only one transfer. The maximum burst size is limited by the amount of space in the Source FIFO (for a write transfer), or the amount of data in the Destination FIFO (for a read transfer).

Figure 22 shows a four-transfer DMA burst. The 9600 will assert its **DREQ#** signals for any FIFOs that are ready to accept a data transfer. All **DREQ#** signals (**DDREQ1#**, **SDREQ1#**) may be active at once. The Host DMA controller selects a target FIFO and asserts one of the **DACK#** signals (**SDDACK1#**, **DDACK1#**).

The 9600 samples signals on the rising edge of **CLK**. One cycle after the 9600 samples **DACK#** as active, the **RDY#** signal will indicate if a data transfer is to take place. A data transfer will take place on the current clock if **RDY#** is sampled active in the current cycle and **DACK#** was sampled active in the previous cycle. When the 9600 samples **DACK#** active and **RDY#** inactive, the current cycle is a wait state.

Holding **DACK#** and **RDY#** active for more than one clock will transfer one byte per clock. This figure shows a four-transfer burst. **DACK#** is sampled active at the end of cycles 3-6, **RDY#** is sampled active at the end of cycles 4-7, and data is transferred at the end of clocks 4-7. The size of bursts the device can handle can be set using the FIFO Threshold values, as described in Section 3.13.

Figure 22 also shows an **EOC#** or **EOR#** and **CMDRDY** occurring at the end of the burst. The **EOR** indicates that the current transfer contains the last byte of a record. The **EOC#** indicates that the current transfer contains the last byte of a command. Note that the **EOR#/EOC#** signal is shown at the end of a burst, but this is not necessary to conclude a burst. Figure 23 shows the timing if **EOR#** or **EOC#** occur in the middle of a burst. **CMDRDY** is asserted only in feedback mode, when Channel 2 has finished the command. Its position relative to **EOC#** is undefined. It may occur at the same time or later. If Channel 2 detects an error, **EOC#** may not be asserted at all. Only **CMDRDY** will be asserted. See Section 4.2.9.

## 5.4 Bus Turn-Around Time

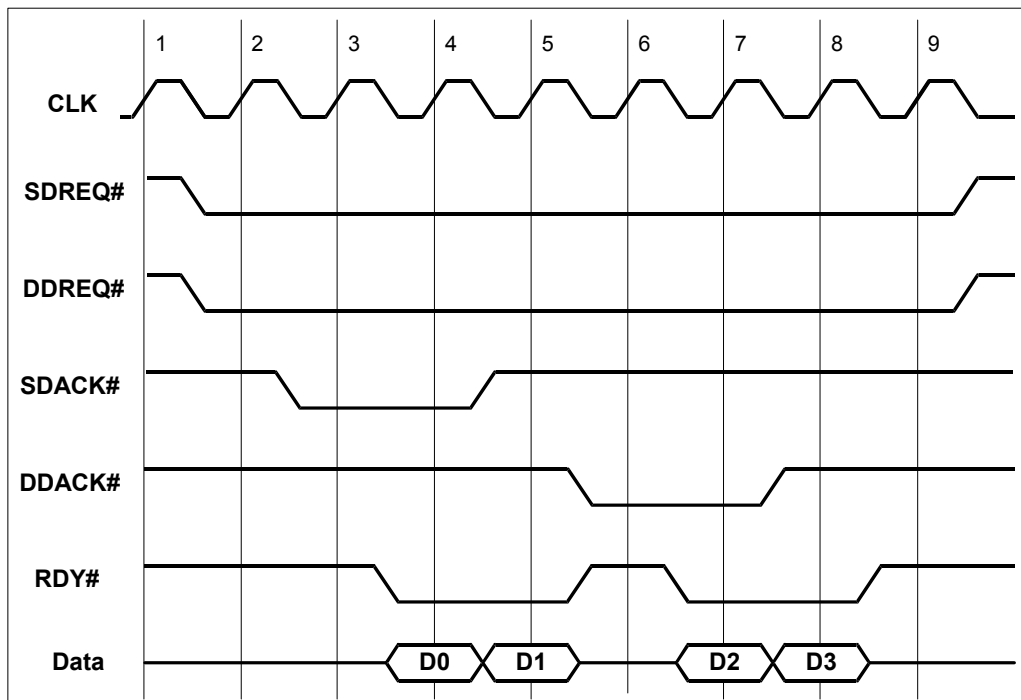


Figure 19. DMA, bus turn around time

Figure 19 shows the bus turn-around time for transfers from two different FIFOs. This occurs when the Host stops servicing one FIFO and switches to another one. For example, when switching between writing to the Source FIFO and reading from the Destination FIFO. Once the Host releases the bus by deasserting one

**DACK#** signal, it may claim the bus by asserting a different **DACK#** after one clock cycle. That is, one dead clock cycle must come between two different accesses to the bus.

The status of the individual **DREQ#** signals is based on their respective FIFO thresholds, and are independent of which **DACK#** is currently active.

