# DATA SHEET 9610

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

1 Product Description ..... 5
2 System Concept ..... 5
3 Pin Connections ..... 6
4 Pin Descriptions ..... 7
4.1 CPU Interface ..... 7
4.2 Port A DMA Interface ..... 7
4.3 Port B DMA Interface ..... 8
4.4 Other Port Signals ..... 8
4.5 Miscellaneous Signals ..... 10
5 Product Overview ..... 10
5.1 Source FIFO Data Flow ..... 10
5.2 Dest FIFO Data Flow ..... 12
5.3 Command/Result Pipeline ..... 13
6 Register Description ..... 14
6.1 Overview ..... 14
6.2 Data (0) ..... 14
6.3 Command Stack (1) ..... 15
6.4 Command Termination Conditions ..... 15
6.5 Commands ..... 16
6.6 Command Stack Fields ..... 19
6.7 Result Stack (2) ..... 22
6.8 Configuration (3) ..... 24
6.9 Interrupt Enable (4) ..... 25
6.10 Status (5) ..... 26
6.11 FIFO Status (6) ..... 29
6.12 FIFO Configuration (7) ..... 30
7 Register Summary ..... 31
8 Timing Description ..... 32
8.1 CPU and DMA Bus Interface ..... 32
9 Electrical Specifications ..... 34
10 Timing Specifications ..... 36

## Figures

Figure 1. Tape application ..... 5
Figure 2. Printer application ..... 5
Figure 3. BYTEALIGN bit locations ..... 9
Figure 4. BYTEALIGN signals ..... 10
Figure 5. Internal Block Diagram ..... 11
Figure 6. Command/Result Pipeline ..... 13
Figure 7. Register list ..... 14
Figure 8. Commands ..... 16
Figure 9. Port and DMA Settings ..... 21
Figure 10. Bus Width ..... 24
Figure 11. Absolute Maximum Ratings ..... 34
Figure 12. Recommended Operating Conditions ..... 35
Figure 13. DC Electrical Characteristics ..... 35
Figure 14. AC Specification Definition ..... 35
Figure 15. Reset Timing ..... 36
Figure 16. External Clock ..... 36
Figure 17. Asynchronous CPU Timing ..... 36
Figure 18. Asynchronous DMA Timing ..... 37

Figure 19. Synchronous CPU Timing....38
Figure 20. Synchronous DMA Timing....39
Figure 21. 144-pin PQFP package....40
Figure 22. 144-pin PQFP pinout....41

# 1 Product Description

The Hi/fn 9610 is a very high-performance lossless data compression processor which may be used in a high speed applications, including tape drive and laser printer applications. This chip implements the industry standard Hi/fn LZS® data compression algorithm. This algorithm has been standardized by many organizations including ANSI (X3.241), QIC (122), IETF (RFC-1967, RFC-1974), TIA/EIA (655), and the Frame Relay Forum (FRF.9).

**Applications**

- Very High-speed applications
- Tape drives
- Laser printers

**Features**

- Industry standard Hi/fn LZS Compression algorithm
- Maximum 50 Mbyte/s compression and decompression speed
- In-line architecture (Two 32-bit busses)
- 16-bit CPU bus
- Internal compression RAM
- Simple operation

| Part Number | Package |
|---|---|
| 9610-20 PF7 | 20 Mbyte/s, 144-pin plastic quad flat pack |
| 9610-50 PF7 | 50 Mbyte/s, 144-pin plastic quad flat pack |

# 2 System Concept

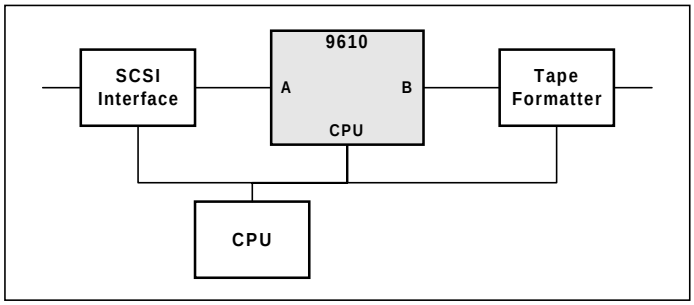



**Figure 1. Tape application**

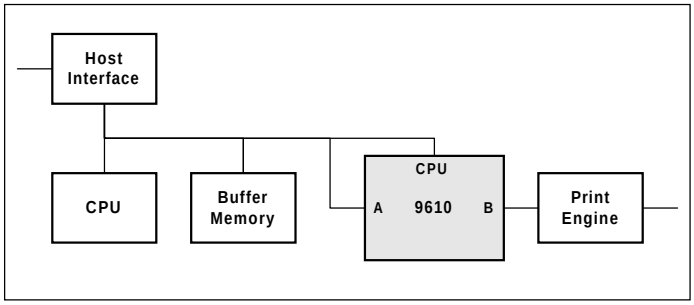



**Figure 2. Printer application**

# 3 Pin Connections

| 144-pin TQFP | Signal | Type | Description |
|---|---|---|---|
| **CPU Interface** | | | |
| 26-29,33-40, 42,44-46 | D15-0 | I/O | 16-bit data bus |
| 23-25 | A2-0 | I | Register select |
| 18 | R/W# | I | Read/Write |
| 20 | CS# | I | Command Strobe |
| 22 | IRQ# | OC | Interrupt output |
| 8 | CPUMODE | I | CPU bus mode |
| 31 | CLKC | I | CPU bus clock |
| **Port A Interface** | | | |
| 49-50, 52, 54, 58-65, 69-76, 81-86, 88, 90, 94-97 | DA31-0 | I/O | 32-bit Port A data bus |
| 56, 67, 80, 92 | PA3-PA0 | I/O | Port A Parity |
| 78 | CLKA | I | Port A bus clock |
| **Port B Interface** | | | |
| 98-101, 105-112, 116-122, 126, 130-137, 141-144 | DB31-0 | I/O | 32-bit Port B data bus |
| 103, 114, 128,139 | PB3-PB0 | I/O | Port B Parity |
| 124 | CLKB | I | Port B bus clock |
| **Other Port Signals** | | | |
| 13 | DREQ0# | O | Port A FIFO DMA request |
| 12 | DACK0# | I | Port A FIFO DMA acknowledge |
| 14 | TC0# | O | Port A FIFO terminal count |
| 10 | DREQ1# | O | Port B FIFO DMA request |
| 9 | DACK1# | I | Port B FIFO DMA acknowledge |
| 11 | TC1# | O | Port B FIFO terminal count |
| 6 | NOALIGN | O | Dest port alignment |
| **Miscellaneous Signals** | | | |
| 7 | RESET# | SI | Reset |
| 3 | CLK | CI | Main clock |
| 4 | XOUT | O | Xtal Output |
| 1, 15, 19, 30, 41, 47, 51, 55, 66, 77, 87, 91, 102, 113, 123, 127, 138 | Vcc | | +5 volts |
| 2, 16, 17, 21, 32, 43, 53, 57, 68, 79, 89, 93, 104, 115, 125, 129, 140 | Gnd | | Ground |
| 5, 48 | NC | | No connection |

