DATA SHEET

9602

Data Compression Processor

hifn

Hi/fn™ supplies two of the Internet's most important raw materials: compression and encryption. Hi/fn is also the world's first company to put both on a single chip, creating a processor that performs compression and encryption at a faster speed than a conventional CPU alone could handle, and for much less than the cost of a Pentium or comparable processor.

As of October 1, 1998, our address is:

Hi/fn, Inc.
750 University Avenue
Los Gatos, CA 95032
info@hifn.com
http://www.hifn.com
Tel: 408-399-3500
Fax: 408-399-3501

Hi/fn Applications Support Hotline:
408-399-3544

Disclaimer

Hi/fn reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

Hi/fn warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with Hi/fn's standard warranty. Testing and other quality control techniques are utilized to the extent Hi/fn deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

HI/FN SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of Hi/fn 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 Hi/fn 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.

Hi/fn 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 Hi/fn 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.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals," must be validated for each customer application by customer's technical experts.

Table of Contents

Figures

1 Product Description

The Hi/fn 9602 is a lossless data compression processor. It uses the industry-standard ALDC (adaptive lossless data compression) algorithm. The 9602 is a dual-channel device that can perform compression and decompression simultaneously. Both channels in the 9602 operate at up to 80MB/s, for a total performance of up to 160MB/s.

Each channel contains input and output FIFOs, a compression/decompression engine, dedicated internal compression memory, and command, status, and result registers. Both channels connect to the shared system bus.

The 9602's 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 bus transfers at a bus clock of up to 66 MHz. The bus interface is straightforward and easy to interface to.

Features

- Dual-channel compressor/decompressor
- 80 MB/s per channel for either compression or decompression, for a total of 160 MB/s
- Industry-standard ALDC-1 algorithm
- Supports simultaneous full-speed compression and decompression
- On-chip memory eliminates the need for external compression RAM
- Processes multiple data records per command
- High-speed, single-cycle burst interface

Part Number	Package
9602 PT4	100-pin PQFP

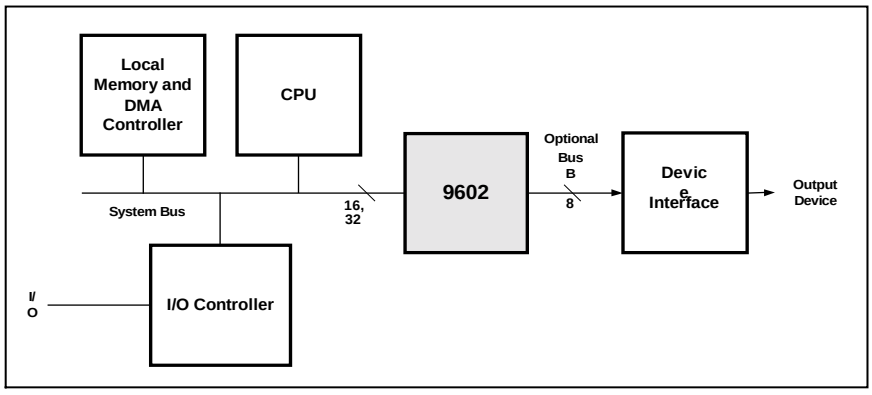


Figure 1. System Block Diagram

2 Product Overview

The 9602 is a dual-channel device with two bus interfaces. Each channel contains input and output FIFOs, a compression/decompression engine, dedicated internal compression memory, and command, status, and result registers. Channel 1 performs both compression and decompression. Channel 2 performs decompression only. The two are otherwise identical.

Figure 2 shows an internal block diagram of the 9602. Each channel 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 both channels to operate at full speed over the I/O bus. Each channel also has its own DMA handshaking signals to control the movement of data into and out of the FIFOs.

The 9602's 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 66 MHz bus speed. The output of Channel 2 can be sent to the dedicated eight-bit output port, Bus B, which is independent of the system bus interface. Bus B uses the Channel 2 Dest FIFO for data buffering. It is clocked externally, and can provide a decompressed data stream at bus speeds of up to 50 MHz. Bus handshaking for the Bus B is identical to that of the system bus' DMA interface.

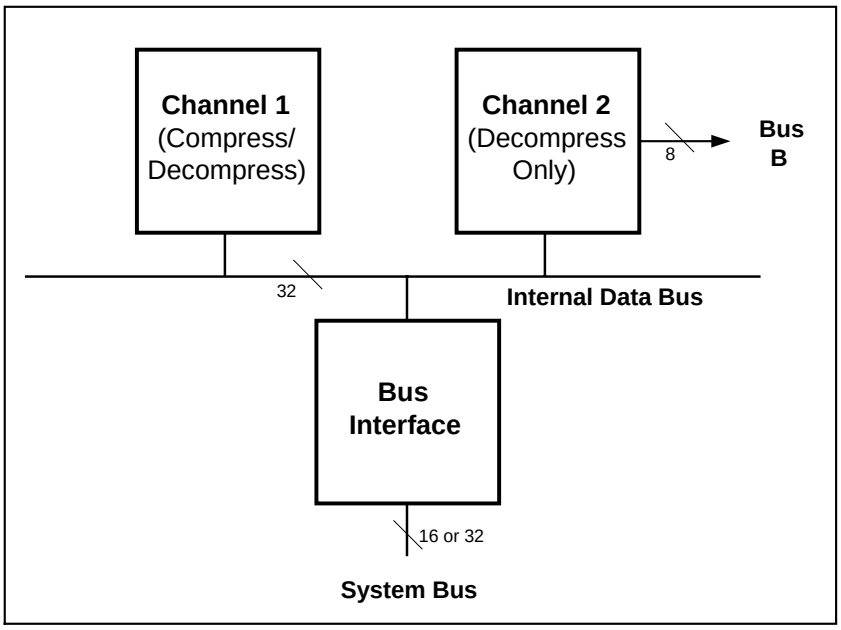


Figure 2. Internal block diagram

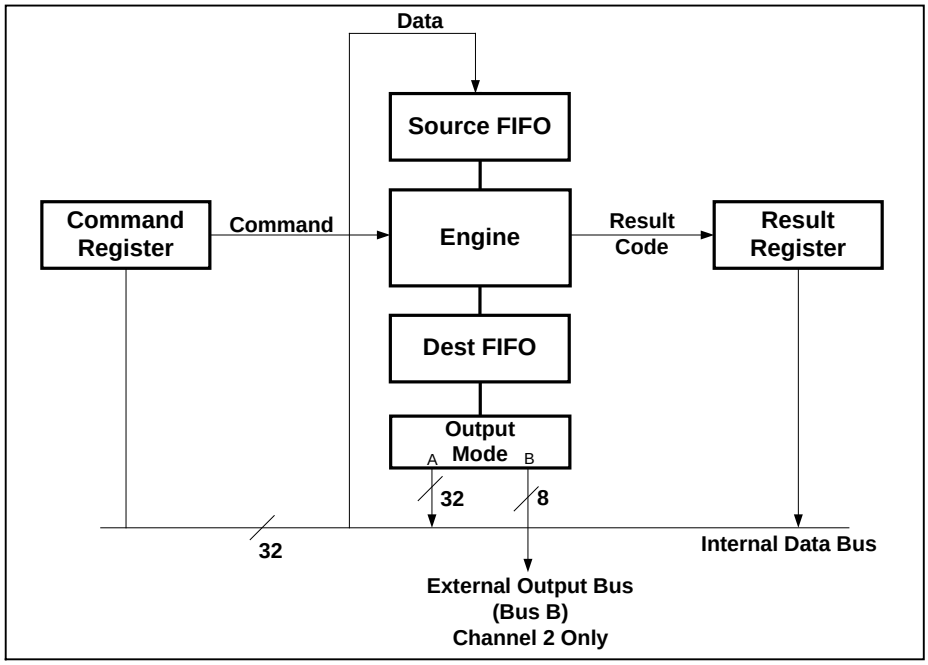


Figure 3. Channel architecture

Figure 3 shows command and data flow in one of the Channels. Note that, 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.

An external DMA controller uses the 9602 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.

2.1 Detailed Block Diagram

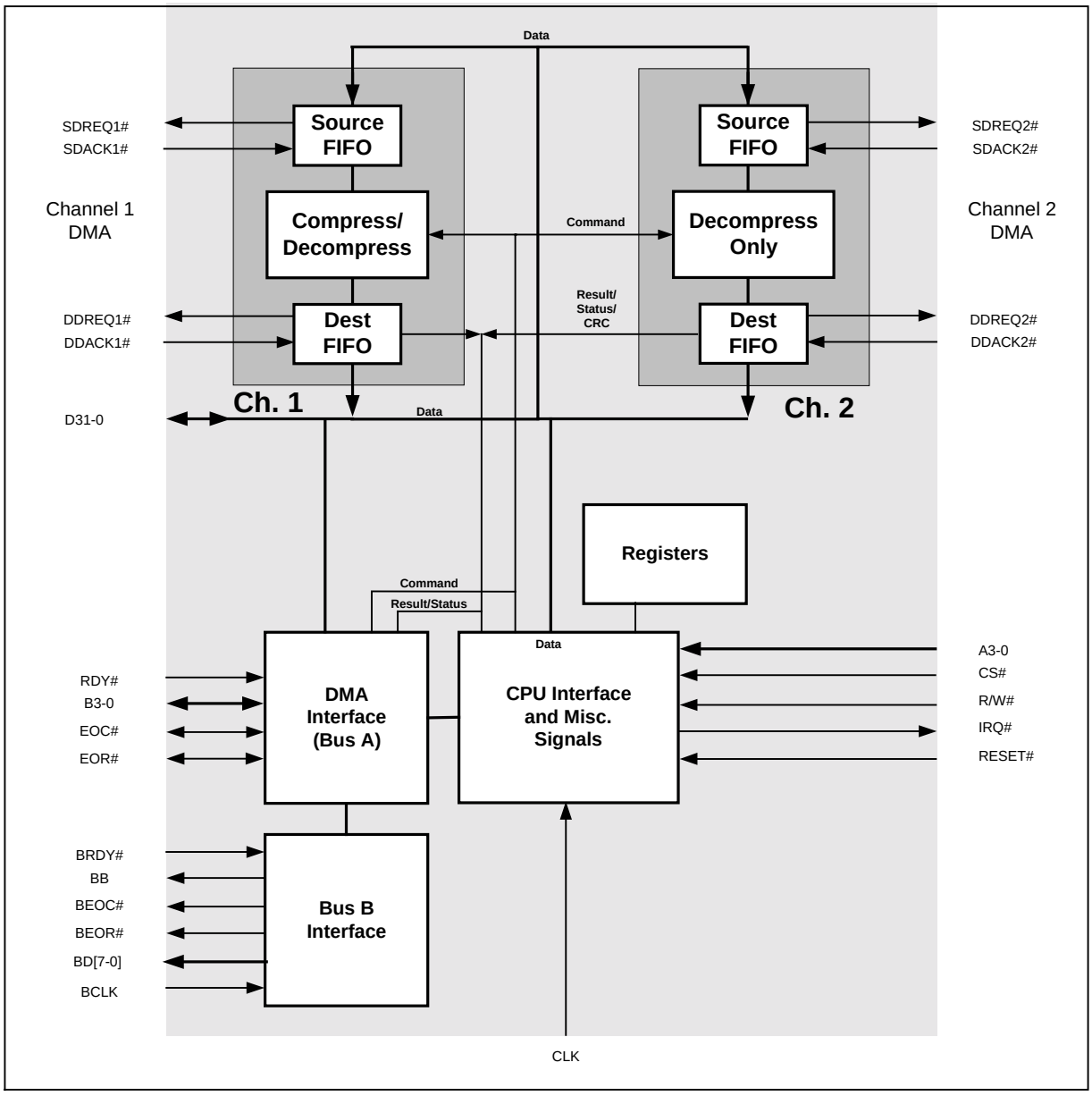


Figure 4. Detailed Block Diagram

2.2 Signal Summary

The following table summarizes the 9602's signals. For a complete signal description, see Section 5. See Figure 34 in section 7.6 for a pin description sorted by pin number, and Figure 35 for a pin description sorted by signal name.