## 5.5 Fixed-Length Burst Transfers

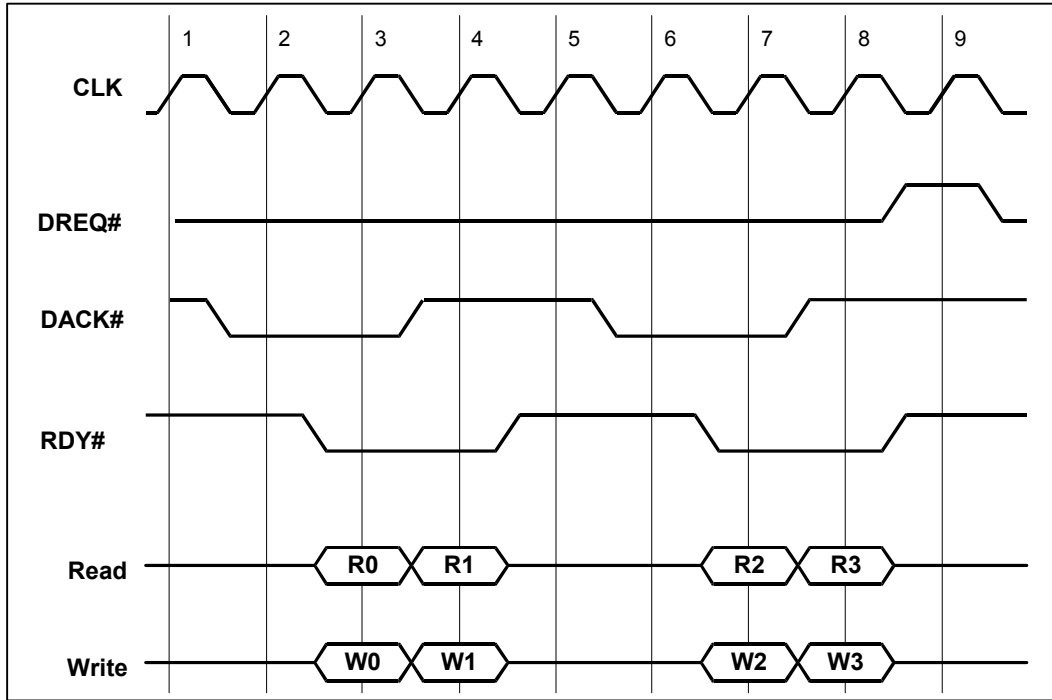


Figure 20. DMA, fixed length transfer

Figure 20 shows a fixed-length burst transfer. During a fixed length burst, a set number of transfers will take place. After the burst has finished, the Host will check **DREQ#** to see if another burst may take place.

The 9600 itself does not differentiate between fixed-length or variable-length transfers. The difference is in how the FIFO thresholds are programmed and how the Host DMA logic treats the **DREQ#** signals. For fixed-length transfers, the FIFOs are set so that both the lower and upper threshold are the same, and are set to accommodate a complete burst. Knowing that the **DREQ#** signals will not be asserted unless the FIFO can read or write an entire burst, the Host DMA logic only needs to sample the **DREQ#** signals at the beginning of the burst.

The **DREQ#** signal may become deasserted at some point during the burst, but this has no effect on the transfer. Bus transfers can occur regardless of the state of **DREQ#**.

Figure 20 shows two two-transfer bursts. Each burst begins at the end of the clock that the 9600 samples **DACK#** active and ends at the end of the clock that the 9600 samples **DACK#** inactive. Once the burst completes, the Host can sample **DREQ#** at the end of the next clock following the end of the burst. This will determine if another burst to the same FIFO may occur. In Figure 20, **DREQ#** is active after the first burst, but deasserted after the second burst. Thus, another burst may not occur.

If a command ends in the middle of a DMA read burst, all transfers after EOC, but still within the burst, will consist of dummy bytes (with all byte enables deasserted). After the command ends and the burst containing **EOC#** is complete, the chip's DMA interface is disabled, with the **DREQ#**, **DACK#**, and **RDY#** signals deasserted. The 9600 will ignore any attempts to read and write over the DMA interface until the next command is issued.

## 5.5.1 Reducing Dead Time

The two-clock delay occurs because **DREQ#** is sampled on the clock edge after the last transfer, and **DACK#** cannot be activated again until it is determined that **DREQ#** is still active.

The figure shows two cycles of dead time for both read and write operations. Dead time can be reduced to one bus cycle by setting both FIFO thresholds to one transfer larger than the burst size. This will cause **DREQ#** to be asserted one cycle earlier. For example, the threshold can be set to 19 instead of 15 for 16-byte bursts with a 32-bit bus. This works because the 9600 will transfer data when **DREQ#** is deasserted.