I=TTL-compatible input, SI=schmitt input, CI=CMOS input, O=TTL-compatible output, OC=open collector output.

# 4 Pin Descriptions

## 4.1 CPU Interface

### 4.1.1 Address (A2-A0)

Address input signals for the CPU Interface.

### 4.1.2 Data (D15-D0)

Bi-directional data bus for the CPU Interface.

### 4.1.3 Command Strobe (CS#)

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

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

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

### 4.1.5 Interrupt (IRQ#)

Active low output. This signal will become active when any event occurs that is enabled in the Interrupt Enable register. 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 $V_{DD}$

### 4.1.6 CPU Bus Mode (CPUMODE)

This pin is used to select the bus timing mode of the CPU bus only. If CPUMODE is tied low, the CPU bus will follow the Asynchronous bus timing specifications. If CPUMODE is tied high, the CPU bus will follow the Synchronous bus timing specifications.

## 4.2 Port A DMA Interface

### 4.2.1 Port A Data (DA31-DA0)

Bi-directional data bus for the Port A DMA Interface. This bus may be configured for either 16-bit or 32-bit operation with the A BUS WIDTH bit in the Configuration Register.

While configured for 16-bit mode, the upper 16 bits (AD31-AD16) will remain tristated. The upper 16 bits must be tied low (or high) in this case to prevent these signals from floating.

### 4.2.2 Port A Parity (PA3-PA0)

Bi-directional parity bus for the Port A DMA Interface. Each bit reflects 8-bits of the data bus. The least-significant bit corresponds to the least significant byte

of the bus. If the bus is configured for 16-bit operation, only PA1-PA0 are significant. The other two parity signals will remain tri-stated, and should be tied low (or high) to prevent these signals from floating.

Odd parity is used, i.e. a byte of "00000000" would result in the corresponding parity bit of "1".

## 4.3 Port B DMA Interface

### 4.3.1 Port B Data (BD31-BD0)

Bi-directional data bus for the Port B DMA Interface. This bus may be configured for either 8-bit, 16-bit, or 32-bit operation with the B BUS WIDTH field in the Configuration Register.

While configured for 8-bit or 16-bit mode, the unused bits of the data bus will remain tri-stated. These unused bits must be tied low (or high) in this case to prevent these signals from floating.

8-bit mode is only available if the B port is a destination.

### 4.3.2 Port B Parity (PB3-PB0)

Bi-directional parity bus for the Port B DMA Interface. Each bit reflects 8-bits of the data bus. The least-significant bit corresponds to the least significant byte of the bus. If the bus is configured for 16-bit operation, only PB1-PB0 are significant. The other two parity signals will remain tri-stated. If the bus is configured for 8-bit operation, only PB0 is significant. The other signals will remain tri-stated, and should be tied low (or high) to prevent these signals from floating.

Odd parity is used, i.e. a byte of "00000000" would result in the corresponding parity bit of "1".

## 4.4 Other Port Signals

### 4.4.1 DMA Request 0 (DREQ0#)

Active low output. The Port associated with this signal is determined by the value set in the SOURCE PORT and DEST PORT fields in the Command Stack register. This signal indicates when the Port FIFO is ready to accept a DMA transfer. This signal will become inactive when the Port FIFO no longer requires a data transfer.

See the *AC Timing* section for timing details.

### 4.4.2 DMA Acknowledge 0 (DACK0#)

Active low input. The Port associated with this signal is determined by the value set in the SOURCE PORT and DEST PORT fields in the Command Stack register. This signal acknowledges a DMA transfer. See the *AC Timing* section for timing details.

### 4.4.3 Terminal Count 0 (TC0#)

Active low output. The Port associated with this signal is determined by the value set in the SOURCE PORT and DEST PORT fields in the Command Stack register. This signal indicates the last DMA transfer of the current command. Any further Port DMA activity will be for the next command in the Command Stack pipeline. See the *AC Timing* section for timing details.

### 4.4.4 DMA Request 1 (DREQ1#)

Active low output. The Port associated with this signal is determined by the value set in the SOURCE PORT and DEST PORT fields in the Command Stack register. This signal indicates when the Port FIFO is ready to accept a DMA transfer. This signal will become inactive when the Port FIFO no longer requires a data transfer.

See the *AC Timing* section for timing details.

### 4.4.5 DMA Acknowledge 1 (DACK1#)

Active low input. The Port associated with this signal is determined by the value set in the SOURCE PORT and DEST PORT fields in the Command Stack register. This signal acknowledges a DMA transfer. See the *AC Timing* section for timing details.

### 4.4.6 Terminal Count 1 (TC1#)

Active low output. The Port associated with this signal is determined by the value set in the SOURCE PORT and DEST PORT fields in the Command Stack register. This signal indicates the last DMA transfer of the current command. Any further Port DMA activity will be for the next command in the Command Stack pipeline. See the *AC Timing* section for timing details.