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

DMA Interface	Signal	Type	Description
Common to both channels	B[3-0]	I/O	Byte Enable
	RDY#	Input	DMA Ready
	EOC#	I/O	End of Command
	EOR#	I/O	End of Record
Channel 1	SDREQ1#	Output*	Channel 1 DMA Input Request (Source FIFO)
	SDACK1#	Input	Channel 1 DMA Input Acknowledge
	DDREQ1#	Output*	Channel 1 DMA Output Request (Dest FIFO)
	DDACK1#	Input	Channel 1 DMA Output Acknowledge
Channel2	SDREQ2#	Output*	Channel 2 DMA Input Request (Source FIFO)
	SDACK2#	Input	Channel 2 DMA Input Acknowledge
	DDREQ2#	Output*	Channel 2 DMA Output Request (Dest FIFO)
	DDACK2#	Input	Channel 2 DMA Output Acknowledge

Bus B Signals	Signal	Type	Description
	BB	Output	Bus B byte enable
	BCLK	Input	Bus B bus clock
	BEOC#	Output	Bus B end of command
	BEOR#	Output	Bus B end of record
	BRDY#	Input	Bus B DMA ready
	BD[7-0]	Output	Bus B data output

	Signal	Type	Description
Miscellaneous Signals	RESET#	Input	Hardware Reset (Schmitt input)
	CLK	Input	Clock Input
	VCC	Input	+3.3 Volt supply
	GND	Input	Ground

* External pull-up/pull-down resistors on these signals set the PLL clock multiple on reset. See section 3.14.

Figure 5. Signal summary

2.3 Register Summary

Name	Address		Description
	Channel 1	Channel 2	
Data	0	8	32-bit data register. Writing to this register adds data to the Source FIFO. Reading removes data from the Dest FIFO.
Command	1	9	3x32-bit Command register. Contains Command word, 32-bit Source Count, and 32-bit Record Count. Successive reads or writes access the three words in sequence.
Result	2	10	5x32-bit Result 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.
Configuration	3		16-bit Configuration register. Sets device configuration, including Device Mode, Bus Width, Big-Endian, and Software Reset fields.
Interrupt Enable	4	12	16-bit Interrupt Enable register. Allows interrupts to be enabled on a variety of conditions
Status	5	13	16-bit R/W Status register. Reflects the state of each channel.
Reserved	6	14	Reserved for future expansion. Do not write to this register.
FIFO Configuration	7	15	16-bit FIFO Configuration 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.
Chip ID	11		16-bit read-only Chip ID.

See section 4 for a complete register description.

Figure 6. Register Summary

3 Operation

3.1 Terminology

3.1.1 Command Terminology

- *Command.* A single instruction to the 9602.
- *EOC.* The end-of-command condition. This can be indicated in a variety of ways, including with the EOC# signal and by the use of the RECORD COUNT fields to specify the amount of data to process.

3.1.2 Record Terminology

- *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 EOR.
- *Data stream.* Any series of input or output bytes.
- *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.

- *EOR.* The end-of-record condition, which can be indicated by the EOR# signal or, for fixed-length records, by the use of the SOURCE COUNT field in the Command 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 9602 automatically.
- *Raw data stream.* A series of uncompressed bytes.
- *Raw record.* A series of uncompressed bytes with the end-of-record condition indicated externally.
- *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.
- *Record.* A portion of a data stream with a definite start and end. Compression and decompression operate on a record-by-record basis.

3.1.3 Data Transfer Terminology

- *DMA.* Transfers using the 9602'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.
- *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.
- *Programmed I/O.* Transfers using the register access protocol. This method can access registers and the FIFOs.

3.2 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 indicator. During compression, the 9602 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 9602 takes a compressed data stream and produces a series of raw records, reconstructing the original raw data stream.

3.3 Commands

A command is a single 9602 instruction. A command can process any number of records. A command is launched when a command structure is written to a Command register. Each Channel has its own 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 4.3 for a description of the Command register and its bit fields.

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

The concept of a command is specific to the 9602; 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.

3.4 Command and Record Termination

3.4.1 Command Termination

Commands can be terminated as follows:

During compression:

- 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).

During decompression:

- 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 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 Dest FIFO, the 9602 asserts the EOC# signal on the last output transfer containing valid data, or on a dummy transfer following the last data transfer. The Dest FIFO then goes idle. After the burst containing EOC, the Dest 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 zero, and vice versa.

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

See section 3.10.4 for details on requirements for data alignment, padding, and dummy bytes during EOC.

3.4.2 Record Termination

Like EOC, the end-of-record 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.

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 9602 on the transfer, from the Dest FIFO, of the last byte of data in the record.

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

On input, the compressed data stream must consist of a multiple of four valid bytes. The 9602 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 Dest 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.

Alignment and Padding
See section 3.10.4 for details on requirements for data alignment, padding, and dummy bytes during EOR.

3.4.3 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.

- The 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.
- The 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.
- The Dest 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 zero 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's Status register is set, or
- EOC# has been asserted by the 9602 during an output transfer from the Channel's Dest FIFO.

3.5 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 3.10.1. Data can also be transferred through the programmed I/O mechanism. See section 3.10.2.

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 7 shows a command that consists of five input records. Records 1, 3, and 4 are terminated when the Source Count reaches zero. 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.

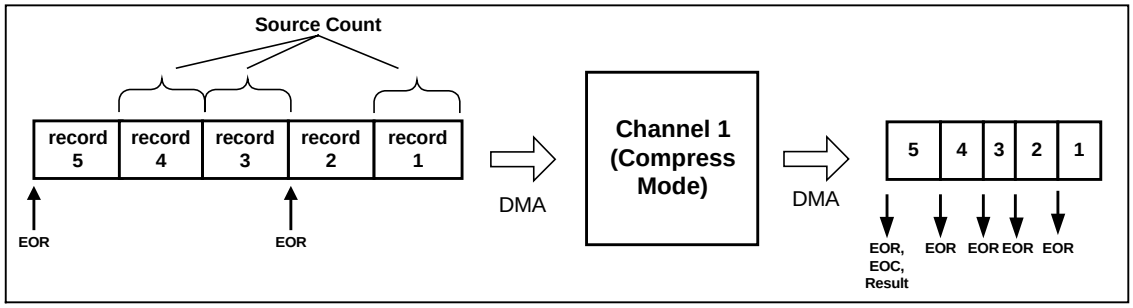


Figure 7. Compression example

On output, end-of-record 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 3.10.4.

The RECORD COUNT field decrements with each record. In the example in Figure 8, 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.

3.5.1 Compression, Step by Step

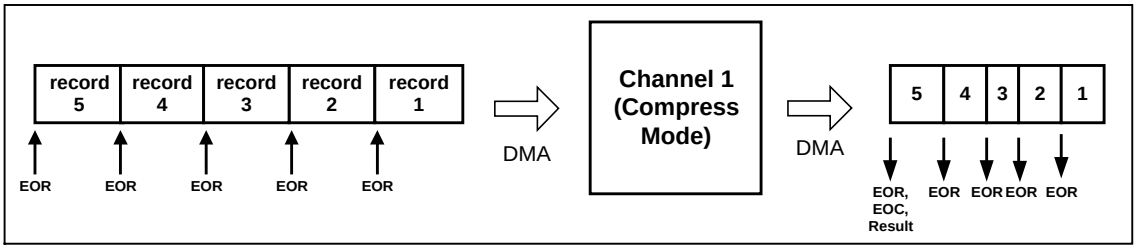


Figure 8. Compression example, fixed-length records

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 8. Compression is only available on Channel 1. (Details of register usage, system bus timing, and signal definitions are given in sections 4, 6.2, and 5.)

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 9602 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 Channel 1's 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.

3.6 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.

3.7 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 3.10.4 for details of alignment and padding requirements for decompression.

3.7.1 Decompression, Step by Step

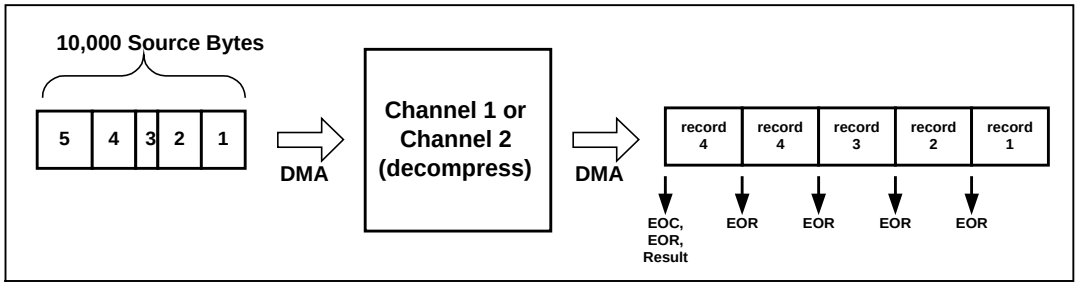


Figure 9. Decompression example

The example in Figure 9 assumes that we have chosen to decompress a block of 10,000 compressed bytes. (The register numbers given below are for Channel 1. Channel 2 operation is identical, except for the registers used.)

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 RECORD 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 end of record 4. If we wished to decompress record 5, the CONSUMED BYTE COUNT field would give the offset of the first byte of record 5.

3.8 Error Handling

The Status register contains error bits for COMMAND/RESULT OVERRUN, DATA ERROR, 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 Dest 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.

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 one to the RESET bit of the Configuration register.

3.9 CRC/LCB Calculation

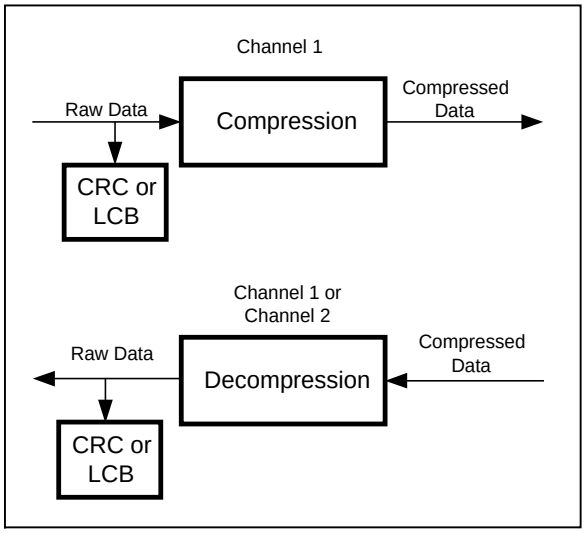


Figure 10. CRC/LCB calculation in compression and decompression.