## 5.6 Variable-Length Burst Transfers

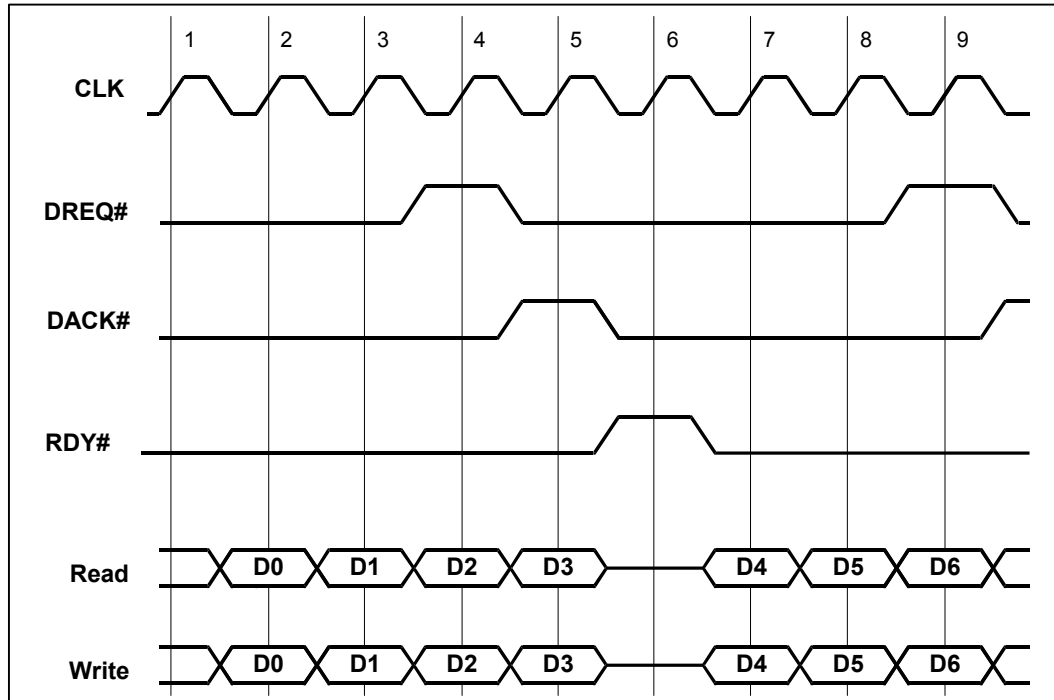


Figure 21. DMA, variable length transfer

Figure 21 shows the timing for a variable length burst transfer. In this case, the Host DMA controller has been designed to sample **DREQ#** on every clock cycle, and to continue DMA transfers until **DREQ#** is deasserted by the 9600. This will occur when FIFO's Lower Threshold is crossed or when EOC is reached. The number of transfers in the burst is not known in advance.

In Figure 21, the DMA controller samples **DREQ#** signal inactive at the end of cycle 3, due to the lower FIFO threshold being crossed after the transfer of D1 at the end of cycle 2. Then, the DMA controller deactivates **DACK#**, which the 9600 samples inactive at the end of cycle 4, in which D3 was transferred. After the **DACK#** signal is deactivated, the bus will still be active until the end of cycle 5, and the D3 transfer may take place. Thus, if variable length transfers are to be done, there may be another two bus transfers after the DMA controller samples **DREQ#** inactive. In order to prevent a data error from occurring, the lower threshold values must be set to accommodate at least two transfers. In this instance, the lower FIFO threshold must be at least 7 (for a 32-bit bus).

The lower threshold could be set to 3 if **RDY#** were deasserted as **DACK#** is deasserted (9600 samples both signals inactive at the end of cycle 4), because that would inhibit the transfer of D3. In this case, the bus will be driven but no data will be transferred. Thus, the lower FIFO Threshold and **RDY#** determine the number of transfers that may occur after **DREQ#** is detected inactive.

## 5.7 Bursting with Wait States

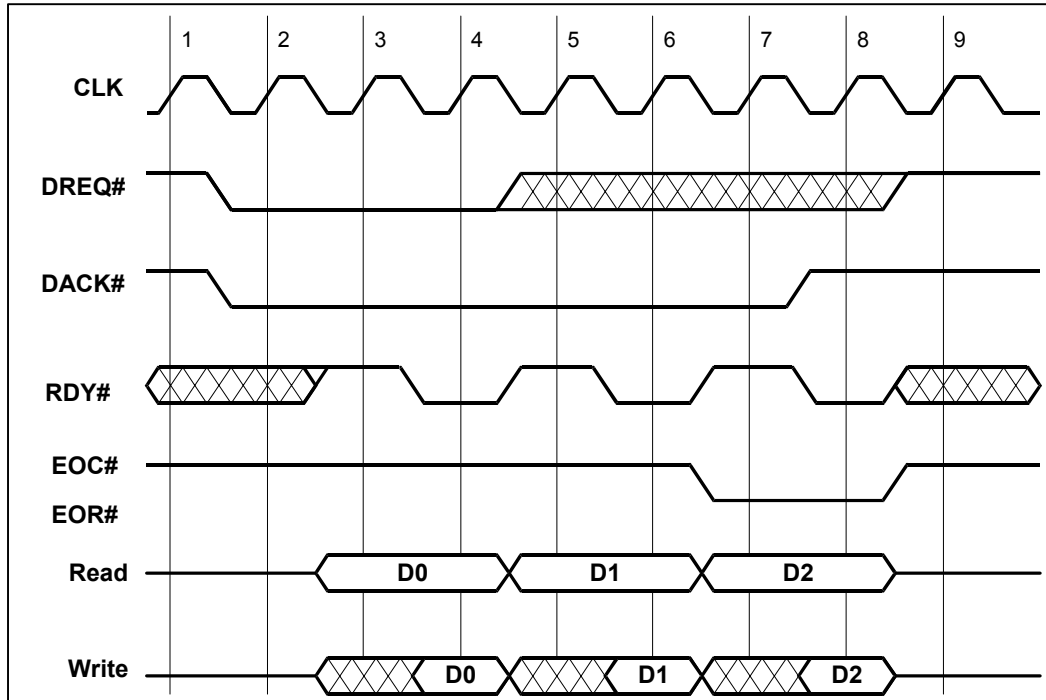


Figure 22. DMA, three-transfer burst with wait states

Figure 22 shows a DMA burst transfer where the Host is using the **RDY#** signal to insert wait states. Wait states are never required by the 9600. They are inserted by the Host to accommodate Host-side delays.

When the **DACK#** signal is held active, the 9600 will continue to be drive the on data transfers from the Destination FIFO. However, the data transfer will complete (either writing the source FIFO or reading the dest FIFO) only at the end of the cycle where **RDY#** is asserted.

Figure 22 also shows the **EOC#** or **EOR#** signal signifying that the D2 transfer represents the last transfer of a record (or command).