### 4.4.7 No Align (NOALIGN)

Active high output. This signal indicates that the last DMA transfer of a command on the destination Port contains some extra, undefined data because the number of valid bytes is not an even multiple of the bus width. This signal is valid only when the corresponding TC signal is asserted.

If the NOALIGN signal is active, the actual number of valid bytes on the last DMA transfer may be determined by external hardware by analyzing the BYTEALIGN signals. These two signals appear in the unused most significant byte lane of the destination Port data bus. See Figure 3 to determine the data bus signals that BYTEALIGN appears on. See Figure 4 for the meaning of the BYTEALIGN signals.

| Destination port bus width | Endian | BYTEALIGN1,0 signal locations |
|---|---|---|
| 32 | Big | D7, D6 |
| 32 | Little | D31, D30 |
| 16 | Big | D7, D6 |
| 16 | Little | D15, D14 |
| 8 | x | na |

**Figure 3. BYTEALIGN bit locations**

| BYTEALIGN1,0 values | valid bytes in last DMA transfer |
|---|---|
| 0, 0 | reserved |
| 0, 1 | 1 |
| 1, 0 | 2 |
| 1, 1 | 3 |

**Figure 4. BYTEALIGN signals**

## 4.5 Miscellaneous Signals

### 4.5.1 Reset (RESET#)

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

Do not access the chip until 12 clock cycles have occurred after RESET#is deasserted. Since the RESET logic propagates through the entire chip, the 12 clock cycles should be for the slowest clock input to the chip.

### 4.5.2 Clock (CLK)

This input drives the clock of all the internal logic and bus interfaces except any bus configured for synchronous operation. The clock frequency determines the speed of the Processing Unit. The speed of the Processing Unit is directly linear with the clock frequency.

### 4.5.3 CPU Bus Clock (CLKC)

This input drives the clock of the CPU bus if it is configured for synchronous operation. The clock frequency determines the speed of the bus. If the CPU bus is configured for asynchronous operation, this signal must be tied low.

### 4.5.4 Port A Bus Clock (CLKA)

This input drives the clock of the Port A bus if it is configured for synchronous operation. The clock frequency determines the speed of the bus. If the Port A bus is configured for asynchronous operation, this signal must be tied low.

### 4.5.5 Port B Bus Clock (CLKB)

This input drives the clock of the Port B bus if it is configured for synchronous operation. The clock frequency determines the speed of the bus. If the Port B bus is configured for asynchronous operation, this signal must be tied low.

# 5 Product Overview

## 5.1 Source FIFO Data Flow

The source FIFO can obtain source data from Port A, Port B, or the CPU Port depending upon the setting of the SOURCE PORT field in the Command Stack register. The following description uses the generic term, “source Port” to indicate the Port used for input data, and “dest Port” to indicate the Port used for output data.

The source Port will not request any data unless there is an active command and there is room in the source FIFO for additional data. After a command is issued, the source Port will request data. The request will remain active until the source FIFO becomes full, or until one of the command termination conditions listed in the Command Stack register description occurs. The most common command termination condition is when the Source Counter reaches zero.

The source Port requests data transfers by asserting the DREQ# signal of the source Port and by setting the SOURCE FIFO READY bit in the Status register to one. If the source Port is the CPU Port, there is no DREQ# signal. Although the source FIFO is only 64 bytes in size, more than 64 bytes could be transferred into the source FIFO in one burst because the Processing Unit is reading data from the source FIFO at the same time making room for additional data.

Once the FIFO becomes full, the source Port will stop requesting data. The source Port will request data again once the number of bytes of empty space in the source FIFO is greater than the value set in the Configure FIFO register.

Most of the command termination conditions affect the input side of the source FIFO. For example, when the number of bytes entering the source FIFO matches the value set in the SOURCE COUNT field, or if the Stop command is issued, the last byte in the source FIFO at the time the termination condition occurs is allowed to flow through to the output side of the source FIFO (and through the Processing Unit) before the command actually terminates.

The source Port will stop requesting data when the number of bytes entering the source FIFO matches the value set in the SOURCE COUNT field. The TC# signal of the source Port will be asserted to indicate the last data transfer into the source Port.

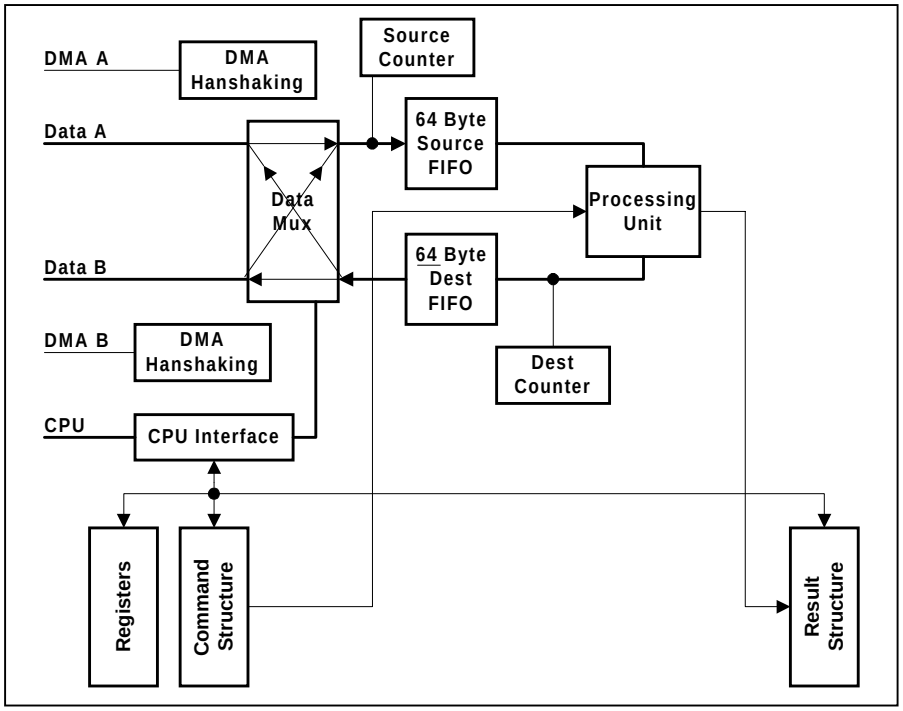



**Figure 5. Internal Block Diagram**

A command must terminate before a new command will start. Although commands may be queued, there will be a small delay from the end of one command to the start of the next command until the data within the FIFOs has been completely transferred. When the last byte of a source data for a given command exits the source FIFO and enters the Processing Unit, any trailing bytes that are not 32-bit (or 16-bit) aligned will be discarded. The first byte of source data for the next command will come from the next 32-bit (or 16-bit) aligned value.