As shown in Figure 10, the 9602 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.

3.9.1 CRC/LCB Algorithms

In all three algorithms, the initial value is all one's (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.

3.9.2 Protected Records

The 9602 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 9602 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 9602 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.

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 9602.

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 9602 treats it like any other data. The CRC is embedded by the host; the 9602 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 9602, it will return a CRC value of 0xDEBB20E3. This value is constant for every protected record with valid 32-bit CRC bytes.

3.10 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 Dest FIFO. There are two mechanisms for transferring data: programmed I/O and DMA. DMA provides higher performance.

Both methods use the FIFOs in an identical manner. Since the programmed I/O method does not use the end-of-record, end-of-command, 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.

3.10.1 DMA

The most efficient way to use the 9602 is through its DMA interface. The DMA interface allows data to move directly between the host's DMA controller and the 9602's FIFOs. Each of the two Channels has two FIFOs: a Source FIFO and a Dest FIFO. Each FIFO has its own a request and acknowledge signals. Thus, Channel 1 has SDREQ1#, SDACK1#, DDREQ1#, and DDACK1#, while Channel 2 has SDREQ2#, SDACK2#, DDREQ2#, and DDACK2#. 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# (end-of-command) EOR# (end-of-record), D[31:0] data bus, and B[3:0] byte enables.

Once a command is issued, DMA transfers can begin. The Source FIFOs are always empty at the beginning of a command, so the SDREQ[1:2]# signals will be asserted immediately. Data to be processed is written by the host to one of the Source FIFOs. The host selects the FIFO to read from or write to and initiates a burst by asserting the desired SDACK# signal.

The SDREQ[1:2]# and DDREQ[1:2]# 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 3.12.

The end-of-record and end-of-command 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 9602.

Data sent to the Channel after end-of-record is considered to be part of the next record (see the discussion of data alignment requirements in section 3.10.4).

Any data sent to the Channel after end-of-command 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 9602 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 4.4.4for details of the CONSUMED BYTE COUNT.)

DMA timing is covered in detail in section 6.2.

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 9602 will discard invalid (dummy) bytes and process those marked valid. When reading from the Dest 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 9602.

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 end-of-record or end-of-command 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 in section 3.12.1.

When reading from the Dest FIFO, the byte-enable signals are written by the 9602 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 end of command or 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 6.2 for a detailed description of the DMA interface protocol.

3.10.2 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 6.1 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's 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 9602. Writes to the Data register send data into the Source FIFO. Reads from the Data register return data from the Dest 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], DDREQ[2-1]#, EOC#, EOR#, and SDREQ[2-1]#. Bits to emulate these signals exist in the Status register. These signals are used to indicate valid bytes, end of records 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's 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 9602. Otherwise, errors, EOC and EOR status, and FIFO flags will be ignored.

See section 3.12.1 for an example of programmed I/O transfers.

3.10.3 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).

3.10.4 Data Alignment Requirements

The 9602 has different alignment requirements depending on what operation is being performed. When using the EOR# and EOC# signals to indicate end-of-record and end-of-command, 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 end-of-record or end-of-

command 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 9602, 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 end of records are in the data stream.

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

Requirements for Writing to the Source FIFO

Start of Record Alignment	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 'End-of-Record Alignment,' below.)
End-of-Record Alignment	Compression	If the EOR# 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 EOR# 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 EOR# signal is ignored on input during decompression.
Start of Command Alignment	Compression and Decompression	No restrictions.
End of Command Alignment	Compression and Decompression	No restrictions. Data after EOC is discarded.
EOR/EOC Location	Compression and Decompression	EOR# and EOC# 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.

Figure 11. Writing to the Source FIFO

Characteristics When Reading From the Dest FIFO

The 9602 never produces dummy bytes except at the end of a record or command. All bytes between the start of a record and EOR are valid.

Start of Record Characteristics	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.
End-of-Record Characteristics:	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.
Start of Command Characteristics	Compression and Decompression	Commands are 32-bit aligned on output. They always start on a new bus transfer.
End of Command Characteristics	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.
EOR/EOC Location	Compression	EOR# and EOC# are attached to the transfer containing the last valid data byte
	Decompression	EOR# and EOC# are part of a dummy transfer that occurs after the chip writes that last valid data.

Figure 12. Reading from the Dest FIFO

3.11 FIFO Data Flow

3.11.1 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.

3.11.2 Dest FIFO Data Flow

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

3.12 FIFO Thresholds

There are four FIFO Threshold values for each engine: *Source FIFO Upper Threshold, Source FIFO Lower Threshold, Dest FIFO Upper Threshold,* and *Dest 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 SDREQ# 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 SDREQ# 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 DDREQ# signal (and the DDREQ bit in the Status register) will be asserted when the number of bytes in the Dest FIFO is greater than the DEST FIFO UPPER THRESHOLD.

The DDREQ# signal (and the DDREQ bit in the Status register) will be deasserted when the number of bytes in the Dest FIFO is less than or equal to the DEST FIFO LOWER THRESHOLD.

If the thresholds are set below 1 for a 16-bit bus, or 3 for a 32-bit bus, the 9602 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 Dest 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 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 the SOURCE FIFO UPPER THRESHOLD to 15 will guarantee that at least 16 bytes of empty space are in the Source FIFO when SDREQ# is asserted. Setting the SOURCE FIFO LOWER THRESHOLD to 15 will guarantee that the SDREQ# signal will be deasserted when the number of bytes of empty space in the Source FIFO is 15 or less.

Similarly, setting the DEST FIFO UPPER THRESHOLD to 15 will guarantee that at least 16 bytes of data are in the Dest FIFO before DDREQ# will be asserted. Setting the DEST FIFO LOWER THRESHOLD to 15 will ensure that the DDREQ# signal will be deasserted if there are 15 or less bytes in the Dest 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 SDREQ# is deasserted. Similarly, setting the DEST FIFO LOWER THRESHOLD to seven

will guarantee that the Dest FIFO will be able to provide two additional 32-bit transfers after the DDREQ# signal is deasserted.

With fixed-length bursts, the FIFO Lower Threshold can be used to give the DMA controller advance warning of the FIFO's 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 FIFO LOWER THRESHOLD were set to 19 rather than 15. This can reduce dead time between back-to-back bursts.