## 5.8 Bursting with Dummy Bytes

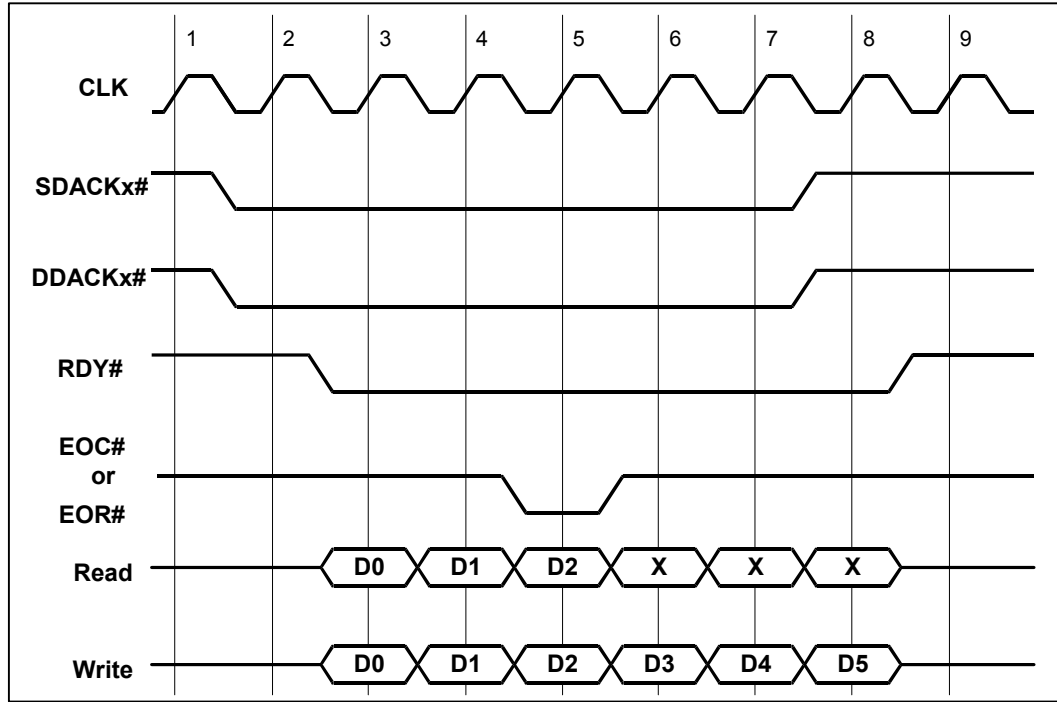


Figure 23. DMA, four-transfer burst

Figure 23 shows a DMA burst where the **EOR#** or **EOC#** signal occurs in the middle of the burst. For example, this might occur at the end of a destination transfer using fixed-length bursts during compression.

See Section 2.10.4 for the data alignment requirements of different kinds of transfers.

### 5.8.1 Write Transfers

For write transfers to the Source FIFO, the Host asserts the **EOR#** signal to indicate that the current transfer contains the last byte of a record. The Host asserts the byte-enable signals to indicate the last valid byte of the record. If the next transfer in the burst contains valid data (as determined by the byte-enables), the data is treated as the start of the next record. On decompression, it is possible that the transfer after **EOR** will contain bytes that the Channel will consider to be part of the current record, if the data in the record was not 32-bit aligned. See Section. 3.11.4

When the Host asserts **EOC#** to signal the end of the command, any transfers between **EOC** and the start of the next command are treated as dummy bytes (regardless of the state of the byte-enables) and discarded.

## 5.8.2 Read Transfers

For read transfers from the Destination FIFO, the EOR and EOC behavior of the 9600 are identical. The **EOR#** and **EOC#** signals indicate that the current transfer contains the last byte of a record or command. Subsequent bus transfers in the burst always consist solely of dummy bytes, and should be ignored by the Host. The next burst (a reassertion of **DACK#** after at least one cycle of **DACK#** deassertion) will read 32-bit-aligned data from the next record.

In spite of the use of dummy bytes, it is still possible to underflow the Destination FIFOs. Dummy bytes are used to pad the end of a burst. A new burst will pull valid data from the FIFO, possibly underflowing it if the DMA controller does not first sample **DREQ#**.



## 6 Specifications

### 6.1 DC Specifications

**Table 10: Absolute maximum ratings**

<b>DC Supply Voltage (VDD)</b>	<b>-0.3V to +7.0V</b>
DC Input Voltage	-0.3V to +7.0V
DC Input Current	±10mA
Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Caution: Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions may affect device reliability.	

**Table 11: Recommended operating conditions**

<b>DC Supply Voltage</b>	<b>+3.0V to +3.6V</b>
<b>Operating Temperature</b>	<b>0°C to +70°C</b>

**Table 12: AC specification definition**

<b>Symbol</b>	<b>Parameter</b>	<b>Conditions</b>
CL2	Load on bus	20 pF
<b>VDD</b>	Supply voltage	3.3V ± 0.3V
<b>VSS</b>	Ground potential	0V
TA	Ambient operating temperature	0°C to +70°C

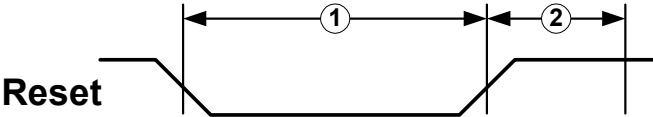
Table 13: DC electrical characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V <sub>IL</sub>	Low level input voltage (I, SI) Clock Input (CI)				0.8	V
					0.8	V
V <sub>IH</sub>	High level input voltage (I,SI) Clock Input (CI)		2.0			V
			2.4			V
V <sub>H</sub>	Schmitt hysteresis			0.8		V
I <sub>IL</sub>	Low level input current (I,SI) With Pull-up (PI)	V <sub>IN</sub> = VSS	-10			μA
		VDD = 3.6V	-40			μA
I <sub>IH</sub>	High level input current	V <sub>IN</sub> = VDD			10	μA
		VDD = 3.3V				
V <sub>OL</sub>	Low level output voltage (02) (04) (06) (08)	VDD = 3.0V				
		IOL = 2mA			0.4	V
		IOL = 4mA			0.4	V
		IOL = 6mA			0.4	V
		IOL = 8mA			0.4	V
V <sub>OH</sub>	High level output voltage (02) (04) (06) (08)	VDD = 3.0V				
		IOL = -2mA	2.4			V
		IOL = -4mA	2.4			V
		IOL = -6mA	2.4			V
		IOL = -8mA	2.4			V
I <sub>OZ</sub>	High impedance leakage current	V <sub>O</sub> = VSS VDD = 3.6V	-10			μA
I <sub>DD</sub>	Quiescent supply current				10	μA
C <sub>IN</sub>	Input capacitance	VDD = 3.3V		2.4		pF
C <sub>OUT</sub>	Output capacitance	VDD = 3.3V		5.6		pF
C <sub>I/O</sub>	Input/Output capacitance	VDD = 3.3V		6.6		pF
P <sub>A</sub>	Power dissipation	VDD = 3.3.V Core Logic Clock = 80MHz Bus Clock = 40 MHz		0.7	1.0	W