## 5.2 Dest FIFO Data Flow

The dest FIFO can output dest data to Port A, Port B, or the CPU Port depending upon the setting of the DEST PORT field in the Command Stack register. The following description uses the generic term, “source Port” to indicate the Port used for input data, and “dest Port” to indicate the Port used for output data.

The dest Port will not request any data transfer unless there is data available in the dest FIFO. After data enters the dest FIFO, the dest Port will begin requesting data transfers once the number of bytes residing in the dest FIFO is greater than the value set in the Configure FIFO register. The request will remain active until the dest FIFO becomes empty.

The dest Port requests data transfers by asserting the DREQ# signal of the dest Port and by setting the DEST FIFO READY bit in the Status register to one. If the dest Port is the CPU Port, there is no DREQ# signal.

Although the dest FIFO is only 64 bytes in size, more than 64 bytes could be transferred out of the dest FIFO in one burst because the Processing Unit is writing data to the dest FIFO at the same time, adding additional data.

Once the FIFO becomes empty, the dest Port will stop requesting data transfers. It will request data again once the number of bytes residing in the dest FIFO is greater than the value set in the Configure FIFO register.

After the last byte of data from a command has entered the dest FIFO, the dest Port will request data transfers, independent of the FIFO threshold set by the Configure FIFO register, until the FIFO is empty. The TC# signal of the dest Port will be asserted to indicate the transfer of the last byte out of the dest FIFO.

A command must terminate before a new command will start. Although commands may be queued, there will be a small delay from the end of one command to the start of the next command until the data within the FIFOs has been completely transfered.

When the last byte of a data enters the dest FIFO from the Processing Unit, the data is padded with dummy data to fill out an entire 32-bit (or 16-bit) value. The value of any dummy byte is undefined. The first byte of destination data from the next command will be 32-bit (or 16-bit) aligned.

The NOALIGN signal indicates that the last DMA data transfer of a command on the destination Port contains some extra, undefined data because the number of valid bytes is not an even multiple of the bus width. If this signal is active, the actual number of valid bytes on the last DMA transfer may be determined by external hardware by analyzing data bus BYTEALIGN signals of the destination Port data bus.

## 5.3 Command/Result Pipeline

In order to eliminate latency while issuing commands and reading results, the command and result registers are architected into a multi-stage pipeline. Commands and results are pipelined to allow the CPU to issue commands and read results without severe timing constraints. The pipeline consists of three stages. The first stage is the Command Stack. The second stage is the Processing Unit. The third stage is the Result Stack.

When a command is written to the chip, it is written to the Command Stack in the pipeline. If the Processing Unit is available (no command is active), the command will be transferred to the Processing Unit. When a command is transferred to the Processing Unit, the Command Stack is ready to accept another command. The COMMAND READY bit in the Status register will be set to one.

Once the Processing Unit has completed the command, the result information will be transferred to the Result Stack. The RESULT READY bit in the Status register will be set to one. When a result is transferred to the Result Stack, the Processing Unit is ready to accept another command from the Command Stack.

After detecting the RESULT READY bit (via an interrupt, or polling), the CPU may read the result from the Result Stack. The three stages allow buffering of up to three operations. The pipeline allows the CPU ample time to issue commands and read results without slowing down the Processing Unit.

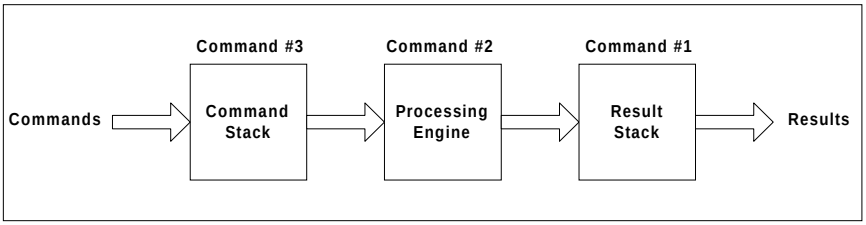



**Figure 6. Command/Result Pipeline**

# 6 Register Description

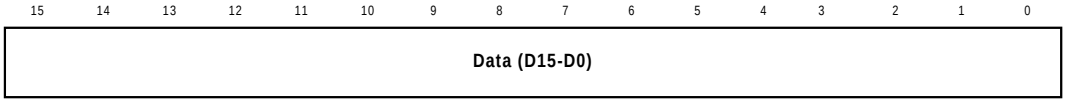
## 6.1 Overview

The 9610 uses 8 address locations for configuration and use. Reserved bits must be written as zeros and ignored when read.

| Name | Address |
|---|---|
| Data | 0 |
| Command Stack | 1 |
| Result Stack | 2 |
| Configuration | 3 |
| Interrupt Enable | 4 |
| Status | 5 |
| FIFO Status | 6 |
| FIFO Configuration | 7 |

**Figure 7. Register list**

## 6.2 Data (0)




This register can be read or written. It is used to transfer data to the Source FIFO and from Dest FIFO. Two bytes (16 bits) will be transferred at each access. See the BIG ENDIAN bit in the Configuration register for a description of the byte ordering.