3.12.1 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 Dest FIFO will cause a fatal error. The following pseudo-code 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 */
```

3.13 Output Bus B

Channel 2 has a dedicated eight-bit output port, Bus B. Channel 2's output can be sent either to the 16/32-bit system bus or to Bus B. Bus B is used in applications where a separate port for the decompressed data stream is useful, such as laser printers.

The output mode is set in the Bus B Enable bit of the Configuration register. Setting this bit to one enables Mode B.

Bus B has its own DMA signals: BD[7:0], BB, BRDY#, BEOC#, BEOR#, and BCLK. These are analogous to D[31:0], B[3:0], RDY#, EOC#, EOR#, and CLK, respectively. If Bus B is enabled, the Channel 2 Dest FIFO's output is redirected to Bus B. The Dest FIFO threshold mechanism is unchanged by Bus B operatoin. Dest FIFO handshaking uses the DDREQ2# and DDACK2# pins as usual. In Bus B mode, all these signals operate in the BCLK clock domain. The bus protocol is identical to the read protocol of system bus, except that AC timing is specified separately, the bus is only eight bits wide (with one byte-enable), and is output-only.

The BCLK input drives Bus B directly. There is no PLL on this input. The maximum frequency for Bus B is 50 MHz.

Because Bus B operates in its own clock domain, the state of the output signals is not reflected accurately in the Status register, so reads from the Channel 2 DDREQ, EOR, EOC, and B bit fields are all invalid when Bus B is enabled (writes to these bits affect the system-bus side of the Channel, not Bus B, and are still valid). The Channel 2 Dest FIFO is not available to programmed I/O when Bus B is enabled.

3.14 Clock

The 9602's internal logic (except for Bus B) 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 section 4). The frequency multiple is controlled by 10 kΩ pull-up or pull-down resistors on four pins, which are sampled after each hardware reset. Their values are shown below:

SDREQ1#	DDREQ1#	SDREQ2#	DDREQ2#	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				Reserved

Figure 13. Clock multiplier selection

A "1" in the table indicates a pull-up resistor. A "0" indicates a pull-down. The SDREQ# and DDREQ# 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 SDREQ# or DDREQ# are being asserted, the host should not sample these outputs for three cycles after RESET# is deasserted. See section 7.2.1.

Clock Multiplier Examples

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
66 MHz	66 MHz	1

Figure 14. 9602 Clock multiplier example

4 Register Descriptions

4.1 Register Overview

The device's register list is shown below. Each Channel has its own set of registers, except for the shared Configuration and Chip ID registers.

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

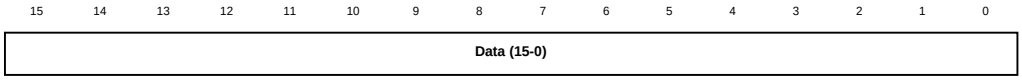
In the register descriptions that follow, some bits will be marked “Reserved.” Reserved bits must be written as zeros and ignored when read.

Name	Address	
	Channel 1	Channel 2
Data	0	8
Command	1	9
Result	2	10
Configuration	3	
Interrupt Enable	4	12
Status	5	13
Reserved	6	14
FIFO Configuration	7	15
Chip ID	11	

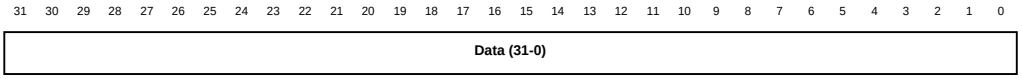
Figure 15. Register list

4.2 Data (0,8)

16-bit Bus Mode



32-bit Bus Mode



Register 0 is the Data register for Channel 1. Register 8 is the Data register for Channel 2.

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.

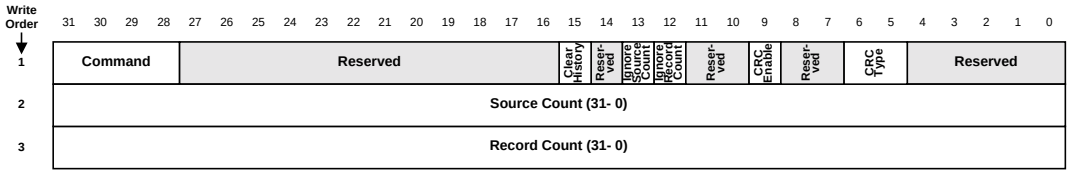
Transfers to or from the Data register can only take place under the same conditions that a DMA transfer can take place. That is, the state of the FIFO must be taken into consideration. The Status register's SDREQ and DDREQ bits reflect the state of the SDREQ# and DDREQ# signals. The Data register should 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 sections 3.10.2 and 3.12.1.

For example, to signal the end of record 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.
When operating in Mode B, the Channel 2 Data register is undefined on reads. It has its normal function on writes.

4.3 Command (1,9)

32-bit Bus Mode



16-bit Bus Mode

Write Order	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Command				Reserved											
2	Clear History	Reser-ved	Ignore Source Count	Ignore Record Count	Reserved		CRC Enable	Reserved		CRC Type		Reserved				
3	Source Count (31-16)															
4	Source Count (15-0)															
5	Record Count (31-16)															
6	Record Count (15-0)															

Register 1 is the Command register for Channel 1. Register 9 is the Command register for Channel 2.

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 PROGREESS 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.

4.3.1 Commands

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

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

Figure 16. Commands

Compress
This channel will compress data. Note that Channel 2 cannot compress data. See section 3.5 for details on how the Compress command operates.

Decompress
The channel will decompress data. See section 3.7 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.

4.3.2 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 zero. On compression, the compression history is cleared between all records if this bit is set to one.

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 zero, 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 zero, 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 zero, the CRC will not be verified against the constant value for a protected record (0xDEBB20E3). If this bit is set to one, 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 3.9.2.

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

Value	Check Type
00	32-bit CRC
01	16-bit CRC
10	8-bit LCB
11	Reserved

See section 3.9 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.

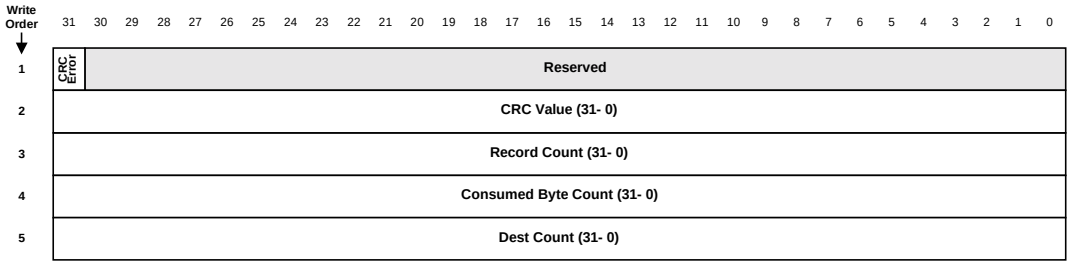
For a Compress Command, the SOURCE COUNT specifies the size of each record to be compressed. Each time the SOURCE COUNT reaches zero (unless the IGNORE SOURCE COUNT bit in the command register is set to one), an end marker will be appended and a new compression operation will begin on the next record.