## 6.2 AC Specifications

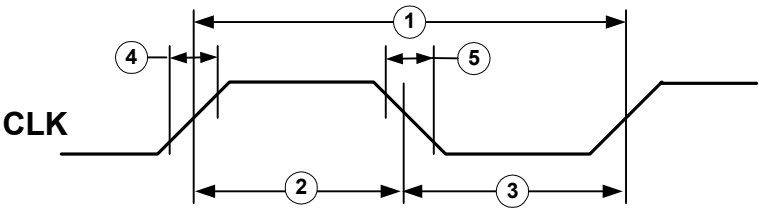
### 6.2.1 Reset and Clock Timing

Table 14: Reset and power-up timing



Number	Description	Min		
1	Reset width	12*CLK		ns
2	First access after Reset (if <b>PLL</b> is already locked)	2*CLK		ns
-	<b>PLL</b> lock time after stable <b>VCC</b>	100		μs

Table 15: CLK timing



Item	Description	Min	Max	Units
	Oscillator frequency		50	MHz
1	Clock period	20		ns
2	Clock width high	8		ns
3	Clock width low	8		ns
4	Clock rise time from $V_{IL}$ to $V_{IH}$		6.5	ns
5	Clock fall time from $V_{IH}$ to $V_{IL}$		6.5	ns