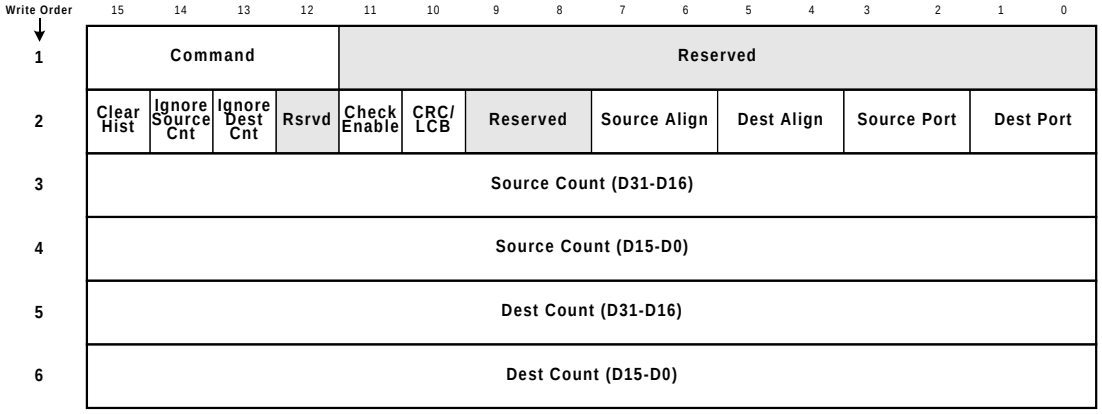
A CPU bus write operation will transfer two bytes to the Source FIFO. A read operation will transfer two bytes from the Dest FIFO.

During normal operation, the DMA interfaces should be used to transfer data to the Source FIFO or from the Dest FIFO. Therefore, this register should only be used under special conditions, or for testing.

Transfers to or from the Data register may only take place under the same conditions that a DMA transfer may take place. This is described in the *Source FIFO Data Flow* and *Dest FIFO Data Flow* sections. Use the SOURCE FIFO READY and DEST FIFO READY bits in the Status register to determine when it is valid to read or write to the Data register.

If too much data is written, or too much data is read, the command will terminate, and the DATA ERROR bit in the Result Stack and in the Status register will be set to one. This error will produce undefined results in the data stream, so the data must be ignored, and the history must be cleared.

## 6.3 Command Stack (1)




This register can be read or written. It is used to specify a command to be executed. In addition, several parameters related to the command are set here. This register operates as a stack. Six write operations are required to properly start a command. The COMMAND IN PROGRESS bit in the Status register may be used to verify the synchronization of write operations. All six writes must take place unless specified differently in the command description.

A new command can only be issued if the COMMAND READY bit in the Status register is set to one before the first write to the Command register.
See the Command/Result Pipeline section for further details of the operation of the Command/Result pipeline.

Although not normally required, the command stack may be read by the CPU. This may be done for testing purposes. Reading the Command Stack register is similar to writing to it. All six words must be read. The Command Stack should not be read until all six words have been written to the Command Stack. Only the last command written to the Command Stack is available to read. The COMMAND IN PROGRESS bit in the Status register may be used to verify the synchronization of read operations. Reading the command stack does not affect the state of the command/result overrun bit.

## 6.4 Command Termination Conditions

All the commands (except the Reset and Stop commands) will terminate when any of the following conditions occur:

1) The number of source bytes read from the Source FIFO by the Processing Unit matches the value of the Source Count field (unless the IGNORE SOURCE COUNT bit is set to one). This will produce a normal termination as defined below. In the Result Stack, the SOURCE ZERO bit will be set to one.

   The Read CRAM command is different from the other commands. It will terminate when the number of bytes written by the Processing Unit into the Dest FIFO matches the value of the Dest Count field. No other command operates this way.

2) The Stop command is issued. The command will terminate when the last byte written to the source FIFO at the time the Stop command is issued, enters the Processing Unit. This will produce a normal termination as defined below.

3) The Reset command is issued. There will be no Result Stack. Everything stops immediately, and the data in the FIFOs will be discarded.

4) An attempt is made to write data to a full source FIFO or an attempt is made to read data from an empty dest FIFO. In the Result Stack, the DATA ERROR bit will be set to one. The command will terminate when the last byte in the source FIFO at the time the data error occurs, enters the Processing Unit. This will produce a normal termination as defined below.

Note: The Dest Counter cannot terminate a command, except for the Read CRAM command. If the Dest Counter decrements below zero during the operation of any command, the command will continue to process incoming data, but no additional data will be output to the Dest FIFO. The DEST OVERRUN bit in the Result Stack will be asserted in this case.

A "normal termination" means that after the last byte of the destination data produced by a command has entered the destination FIFO, the Result Stack register for the terminated command is available, and must be read.

## 6.5 Commands

The COMMAND field specifies the command to be executed. The valid values are listed in Figure 8. All other values are invalid.

| Command | Command field |
|---|---|
| Compress | 0 |
| Decompress | 1 |
| Update History | 2 |
| Pass-Through | 3 |
| Reset | 4 |
| Stop | 5 |
| Read CRAM | 6 |
| Write CRAM | 7 |

Note: All other values for the command field are reserved.

**Figure 8. Commands**

### 6.5.1 Compress

This command will compress a block of data .

At termination, the Processing Unit will flush out its internal data, and append an End Marker to the compressed data stream. The compressed data stream after the End Marker will be padded with zeroes to round out to a whole byte.

This command will optionally append an LCB or CRC to the compressed data stream as set by the other options in this command.

### 6.5.2 Decompress

This command will decompress a block of data .

If the End Marker is found before the command terminates, all the source data between the End Marker and the last source byte will be discarded. If the End Marker is present in the last source byte, then no data will be discarded. If there is no End

Marker in the source data, the Decompression History may be invalid. This condition will be cleared automatically in the next compress or decompress operation.

This command will optionally analyze the compressed data stream by verifying LCB or CRC as set by the other options in this command.

### 6.5.3 Update History

This command will pass a block of uncompressed data from the source FIFO to the dest FIFO without modifying the data. The Decompression History will be updated with the uncompressed source data.

### 6.5.4 Pass-Through

This command will pass a block of data from the source FIFO to the destination FIFO without modifying the data. The Compression and Decompression Histories are not affected.

### 6.5.5 Reset

This command will cause the chip to immediately enter a known start-up state. This is a single word command. Any additional writes to the Command Stack register will be interpreted as the start of the next command. The affect of a Reset command is identical to asserting the RESET pin, except that the Configuration register will not be initialized.