If the SOURCE COUNT is set to zero, the record length is set to 2^{32} bytes on compression.

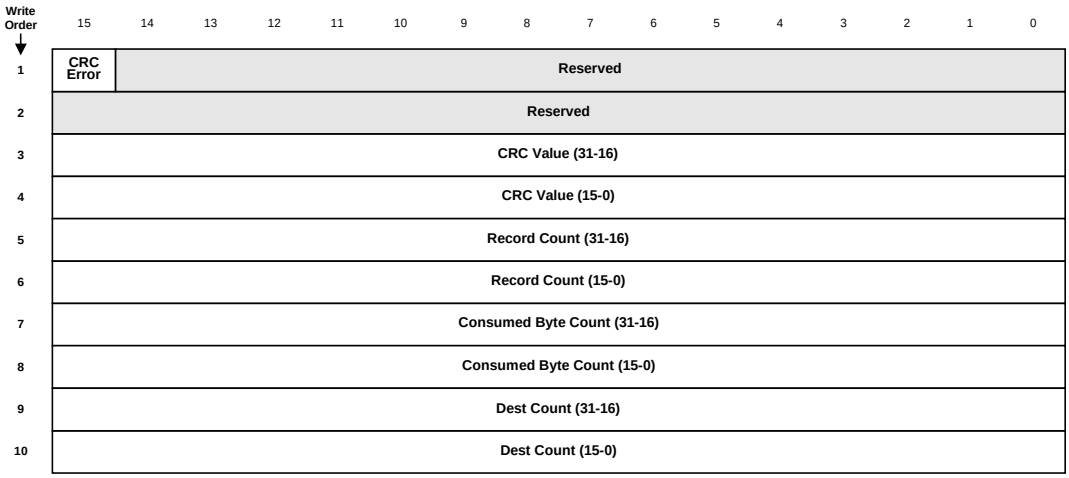
Record Count
This field is the initial value used for the Record Counter. The RECORD COUNT specifies the number of records that will be compressed or decompressed during the command. When the RECORD COUNT reaches zero (unless the IGNORE RECORD COUNT bit in the command register is set to one), the command will terminate. If the RECORD COUNT is set to zero, 2^{32} records will be processed before the command terminates.

4.4 Result (2,10)

32-bit Bus Mode



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 is reset at the beginning of every command, so its contents are only valid between the end of one command and the start of another.

4.4.1 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 one and the 32-bit CRC algorithm is selected. It is used to check the integrity of protected records.

If this bit is set to zero, the CRC in the CRC VALUE field was correct, or the test was not enabled. If this bit is set to one, 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 3.9 for details on when this bit will be set.

4.4.2 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.

4.4.3 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 zero.)

4.4.4 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.

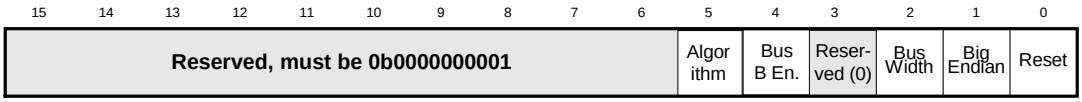
The CONSUMED BYTE COUNT is cleared at the beginning of the command. A temporary register at the input of the Engine (below the Source FIFO) counts the valid bytes in the input data stream. Valid bytes are those bytes whose address enables were asserted. At each EOR, the count in the temporary register 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.

4.4.5 Dest Count

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

4.5 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 zero.

Algorithm.
This bit determines the compression algorithm to use. It is decoded as follows:

Value	Algorithm
0	ALDC-1
1	LZV

Bus B Enable.
When one, this bit enables Bus B output on Channel 2. When zero, Channel 2's output is directed to the System Data Bus.

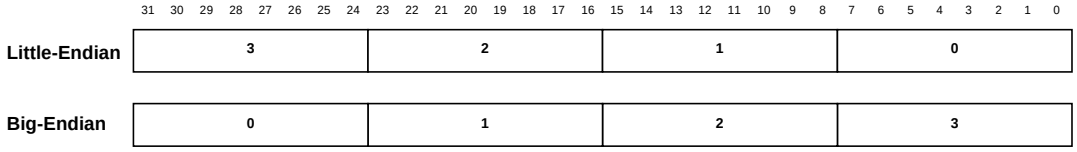
Value	Mode
0	Mode A (Channel 2 output goes to System Data Bus)
1	Mode B (Channel 2 output goes to Bus B)

4.5.1 Bus Width

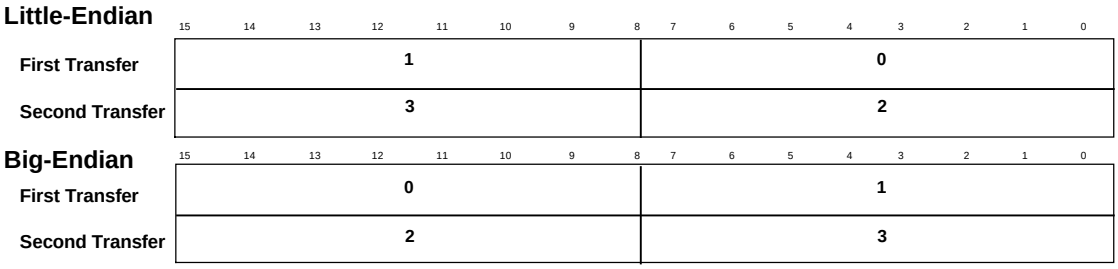
This bit selects the width of the system bus. If this bit is set to one, the bus width will be 32 bits. If this bit is set to zero, the bus width will be 16 bits. This bit is significant only for the data bus and three registers: the Data register, the Command register, and the Result register. All other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

4.5.2 Big Endian

32-bit Bus Mode



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 zero, 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 one, 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.

4.5.3 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 zero. Writing a zero to this bit will not clear the reset, it will clear automatically.

4.6 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 zero.

4.6.1 Command Ready Interrupt Enable

While this bit is set to one, the IRQ# signal will be asserted while the COMMAND READY bit in the Status register is set to one.

4.6.2 SDREQ Interrupt Enable

While this bit is set to one, the IRQ# signal will be asserted while the SDREQ bit in the Status register is set to one.

4.6.3 DDREQ Interrupt Enable

While this bit is set to one, the IRQ# signal will be asserted while the DDREQ bit in the Status register is set to one. This bit must not be set on Channel 2 when it is running in Mode B, as the Dest FIFO status is not available.

4.6.4 Command/Result Overrun Interrupt Enable

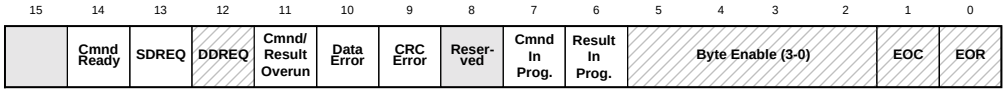
While this bit is set to one, the IRQ# signal will be asserted while the COMMAND/RESULT OVERRUN bit in the Status register is set to one.

4.6.5 Data Error Interrupt Enable

While this bit is set to one, the IRQ# signal will be asserted while the DATA ERROR bit in the Status register is set to one.

4.6.6 CRC Error Interrupt Enable

While this bit is set to one, the IRQ# signal will be asserted while the CRC ERROR bit in the Status register is set to one.



4.7 Status (5,13)

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 Dest FIFO logic.

The error bits – 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.

4.7.1 Command Ready

The read-only COMMAND READY bit is set to one 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 one again until the chip is ready to accept another new command (after the current command has completed).

4.7.2 SDREQ

This read-only bit is set to one 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 zero 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 SDREQ# signal. That is, this bit will be zero when the SDREQ# signal is inactive (high), and one when the SDREQ# signal is active (low).

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

4.7.3 DDREQ

This read-only bit is set to one when the number of bytes in the Dest FIFO is greater than the DEST FIFO UPPER THRESHOLD value programmed in the FIFO Configuration register.

This bit is set to zero when the number of bytes in the Dest FIFO is less than or equal to the DEST FIFO LOWER THRESHOLD value programmed in the FIFO Configuration register.

The DDREQ bit will operate the same as the DDREQ# signal. That is, this bit will be zero when the DDREQ# signal is inactive (high), and one when the DDREQ# signal is active (low).

If CPU I/O is used, the DDREQ# signal is not used. However, the operation of the DDREQ bit remains the same as the DDREQ# signal. The Channel 2 DDREQ bit is reserved when Channel 2 is in Mode B.

4.7.4 Command/Result Overrun

This read-only bit is set to one 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.

4.7.5 Data Error

This read-only bit is set to one if the Source FIFO is written when it is full (overflow), or the Dest 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.

4.7.6 CRC Error