Table 16: CPU timing

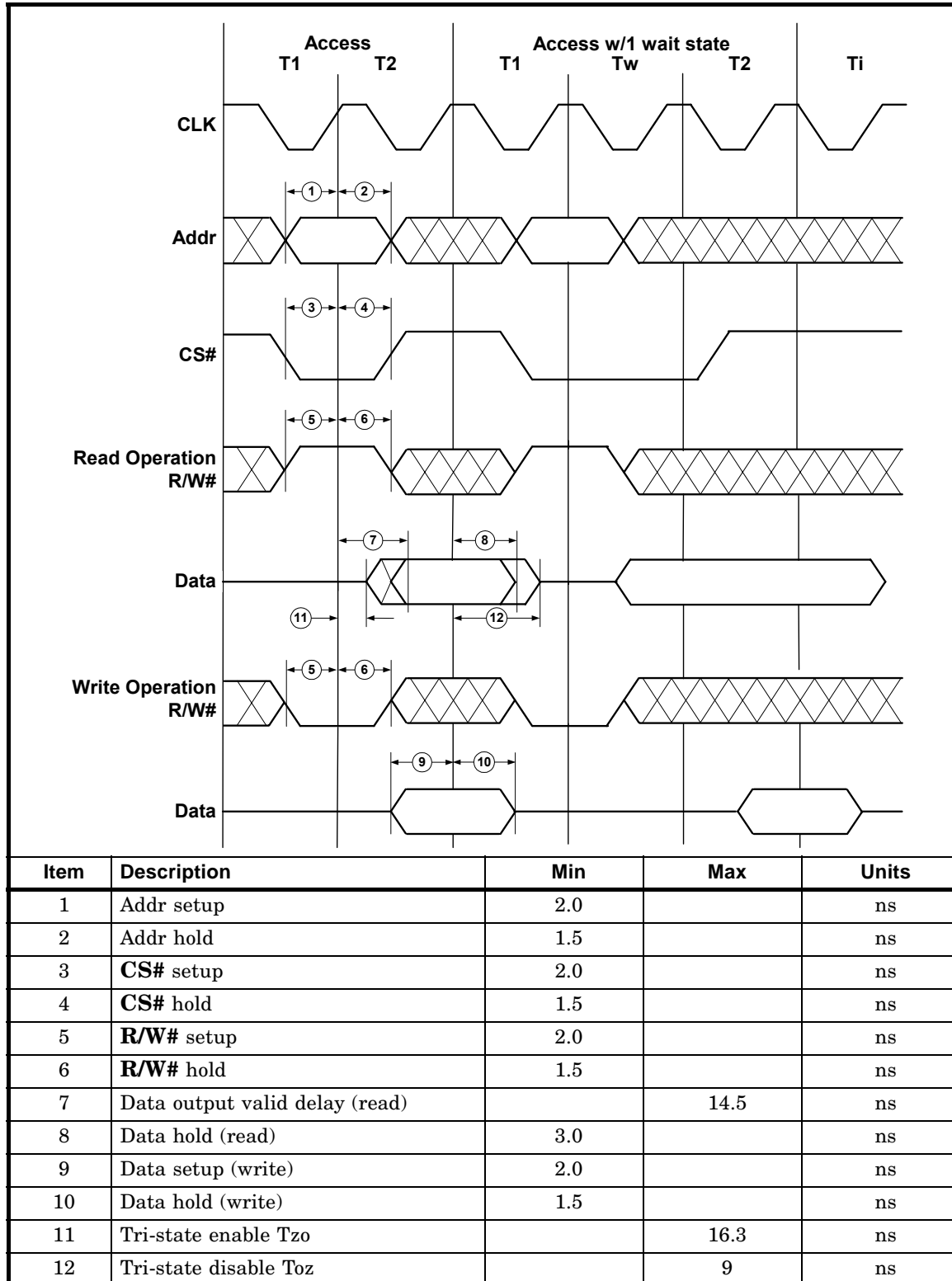
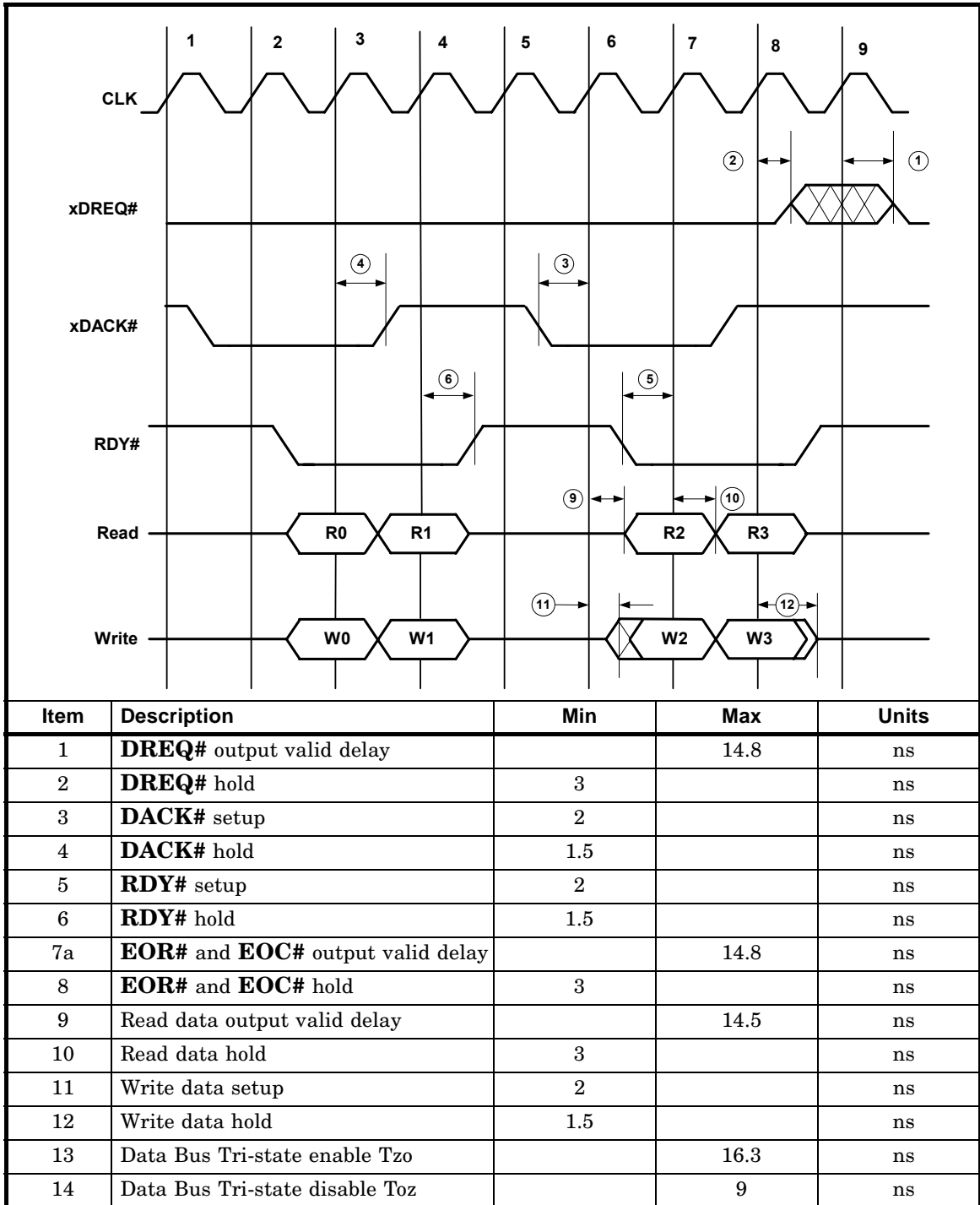