When this command is issued, all chip operations will immediately Stop. Any queued commands will be discarded. The Source and Dest FIFOs will be cleared. Any intermediate information still residing in the chip will be lost. If a Compress operation is stopped in this manner, the Compression History will be corrupt and must be cleared next time it is used.

This command is not queued, but will operate immediately. The COMAND IN PROGRESS bit in the Status register must be zero when this command is issued. No result data is generated by this command.

Do not access the chip after a Reset command is issued until after 12 cycles of the slowest of the input clocks have occurred.

### 6.5.6 Stop

This command will cause the current Compress, Decompress, Update History, or Pass-Through command to terminate as soon as the last byte currently in the source FIFO at the time the Stop command is issued, enters the Processing Unit. All normal termination processing will take place. This is a single word com-

mand. Any additional writes to the Command Stack register will be interpreted as the start of the next command.

Since there is no source count associated with the stop command, all four bytes from the last source FIFO transfer are processed. That is, there is limited control over exactly how many bytes are processed when utilizing the Stop command. The number of bytes can be controlled by offsetting the source data and using the dest aligned bits in the Command Stack.

This command will not affect any command residing in the Command Stack pipeline. If there is no command currently operating, then the Stop command will be ignored.

This command is useful to terminate a command if the IGNORE SOURCE COUNT bit is set.

This command is not queued, but will operate immediately. No result data is generated by this command (except for the result data of the command that is being stopped).

The Stop command does not affect an active Read CRAM command. The Read CRAM is terminated only by the dest counter reaching zero.

### 6.5.7 Read CRAM

This command is used to transfer data directly from the CRAM (Compression RAM) to the destination FIFO. This is normally used for debugging purposes or for CRAM memory testing.
The starting byte address of the CRAM must be set in the SOURCE COUNT field.

The DEST COUNT field must be set to the number of bytes to read from the CRAM. DESTINATION COUNT field should not be set to zero.The total size of the CRAM is 2Kbytes. Setting DEST COUNT to a larger value will result in address wrapping and reading the same contents multiple times. Only bits 10:0 of SOURCE COUNT are used (upper bits are ignored).

To properly terminate the ReadCRAM command, the IGNORE DEST COUNT bit must be set to zero. Since the source FIFO is not involved with a Read CRAM command, issuing a stop command has no affect on Read CRAM command.

### 6.5.8 Write CRAM

This command is used to transfer data directly to the CRAM (Compression RAM) from the source FIFO. This is normally used for debugging purposes or for CRAM memory testing.

The starting byte address of the CRAM must be set in the DEST COUNT field.

The SOURCE COUNT field must be set to the number of bytes to write to the CRAM.

The total size of the CRAM is 2Kbytes. Setting SOURCE COUNT to a larger value will result in address wrapping and writing the same CRAM addresses multiple times. Only bits 10:0 of DEST COUNT are used (upper bits are ignored).

## 6.6 Command Stack Fields

### 6.6.1 Clear History

This bit is significant for the Compress command only. If this bit is set to one, the Compression History will be cleared before the compress operation begins. This bit must be set to zero for all other commands.

If this bit is set to zero, the Compression History will not be cleared. The compress operation will use history information from previous compress operations.

Note: This bit must be set to one for the first compress operation after a hardware reset.

Due to the nature of the Hi/fn LZS compression format, it is never necessary to clear a Decompression History.

### 6.6.2 Ignore Source Count

This bit is significant for the Compress, Decompress, Update History, Pass-Through, and Write CRAM commands. If this bit is set to one, the source counter will not terminate the command when the counter reaches zero. The source counter will wrap from zero to $FFFFFFFF_{16}$.

If this bit is set to zero, the specified command will terminate when the source counter reaches zero.
In either case, the source counter will be initialized to the value set in the SOURCE COUNT field. Also, the command may terminate from other conditions listed previously.

During the Read CRAM command, since the source FIFO and source counter logic are not involved with the operation, the source counter value is not updated from the command stack and therefore an invalid value of the source counter is returned in the result stack.

### 6.6.3 Ignore Dest Count

This bit is significant for the Compress, Decompress, Update History, Pass-Through, and Read CRAM commands. If this bit is set to one, the dest counter will not cause the Processing Unit to stop producing data when the counter reaches zero. The dest counter will wrap from zero to $FFFFFFFF_{16}$.

If this bit is set to zero, the Processing Unit will stop producing data when the dest counter reaches zero. Also, the dest counter will not decrement further.

In either case, the dest counter will be initialized to the value set in the DEST COUNT field.

During the Read CRAM command, the IGNORE DEST COUNT bit must be set to zero, since the Read CRAM is terminated only by the dest counter reaching zero. The Stop command does not affect an active Read CRAM command.

### 6.6.4 Check Enable

This bit is significant for the Compressand Decompress commands. It enables the in-line generation and verification of a check algorithm. If this bit is set to zero, there will be no in-line check fields generated or verified. The CHECK ENABLE bit does not affect the calculation of the CHECK FIELD in the Result Stack register.

For a Compress command, if this bit is set to one, a check field will be appended to the compressed destination data stream.

For a Decompress command, if this bit is set to one, the check field will be verified. If the check value calculated for the current block of data does not match the check field embedded in the data stream, the CHECK ERROR bit in the Result Stack will be set to one. Also, the CHECK ERROR in the Status register will be set to one.

Bits 7:0 of the 16-bit CRC are appended first to the datastream, followed by bits 15:8. In case of LCB, only bits 7:0 are appended to the data stream.

### 6.6.5 CRC/LCB

This bit is significant for the Compress, Decompress, Update History, Pass-Through, Read CRAM, and Write CRAM commands. It selects between the use of an LCB or a CRC check algorithm. This affects the value reported in the Result Stack as well as the check field optionally inserted or checked in the data stream (check field is for compress and decompress commands only).
If this bit is set to zero, an 8-bit LCB (Longitudinal Check Byte) will be used. The LCB is an exclusive-OR sum of each uncompressed byte. The LCB is initialized to $FF_{16}$ at the start of each new command.