This read-only bit follows the state of the CRC ERROR bit in the Result register. See section 4.4.1.

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

4.7.7 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 one after the first access to the Command register. This bit returns to zero after the last access to the Command register.

4.7.8 Result In Progress

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

4.7.9 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.

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 Figure 17 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.

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 Figure 17 to determine which bits correspond to which bytes. The Channel 2 byte enable fields are reserved on reads when Channel 2 is in Mode B.

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

Figure 17. Byte Enable

4.7.10 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.

Write Transfer
During a write transfer to the Data register, this bit must be set to one 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 zero after each write to the Data register.

Read Transfer
During a read transfer from the Data register, this bit will be set to one if the next read transfer is the last transfer of a command. No new data will be processed until a new command is issued. The Channel 2 EOC bit is reserved on reads when Channel 2 is in Mode B.

4.7.11 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.

Write Transfer
During a write transfer from the Data register, this bit must be set to one 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.

Read Transfer
During a read transfer from the Data register, this bit will be set to one if the next read transfer is the last transfer of a record. Data following the next transfer will be from the next record. The Channel 2 EOR bit is reserved on reads when Channel 2 is in Mode B.

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

4.8 FIFO Configuration (7,15)

Register 7 is the FIFO Configuration register for Channel 1. Register 15 is the FIFO Configuration register for Channel 2.

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 3.12 for more information on FIFO programming.

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

4.8.1 U/L

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

On writes, if this bit is zero, the lower FIFO thresholds for the Source and Dest FIFOs will be accessed. If this bit is set to one, 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.

4.8.2 Source FIFO Threshold

This field sets both the upper and lower source FIFO thresholds. If the U/L bit is set to zero, the lower FIFO threshold will be set. If the UPPER/LOWER bit is set to one, 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 one and the SDREQ# 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 SDREQ# pin will become inactive. See section 3.11.1 and 3.12 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).

4.8.3 Dest FIFO Threshold

This field sets both the upper and lower Dest 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 one, the upper FIFO threshold will be set.

When the number of bytes in the Dest FIFO is greater than the value set in the DEST FIFO UPPER THRESHOLD field, the DDREQ bit in the Status register will be set to one and the DDREQ# pin will become active. When the number of bytes in the Dest FIFO is less than or equal to the value set in the DEST FIFO LOWER THRESHOLD field, the DDREQ bit in the Status register will be set to zero and DDREQ# pin will become inactive. See sections 3.11.2 and 3.12 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.

4.9 Chip ID (11)

This register may only be read. The upper 8 bit 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 0x0AXX.

5 Signal Description

5.1 CPU Interface

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

5.1.1 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.

5.1.2 Address (A[3-0])

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

5.1.3 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.

5.1.4 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.

5.1.5 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 9602 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.

5.2 DMA Interface

This section describes the DMA interface signals on the system bus. Inputs to the 9602 are sampled on the rising edge of CLK. Outputs are valid on the rising edge of CLK. The bus protocol is covered in Section 6.1. AC timing parameters are covered in section 4.

5.2.1 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 Dest FIFO.

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

5.2.2 DMA Ready (RDY#)

Active low input. This signal enables data transfers within the 9602. When asserted, a transfer on the current cycle is completed (written into the Source FIFO or read from the Dest FIFO). When deasserted, the transfer is deferred.

Deasserting RDY# on DMA reads from the Dest FIFOs causes output wait states. The 9602 will drive the same Dest 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 all four FIFOs.

5.2.3 End of Command (EOC#)

Bi-directional end-of-command 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 Dest FIFO, this signal operates as an output.

5.2.4 End of Record (EOR#)

Bi-directional end-of-record 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 Dest FIFO, this signal operates as an output.

5.2.5 Source FIFO DMA Request (SDREQ[1-2]#)

Active low output. SDREQ1# is for Channel 1, and SDREQ2# is for Channel 2. These signals become active when the Source FIFO contains enough free space to accept a DMA burst, as determined by the upper FIFO thresholds. These signals will become inactive when the Source FIFO contains a number of data bytes greater than the source FIFO lower threshold. See section 3.11.1 and 3.12 for more information on FIFO usage. All four DREQ# signals (DDREQ1#, DDREQ2,# SDREQ1#, and SDREQ2#) may be active at the same time. It is the responsibility of the host to choose which FIFO to service.

5.2.6 Source DMA Acknowledge (SDACK[1-2]#)