Table 17: DMA timing



## 6.3 Pin Description

Table 18: Pin Description, numerical order

Pin	Signal	Pin	Signal	Pin	Signal
1	<b>CMDRDY</b>	35	<b>D15</b>	69	<b>IRQ#</b>
2	<b>VSS</b>	36	<b>D14</b>	70	<b>VSS</b>
3	<b>A3</b>	37	<b>DIV1</b>	71	<b>VSS</b>
4	<b>A2</b>	38	<b>VDD</b>	72	<b>VDD</b>
5	<b>A1</b>	39	<b>D13</b>	73	<b>RESET#</b>
6	<b>A0</b>	40	<b>VSS</b>	74	<b>EOR#</b>
7	<b>R/W#</b>	41	<b>VDD</b>	75	<b>EOC#</b>
8	<b>CS#</b>	42	<b>D12</b>	76	NC
9	<b>VSS</b>	43	<b>D11</b>	77	NC
10	<b>D31</b>	44	<b>D10</b>	78	NC
11	<b>D30</b>	45	<b>DIV0</b>	79	NC
12	<b>D29</b>	46	<b>VDD</b>	80	<b>VSS</b>
13	<b>VDD</b>	47	<b>RDY#</b>	81	<b>CLK</b>
14	<b>D28</b>	48	<b>D9</b>	82	NC
15	<b>VSS</b>	49	<b>VDD</b>	83	<b>VDD</b>
16	<b>D27</b>	50	<b>D8</b>	84	<b>VDD</b>
17	<b>D26</b>	51	<b>VSS</b>	85	<b>VDD</b>
18	<b>D25</b>	52	<b>D7</b>	86	<b>VDD</b>
19	<b>D24</b>	53	<b>D6</b>	87	NC
20	<b>D23</b>	54	<b>D5</b>	88	<b>VSS</b>
21	<b>D22</b>	55	<b>D4</b>	89	<b>VSS</b>
22	<b>VDD</b>	56	<b>D3</b>	90	<b>VSS</b>
23	<b>D21</b>	57	<b>D2</b>	91	<b>VDD</b>
24	<b>SDREQ1#</b>	58	<b>VSS</b>	92	NC
25	<b>D20</b>	59	<b>D1</b>	93	<b>VDD</b>
26	<b>VSS</b>	60	<b>VDD</b>	94	NC
27	<b>D19</b>	61	<b>D0</b>	95	<b>P3</b>
28	<b>D18</b>	62	<b>VDD</b>	96	<b>P2</b>
29	<b>D17</b>	63	<b>VSS</b>	97	<b>P1</b>
30	<b>SDACK1#</b>	64	<b>B3</b>	98	<b>P0</b>
31	<b>DDREQ1#</b>	65	<b>B2</b>	99	<b>VSS</b>
32	<b>VDD</b>	66	<b>VSS</b>	100	NC
33	<b>DDACK1#</b>	67	<b>B1</b>		
34	<b>D16</b>	68	<b>B0</b>		

**Table 19: Pin Description, alphabetical order**

Signal	Pin	Signal	Pin	Signal	Pin
<b>A0</b>	6	<b>D23</b>	20	<b>VDD</b>	13
<b>A1</b>	5	<b>D24</b>	19	<b>VDD</b>	22
<b>A2</b>	4	<b>D25</b>	18	<b>VDD</b>	32
<b>A3</b>	3	<b>D26</b>	17	<b>VDD</b>	38
<b>B0</b>	68	<b>D27</b>	16	<b>VDD</b>	41
<b>B1</b>	67	<b>D28</b>	14	<b>VDD</b>	46
<b>B2</b>	65	<b>D29</b>	12	<b>VDD</b>	49
<b>B3</b>	64	<b>D30</b>	11	<b>VDD</b>	60
<b>CLK</b>	81	<b>D31</b>	10	<b>VDD</b>	62
<b>CMDRDY</b>	1	<b>DDACK1#</b>	33	<b>VDD</b>	72
<b>CS#</b>	8	<b>DDREQ1#</b>	31	<b>VDD</b>	83
<b>D00</b>	61	<b>DIV0</b>	45	<b>VDD</b>	84
<b>D01</b>	59	<b>DIV1</b>	37	<b>VDD</b>	85
<b>D02</b>	57	<b>EOC#</b>	75	<b>VDD</b>	86
<b>D03</b>	56	<b>EOR#</b>	74	<b>VDD</b>	91
<b>D04</b>	55	<b>IRQ#</b>	69	<b>VDD</b>	93
<b>D05</b>	54	NC	76	<b>VSS</b>	2
<b>D06</b>	53	NC	77	<b>VSS</b>	9
<b>D07</b>	52	NC	78	<b>VSS</b>	15
<b>D08</b>	50	NC	79	<b>VSS</b>	26
<b>D09</b>	48	NC	82	<b>VSS</b>	40
<b>D10</b>	44	NC	87	<b>VSS</b>	51
<b>D11</b>	43	NC	92	<b>VSS</b>	58
<b>D12</b>	42	NC	94	<b>VSS</b>	63
<b>D13</b>	39	NC	100	<b>VSS</b>	66
<b>D14</b>	36	<b>P0</b>	98	<b>VSS</b>	70
<b>D15</b>	35	<b>P1</b>	97	<b>VSS</b>	71
<b>D16</b>	34	<b>P2</b>	96	<b>VSS</b>	80
<b>D17</b>	29	<b>P3</b>	95	<b>VSS</b>	88
<b>D18</b>	28	<b>R/W#</b>	7	<b>VSS</b>	89
<b>D19</b>	27	<b>RDY#</b>	47	<b>VSS</b>	90
<b>D20</b>	25	<b>RESET#</b>	73	<b>VSS</b>	99
<b>D21</b>	23	<b>SDACK1#</b>	30		
<b>D22</b>	21	<b>SDREQ1#</b>	24		