If this bit is set to one, a 16-bit CRC will be used. The CRC is calculated on the uncompressed data. The CRC is initialized to $FFFF_{16}$ at the start of each new command. The CRC polynomial is:

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

### 6.6.6 Source Align

This field is significant for the Compress, Decompress, Update History, Pass-Through, and Write CRAM commands. If the first byte of source data is not 32-bit (or 16-bit) aligned, the SOURCE ALIGN field may be used to ignore the first few bytes of source data. The number of bytes ignored is set in this field.

The ignored bytes are counted by the source counter.

If the source port bus width is configured for 16 bits, then the SOURCE ALIGN field must be one or zero. Otherwise the values three to zero are valid.

### 6.6.7 Dest Align

This field is significant for the Compress, Decompress, Update History, Pass-Through, and Read CRAM commands. If it is desired to force the first byte of destination data to a non-32-bit (or non-16-bit) alignment, the DEST ALIGN field may be used to produce a few bytes of data that will be inserted in front of the destination data. The value of the inserted bytes are undefined. The number of bytes inserted is set in this field.

The inserted bytes are counted by the dest counter.

If the dest port bus width is configured for 16 bits, then the DEST ALIGN field must be one or zero. Otherwise the values three to zero are valid.

### 6.6.8 Source Port

This field is significant for all commands except for Reset, Stop, and Read CRAM. This field determines which Port and DMA channel will be used as an input for source data. See Figure 9 for details.

| Source Port field | Dest Port field | Source Port | Dest Port | Source DMA | Dest DMA |
|---|---|---|---|---|---|
| 00 | 00 | Port A | Port A | DMA0 | DMA1 |
| 00 | 01 | Port A | Port B | DMA0 | DMA1 |
| 00 | 10 | Port A | CPU Port | DMA0 | none |
| 01 | 00 | Port B | Port A | DMA1 | DMA0 |
| 01 | 01 | Port B | Port B | DMA1 | DMA0 |
| 01 | 10 | Port B | CPU Port | DMA1 | none |
| 10 | 00 | CPU Port | Port A | none | DMA0 |
| 10 | 01 | CPU Port | Port B | none | DMA1 |
| 10 | 10 | CPU Port | CPU Port | none | none |

**Figure 9. Port and DMA Settings**

### 6.6.9 Dest Port

This field is significant for all commands except for Reset, Stop, and Write CRAM. This field determines which Port and DMA channel will be used as an output for destination data. See Figure 9 for details.

### 6.6.10 Source Count

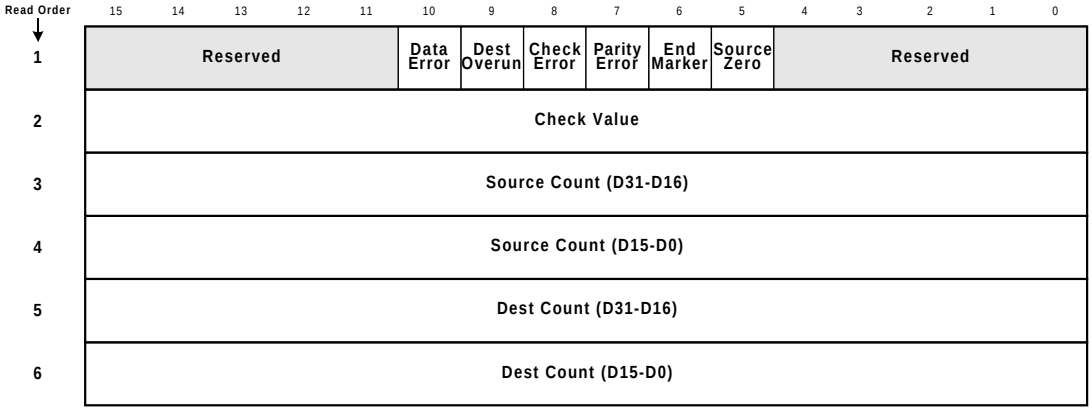
This field is significant for all commands except the Reset and Stop commands. This is the initial value used for the Source Counter. The Source Counter is decremented for each source byte processed by the Processing Unit. For the Read CRAM command this is the first CRAM address to be read.
The Source Counter will decrement for every byte processed, including bytes specified by the SOURCE ALIGN field, data, and an input check value (if enabled). It will not decrement for any extra bytes inserted due to the final transfer not filling up the whole word.

### 6.6.11 Dest Count

This field is significant for all commands except the Reset and Stop commands. This is the initial value used for the Dest Counter. The Dest Counter is decremented for each destination byte produced by the Processing Unit. For the Write CRAM command this is the first CRAM address to be written.

The Dest Counter will decrement for every byte processed, including bytes specified by the DEST ALIGN field, data, and an output check value (if enabled). It will not decrement for any extra bytes inserted due to the final transfer not filling up the whole word.

## 6.7 Result Stack (2)




This register can be read only. It is used to read the result of a completed command.

This register operates as a stack. Six read operations 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 six reads must take place even if some of the information will not be used.

See the Command/Result Pipeline section for further details of the operation of the Command/Result pipeline.

### 6.7.1 Data Error

This bit is significant for all commands. This bit will be set to one if an attempt is made to write to a full source FIFO or an attempt is made to read data from an empty dest FIFO.

This error condition will cause the automatic execution of a Stop command. The current command will terminate as soon as the source FIFO becomes empty. See the Stop command description in the Command Stack for additional information. Also, any pipelined commands will be cleared. No new commands may be written until the DATA ERROR bit in the Status register is cleared.

This condition may be one reason why the operation terminated.

### 6.7.2 Dest Overrun

If this bit is set to one, the command produced more data than expected. The dest counter attempted to decrement below zero, and the IGNORE DEST COUNT bit was set to zero. This condition does not terminate a command, however, the command continued to process incoming data, but no additional data was output to the dest FIFO. Any further source data is discarded until the command terminates. The dest counter will remain at zero.If this bit is set to zero, then the dest

counter did not attempt to decrement below zero, or the IGNORE DEST COUNT bit was set to one.