Active low input. SDACK1# is for Channel 1, and SDACK2# is for Channel 2. The host asserts one of the DDACK# signals to initiate DMA reads from the Dest FIFOs. Only one of the four DACK# signals should be asserted at the same time. These signals select one of the Source FIFOs for input, then writes the data on the next clock cycle. This 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 3.11.1 and 3.12 for more information on FIFO usage.

5.2.7 Destination FIFO DMA Request (DDREQ[1-2]#)

Active low output. DDREQ1# is for Channel 1, and DDREQ2# is for Channel 2. These signals become active when the Dest FIFO contains enough data to deliver a DMA burst, as determined by the upper FIFO thresholds. The signals will become inactive when the Dest 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 sections 3.11.2 and 3.12 for more information on FIFO usage. All four DREQ# signals (DDREQ1#, DDREQ2,# SDREQ1#, and SDREQ2#) may be active at the same time. It is the responsibility of the host to choose which FIFO to service.

5.2.8 Destination DMA Acknowledge (DDACK[1-2]#)

Active low input. DDACK1# is for Channel 1, and DDACK2# is for Channel 2. The host asserts one of the DDACK# signals to initiate DMA reads from the Dest FIFOs. Only one of the four DACK# signals (DDACK1#, DDACK2,# SDACK1#, and SDACK2#) should be asserted at the same time. The DDACK[1-2]# signals cause the 9602 to drive the data bus with data from the selected Dest FIFO after a one-cycle delay. RDY# is used to insert wait states. See sections 3.11.2 and 3.12 for more information on FIFO usage.

5.3 Bus B Signals

5.3.1 Bus B Byte Enable (BB)

Byte-enable output for Bus B. A Bus B transfer contains valid data if this signal is asserted, and is a dummy byte if it is deasserted.

5.3.2 Bus B Clock (BCLK)

Clock input for Bus B, which operates in its own clock domain. The Bus B signals and bus interface logic use this clock.

5.3.3 Bus B EOC (BEOC#)

End-of-command output for Bus B. BEOC# is asserted on the last data output transfer of the command. This signal is equivalent to EOC#.

5.3.4 Bus B EOR (BEOR#)

End-of-record output for Bus B. BEOR# is asserted on the last data output transfer of the record. This signal is equivalent to EOR#.

5.3.5 Bus B Data (BD[7-0])

Bus B 8-bit data output bus.

5.3.6 Bus B DMA Ready (BRDY#)

Host handshake signal for Bus B. Deasserting BRDY#. creates output wait states, with the same data driven on the B[7:0] bus every cycle until BRDY# is asserted. Asserting BRDY# completes the current transfer. New data will appear on the bus for the next transfer.

5.4 Miscellaneous Signals

5.4.1 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.

5.4.2 Clock (CLK)

CLK is the system bus clock. The 9602's internal Engines are driven at 1-4 times the rate of CLK. See section 3.14.

6 Timing Description

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.

6.1 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.

6.2 DMA Interface

A DMA access is initiated when the host asserts one of the DACK# signals (DDACK1#, DDACK2#, SDACK1#, or SDACK2#), 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 Dest FIFO, as indicated by the SDREQ# and DDREQ# 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.

6.3 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

6.3.1 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 18, XDACK# is driven low by the system at the beginning of cycle 3, and it is sampled low by the 9602 at the end of clock 3.

6.3.2 Burst Transfer

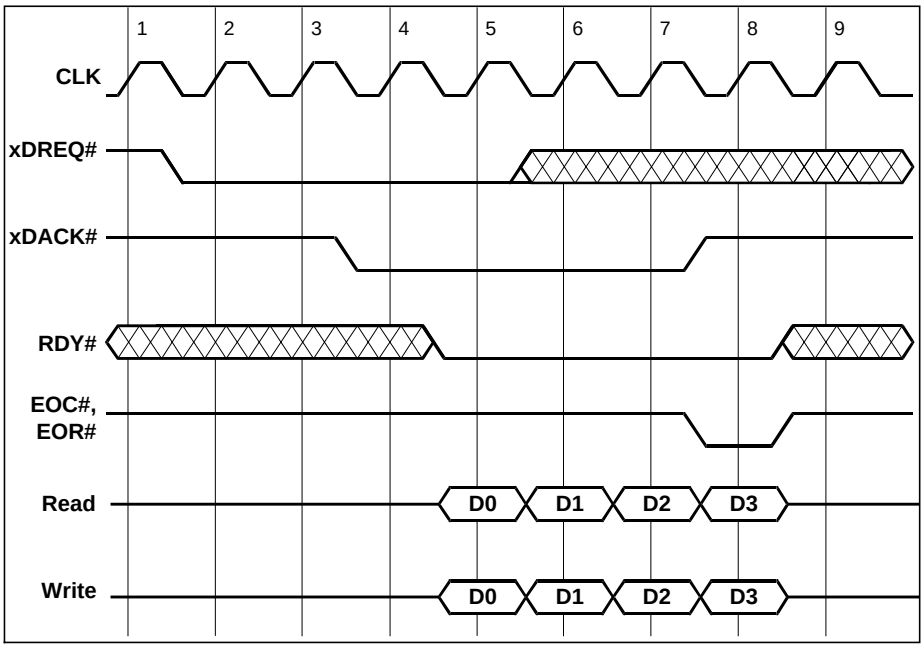


Figure 18. Four-transfer DMA burst

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 Dest FIFO (for a read transfer).

Figure 18 shows a four-transfer DMA burst. The 9602 will assert its DREQ# signals for any FIFOs that are ready to accept a data transfer. All four DREQ# signals (DDREQ1#, DDREQ2#, SDREQ1#, and SDREQ2#) may be active at once. The host DMA controller selects a target FIFO and asserts one of the DACK# signals (SDDACK1#, SDDACK2#, DDACK1#, or DDACK2#).

The 9602 samples signals on the rising edge of CLK. One cycle after the 9602 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 9602 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 3.12.

Figure 18 also shows an EOC# or EOR# 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 bye 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.

6.4 Bus Turn-Around Time

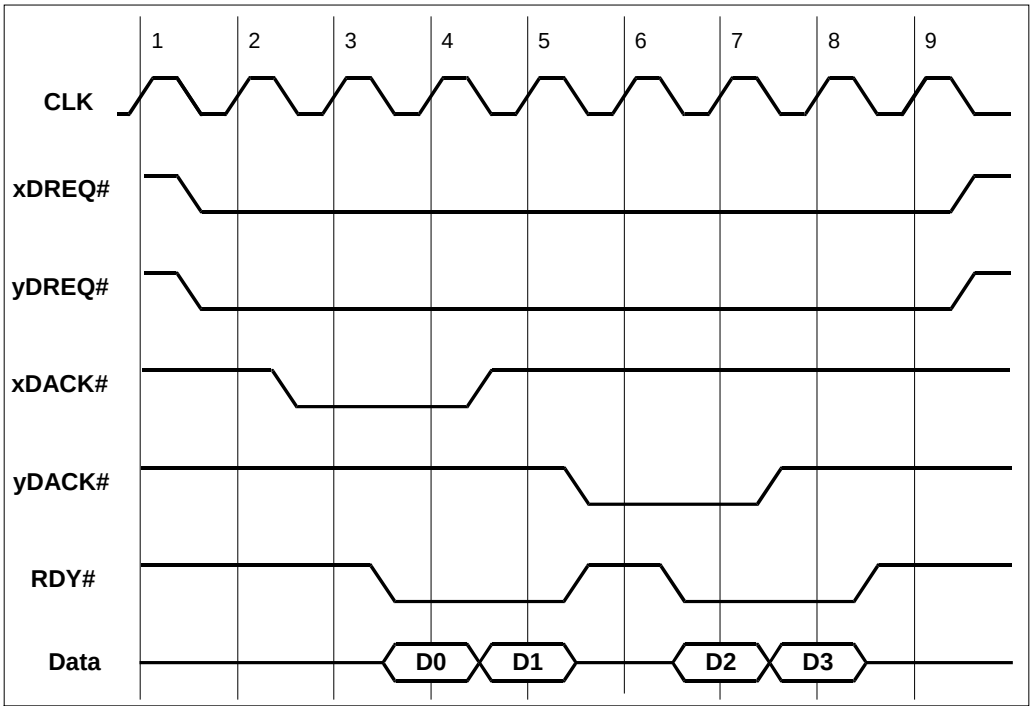


Figure 19. DMA, bus turn around time

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 Dest 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.

6.5 Fixed-Length Burst Transfers

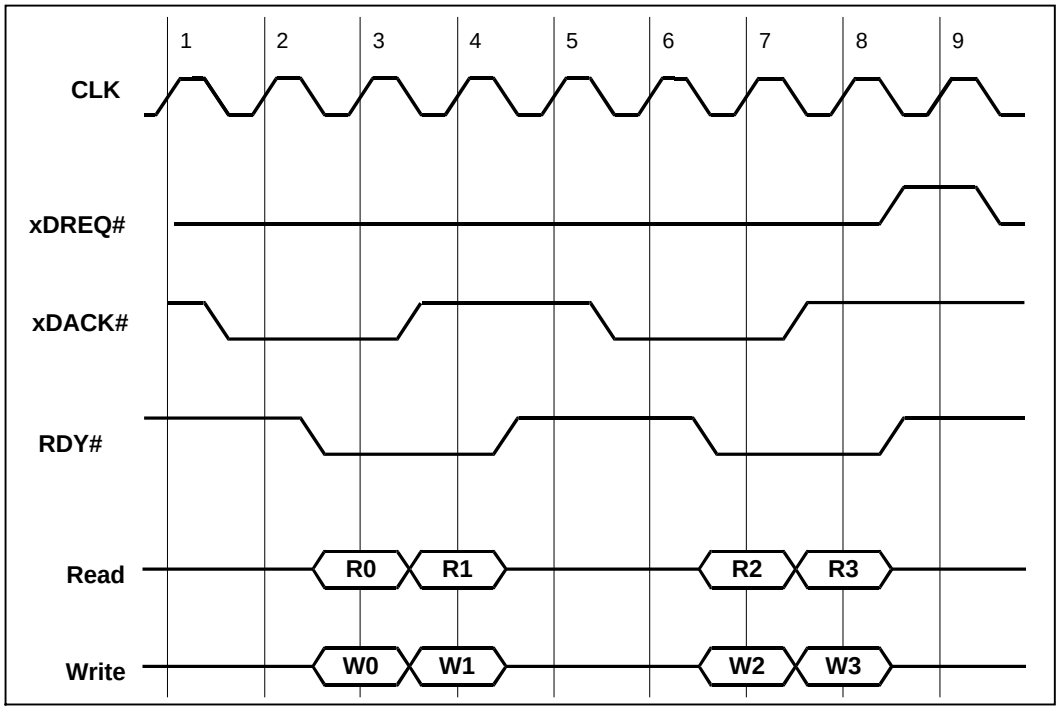


Figure 20. DMA, fixed length transfer

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 9602 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 9602 samples DACK# active and ends at the end of the clock that the 9602 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, xDREQ# 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 9602 will ignore any attempts to read and write over the DMA interface until the next command is issued.

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 sixteen-byte bursts with a 32-bit bus. This works because the 9602 will transfer data when DREQ# is deasserted.

6.6 Variable-Length Burst Transfers

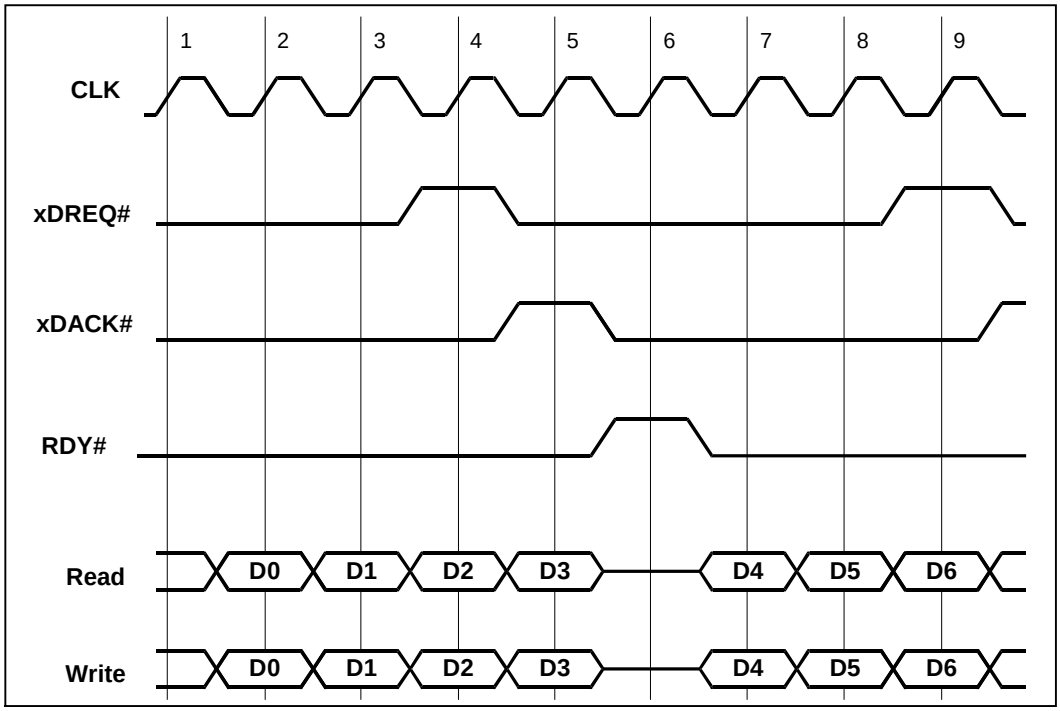


Figure 21. DMA, variable length transfer

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 9602. 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 9602 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 contoller 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 (9602 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.

6.7 Bursting with Wait States

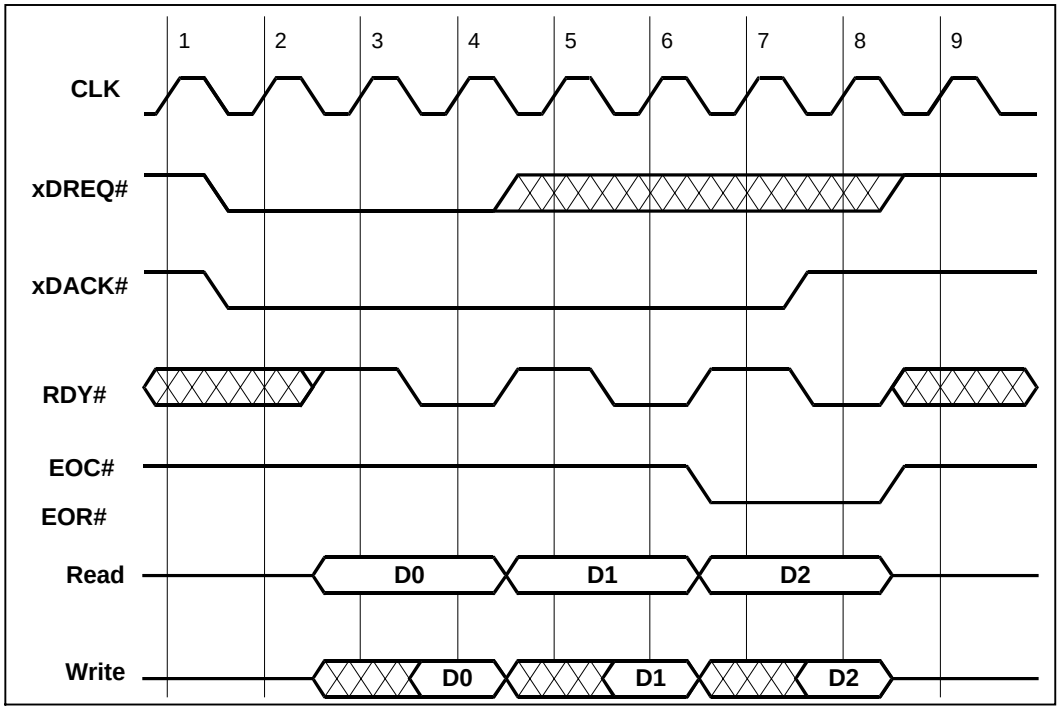


Figure 22. DMA, three-transfer burst with wait states

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 9602. They are inserted by the host to accommodate host-side delays.

When the DACK# signal is held active, the 9602 will continue to be drive the on data transfers from the Dest 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 end of command or end of record signal signifying that the D2 transfer represents the last transfer of a record (or command).

6.8 Bursting with Dummy Bytes