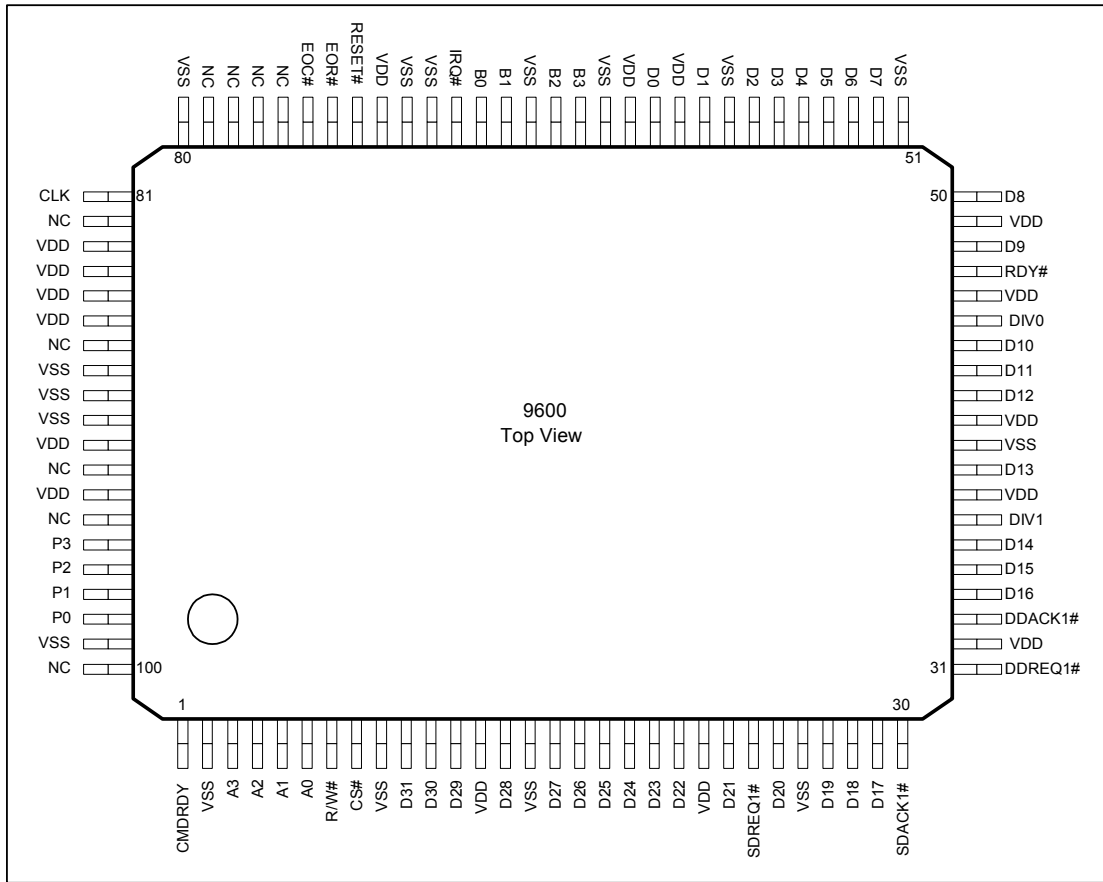


Figure 24. Pinout diagram



## 6.4 Package Dimensions

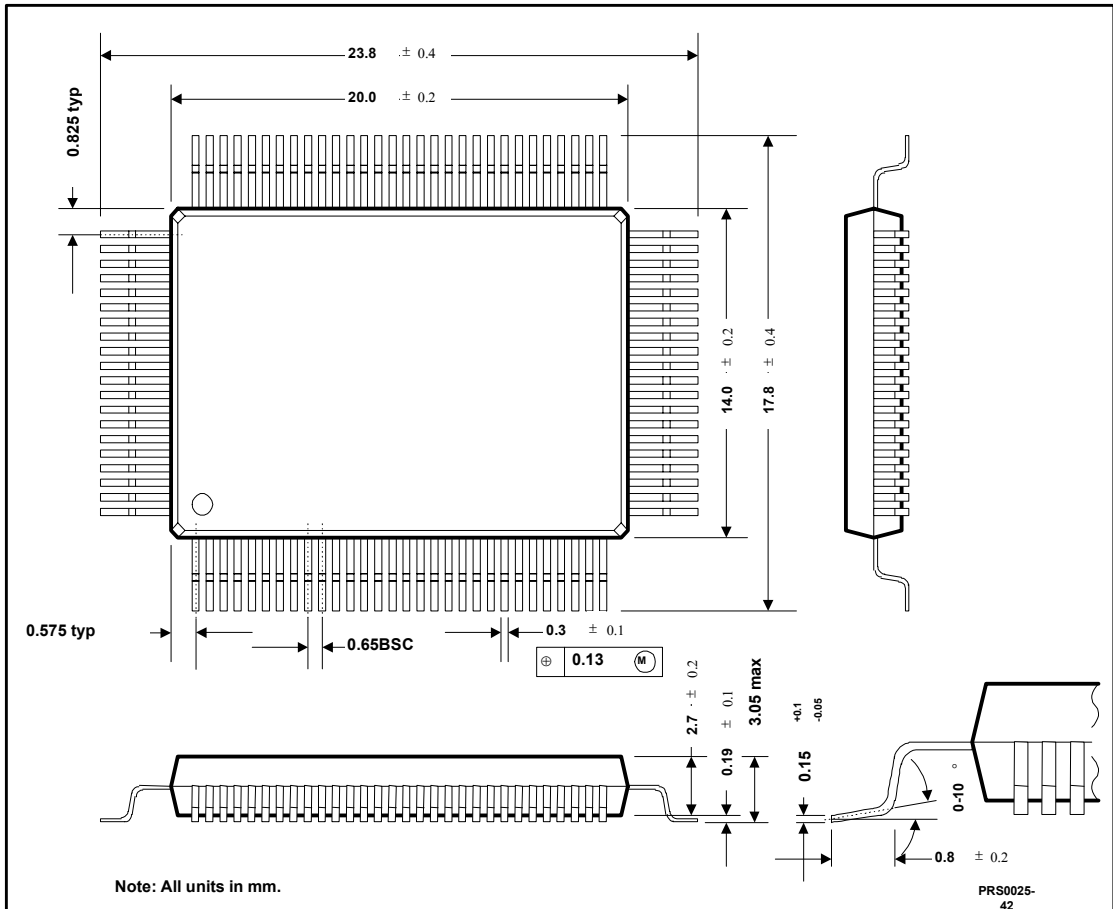


Figure 25. 100-pin PQFP package dimensions



750 University Avenue  
Los Gatos, CA 95032  
tel: 408.399.3500  
fax: 408.399.3501  
web: [www.hifn.com](http://www.hifn.com)