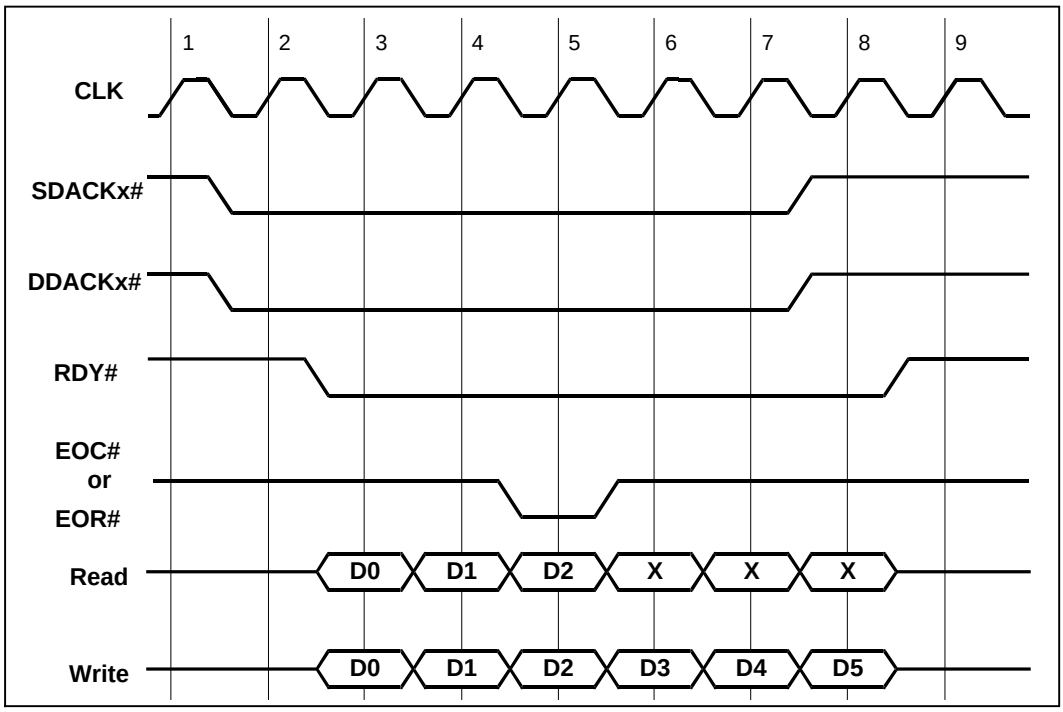


Figure 23. DMA, four-transfer burst

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 3.10.4 for the data alignment requirements of different kinds of transfers.

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. 3.10.4.

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.

Read Transfers
For read transfers from the Dest FIFO, the EOR and EOC behavior of the 9602 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 Dest 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# .

7 Specifications

7.1 DC Specifications

DC Supply Voltage (V_{DD})	-0.3V to +7.0V
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.	

Figure 24. Absolute maximum ratings

DC Supply Voltage	+3.0V to +3.6V
Operating Temperature	0°C to +70°C

Figure 25. Recommended operating conditions

Symbol	Parameter	Conditions
C_{L2}	Load on bus	20 pF
V_{DD}	Supply voltage	3.3V ± 0.3V
V_{SS}	Ground potential	0V
T_A	Ambient operating temperature	0°C to +70°C

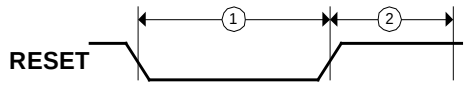
Figure 26. AC specification definition

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IL}	Low level input voltage (I, SI)				0.8	V
	Clock Input (CI)				0.8	V
V_{IH}	High level input voltage (I,SI)		2.0			V
	Clock Input (CI)		2.4			V
V_H	Schmitt hysteresis			0.8		V
I_{IL}	Low level input current (I,SI)	VIN = VSS	-10			µA
	With Pull-up (PI)	VDD = 3.6V	-40			µA
I_{IH}	High level input current	VIN = VDD VDD = 3.3V			10	µA
V_{OL}	Low level output voltage	VDD = 3.0V				
	(02)	IOL = 2mA			0.4	V
	(04)	IOL = 4mA			0.4	V
	(06)	IOL = 6mA			0.4	V
	(08)	IOL = 8mA			0.4	V
V_{OH}	High level output voltage	VDD = 3.0V				
	(02)	IOL = -2mA	2.4			V
	(04)	IOL = -4mA	2.4			V
	(06)	IOL = -6mA	2.4			V
	(08)	IOL = -8mA	2.4			V
I_{OZ}	High impedance leakage current	VO = VSS VDD = 3.6V	-10			µA
I_{DD}	Quiescent supply current				10	µA
C_{IN}	Input capacitance	VDD = 3.3V		2.4		pF
C_{OUT}	Output capacitance	VDD = 3.3V		5.6		pF
$C_{I/O}$	Input/Output capacitance	VDD = 3.3V		6.6		pF
P_A	Power dissipation	VDD = 3.3V CLK=80MHz		0.7	1.0	W
P_A	Power dissipation	VDD = 3.3V CLK =66MHz		0.5	0.7	W

Figure 27. DC electrical characteristics

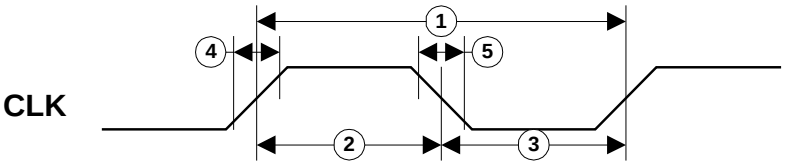
7.2 AC Specifications

7.2.1 Reset and Clock Timing



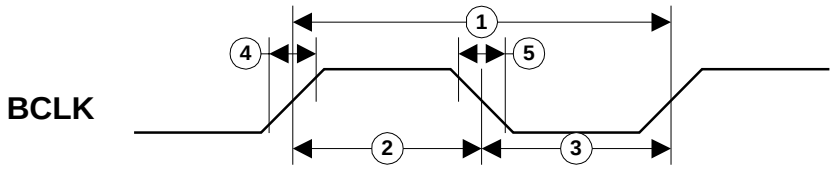
Number	Description	Min	Max	Units
1	Reset width	12*CLK		ns
2	First access after Reset (if PLL is already locked)	2*CLK		ns
-	PLL lock time after stable V_{CC}	100		µs

Figure 28. Reset and power-up timing



Number	Description	Min	Max	Units
	Oscillator frequency		66	MHz
1	Clock period	15		ns
2	Clock width high	6		ns
3	Clock width low	6		ns
4	Clock rise time from V_{IL} to V_{IH}		5	ns
5	Clock fall time from V_{IH} to V_{IL}		5	ns

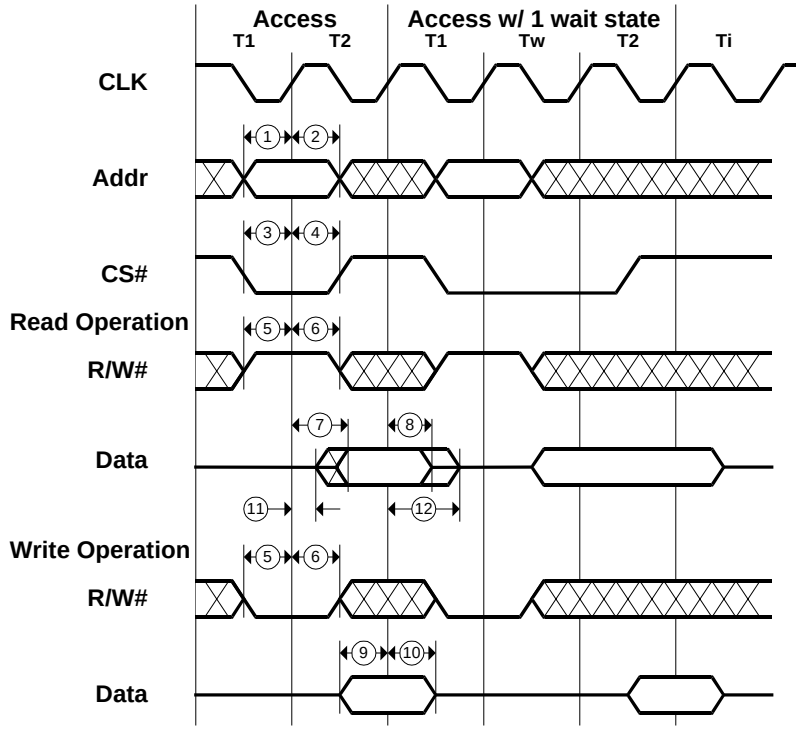
Figure 29. CLK timing



Number	Description	Min	Max	Units
	Oscillator frequency		50	MHz
1	Clock period	20		ns
2	Clock width high	7		ns
3	Clock width low	7		ns
4	Clock rise time from V_{IL} to V_{IH}		5	ns
5	Clock fall time from V_{IH} to V_{IL}		5	ns

Figure 30. BCLK timing

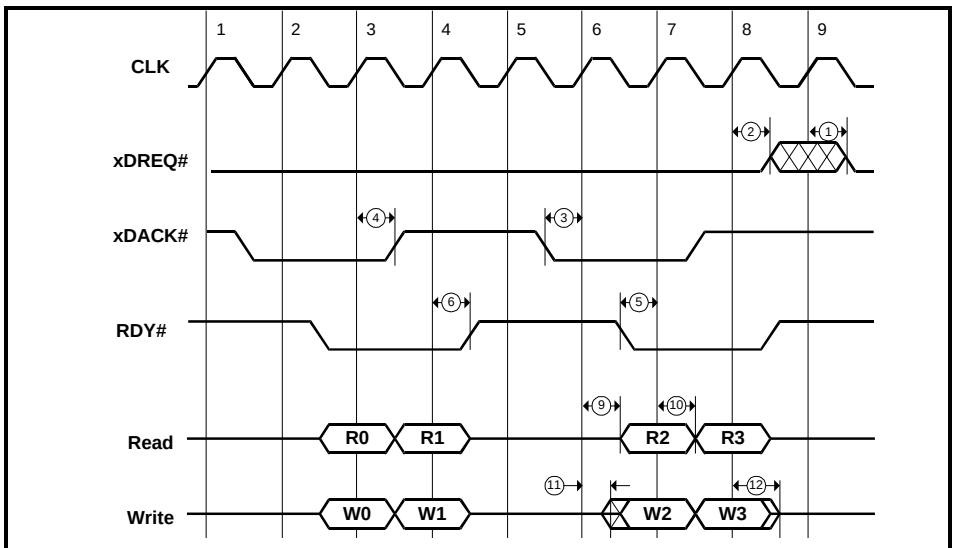
7.3 CPU Timing



Number	Description	Min	Max	Units
1	Addr setup	2.0		ns
2	Addr hold	1.5		ns
3	CS# setup	2.0		ns
4	CS# hold	1.5		ns
5	R/W# setup	2.0		ns
6	R/W# hold	1.5		ns
7	Data output valid delay (read)		11	ns
8	Data hold (read)	3.0		ns
9	Data setup (write)	2.0		ns
10	Data hold (write)	1.5		ns
11	Tri-state enable Tzo		11	ns
12	Tri-state disable Toz		9	ns

Figure 31. CPU Timing

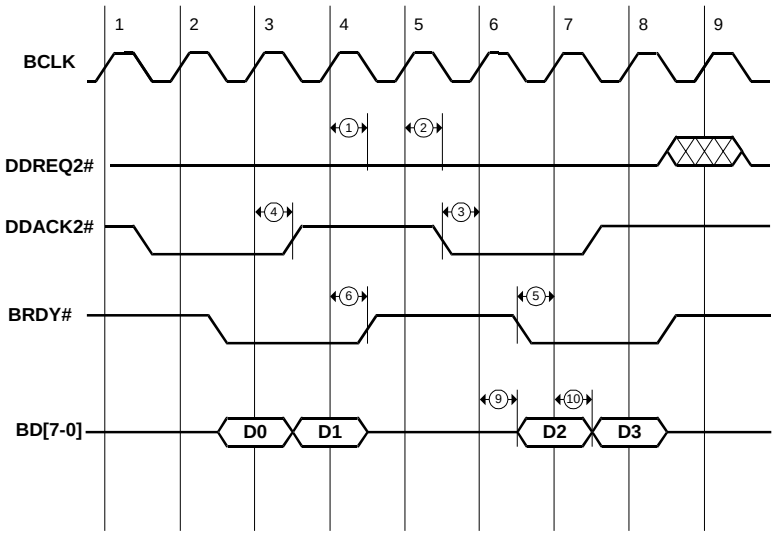
7.4 DMA Timing



Number	Description	Min	Max	Units
1	DREQ# output valid delay		11	ns
2	DREQ# hold	3		ns
3	DACK# setup	2		ns
4	DACK# hold	1.5		ns
5	RDY# setup	2		ns
6	RDY# hold	1.5		ns
7a	EOR# and EOC# output valid delay		11	ns
8	EOR# and EOC# hold	3		ns
9	Read data output valid delay		11	ns
10	Read data hold	3		ns
11	Write data setup	2		ns
12	Write data hold	1.5		ns
13	Tri-state enable Tzo		11	ns
14	Tri-state disable Toz		9	ns

Figure 32. DMA Timing

7.5 Bus B Timing



Number	Description	Min	Max	Units
1	DDREQ2# output valid delay		15	ns
2	DDREQ2# hold	3		ns
3	DDACK2# setup	3		ns
4	DDACK2# hold	1.5		ns
5	BRDY# setup	3		ns
6	BRDY# hold	1.5		ns
7a	BEOR# and BEOC# output valid delay		15	ns
8	BEOR# and BEOC# hold	3		ns
9	Read data output valid delay		15	ns
10	Read data hold	3		ns
11	BCLK frequency		50	MHz

Figure 33. Bus B Output Timing

7.6 Pin Description

Pin	Signal	Pin	Signal	Pin	Signal
1	NC	35	D15	69	IRQ#
2	VSS	36	D14	70	VSS
3	A3	37	SDREQ2#	71	CLKB
4	A2	38	SDACK2#	72	VDD
5	A1	39	D13	73	RESET#
6	A0	40	VSS	74	EOR#
7	R/W#	41	VDD	75	EOC#
8	CS#	42	D12	76	BEOR#
9	VDD	43	D11	77	BEOC#
10	D31	44	D10	78	BD7
11	D30	45	DDREQ2#	79	BD6
12	D29	46	DDACK2#	80	VSS
13	VDD	47	RDY#	81	CLK
14	D28	48	D9	82	NC
15	VSS	49	VDD	83	VDD
16	D27	50	D8	84	VDD
17	D26	51	VSS	85	VDD
18	D25	52	D7	86	VDD
19	D24	53	D6	87	NC
20	D23	54	D5	88	VSS
21	D22	55	D4	89	VSS
22	VDD	56	D3	90	VSS
23	D21	57	D2	91	VDD
24	SDREQ1#	58	VSS	92	BD5
25	D20	59	D1	93	VDD
26	VSS	60	VDD	94	BD4
27	D19	61	D0	95	BD3
28	D18	62	BRDY#	96	BD2
29	D17	63	VSS	97	BD1
30	SDACK1#	64	B3	98	BD0
31	DDREQ1#	65	B2	99	VDD
32	VDD	66	VSS	100	BB
33	DDACK1#	67	B1		
34	D16	68	B0		

Figure 34. Pin description, numerical order

Signal	Pin	Signal	Pin	Signal	Pin
A0	6	D11	43	SDACK1#	30
A1	5	D12	42	SDACK2#	38
A2	4	D13	39	SDREQ1#	24
A3	3	D14	36	SDREQ2#	37
B0	68	D15	35	VDD	9
B1	67	D16	34	VDD	13
B2	65	D17	29	VDD	22
B3	64	D18	28	VDD	32
BB	100	D19	27	VDD	41
BD0	98	D20	25	VDD	49
BD1	97	D21	23	VDD	60
BD2	96	D22	21	VDD	72
BD3	95	D23	20	VDD	83
BD4	94	D24	19	VDD	84
BD5	92	D25	18	VDD	85
BD6	79	D26	17	VDD	86
BD7	78	D27	16	VDD	91
BEOC#	77	D28	14	VDD	93
BEOR#	76	D29	12	VDD	99
BRDY#	62	D30	11	VSS	2
CLK	81	D31	10	VSS	15
CLKB	71	DDACK1#	33	VSS	26
CS#	8	DDACK2#	46	VSS	40
D0	61	DDREQ1#	31	VSS	51
D1	59	DDREQ2#	45	VSS	58
D2	57	EOC#	75	VSS	63
D3	56	EOR#	74	VSS	66
D4	55	IRQ#	69	VSS	70
D5	54	NC	1	VSS	80
D6	53	NC	82	VSS	88
D7	52	NC	87	VSS	89
D8	50	R/W#	7	VSS	90
D9	48	RDY#	47		
D10	44	RESET#	73		

Figure 35. Pin Description, Alphabetical Order

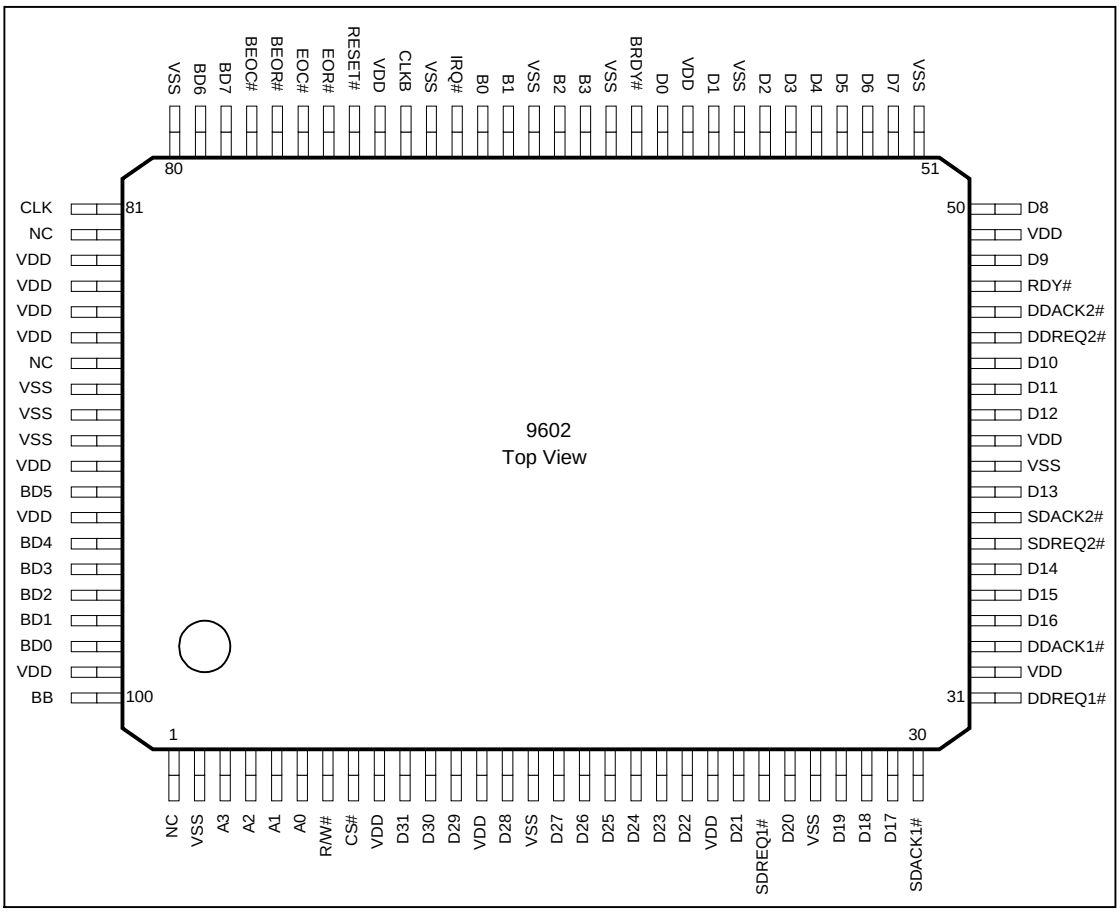


Figure 36. Pinout Diagram

7.7 Package Dimensions

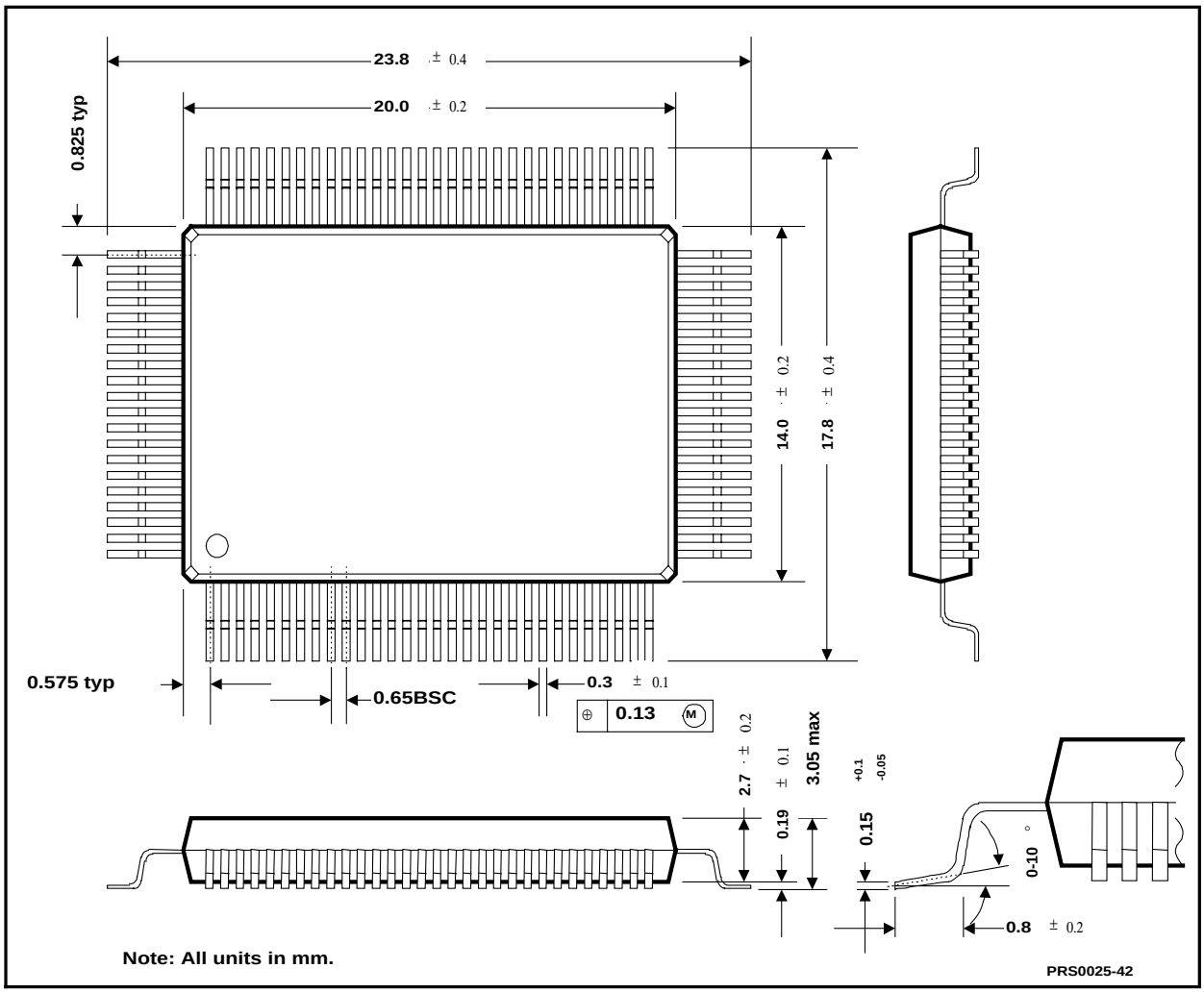


Figure 37. 100-pin PQFP package dimensions