### 6.7.3 Check Error

This bit is significant for the Decompress command only, and only if the CHECK ENABLE bit was set in the Decompress command. If the CHECK ERROR bit is set to one, the check value calculated for the current block of data did not match the check field embedded in the data stream, or the check value is missing from the data stream (due to the Source Count reaching zero prematurely, for example).

If this bit is zero, the check value matched correctly.

### 6.7.4 Parity Error

This bit is significant for all commands. This bit will be set to one if a parity error is detected on a DMA bus (indicating that the source data was not valid). The value of this bit will not affect the operation of the current command.

### 6.7.5 End Marker

This bit is significant for the Decompress command only. If this bit is set to one, the Processing Unit detected the End Marker in the source data stream. This condition may be one reason why the decompress operation terminated.

If this bit is set to zero, the End Marker was not detected before the operation terminated.

### 6.7.6 Source Zero

This bit is significant for the Compress, Decompress, Update History, Write CRAM, and Pass-Through commands. If this bit is set to one, the source counter reached zero. This condition may be one reason why the operation terminated.

If this bit is set to zero, the source counter did not reach zero before the operation terminated, or the IGNORE SOURCE COUNT bit was set to one.

### 6.7.7 Check Value

This field is significant for the Compress, Decompress, Update History, Pass-Through, Read CRAM, and Write CRAM commands. This is the calculated LCB or CRC (as set by the CRC/LCB bit in the Command Stack) check value. If configured for LCB, then only bits 7 to 0 are significant. The others are unused. If configured for CRC, then bits 15 to 0 are significant.

This field is significant even if the CHECK ENABLE bit in the Command Stack register is set to zero.

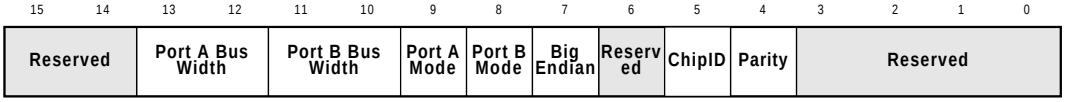
### 6.7.8 Source Count

This field is significant for all commands except the Reset, Stop, and Read CRAM commands. This is the final value of the source counter at command termination. The source counter is decremented for each source byte processed by the Processing Unit.

### 6.7.9 Dest Count

This field is significant for all commands except the Reset, Stop, and Write CRAM commands. This is the final value of the dest counter at command termination. The dest counter is decremented for each destination byte produced by the Processing Unit.

## 6.8 Configuration (3)




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.

### 6.8.1 Port A Bus Width

This field selects the Port A bus width. See Figure 10 for the relationship between the value in this field and the actual Port A bus width.

| Bus width field | Port A bus width | Port B bus width |
|---|---|---|
| 0 | reserved | 8 |
| 1 | 16 | 16 |
| 2 | 32 | 32 |
| 3 | reserved | reserved |

**Figure 10. Bus Width**

### 6.8.2 Port B Bus Width

This bit selects the Port B bus width. See Figure 10 for the relationship between the value in this field and the actual Port A bus width.

### 6.8.3 Port A Mode

This bit selects the bus timing mode of the Port A bus. If this bit is set to zero, the bus will follow the Asynchronous bus timing specifications. If this bit is set to one, the bus will follow the Synchronous bus timing specifications.

### 6.8.4 Port B Mode

This bit selects the bus timing mode of the Port B bus. If this bit is set to zero, the bus will follow the Asynchronous bus timing specifications. If this bit is set to one, the bus will follow the Synchronous bus timing specifications.

### 6.8.5 Big Endian

This bit selects the byte order of data transferred via the DMA interfaces, and via the Data register.

If this bit is set to zero, this chip operates as if it were attached to a Little Endian processor. The least significant byte of transferred data will enter or leave the Processing Unit first.

If this bit is set to one, this chip operates as if it were attached to a Big Endian processor. The most significant byte of transferred data will enter or leave the Processing Unit first.

### 6.8.6 ChipID

This bit allows the ChipID value to be read from the Status register. When this bit is set to one, the ChipID value can be read once from the Status register. After the ChipID value is read from the Status register, this bit returns to zero.

This bit is readable. When this bit is set to one, it returns the value one until after the ChipID value is read from the Status register. After the ChipID value is read from the Status register this bit is read as the value zero.

### 6.8.7 Parity

This bit is used to enable the DMA bus parity. If this bit is set to one, DMA bus parity will be generated and checked. If this bit is set to zero, DMA parity will not be generated or checked.

## 6.9 Interrupt Enable (4)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 5 4 3 2 1 0 |
|---|---|---|---|---|---|---|---|---|---|
| Result Ready Int En. | Cmnd Ready Int En. | Source FIFO Ready Int En. | Dest FIFO Ready Int En. | Cmnd/ Result Overrn Int En. | Data Error Int En. | Dest Overun Int En. | Check Error Int En. | Parity Error Int En. | Reserved |

This register may be read or written. It is used to configure the chip. The default value of all the fields in this register after a hardware reset is zero.

### 6.9.1 Result Ready Interrupt Enable

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

### 6.9.2 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.

### 6.9.3 Source FIFO Ready Interrupt Enable

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

### 6.9.4 Dest FIFO Ready Interrupt Enable

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

### 6.9.5 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.

### 6.9.6 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.

### 6.9.7 Dest Overrun Interrupt Enable

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

### 6.9.8 Check Error Interrupt Enable

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

### 6.9.9 Parity Error Interrupt Enable

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

## 6.10 Status (5)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 5 4 | 3 | 2 | 1 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Result Ready | Cmnd Ready | Source FIFO Ready | Dest FIFO Ready | Cmnd/ Result Overun | Data Error | Dest Overun | Check Error | Parity Error | Reserved | Cmnd In Prog. | Result In Prog. | Reserved |

This register may be read or written. It is used to monitor the status of the chip. The default value of the bits in this register are the valid values for a quiescent chip, and are listed under each bit description.

### 6.10.1 Result Ready

This bit is set to one when a result is ready to be read from the Result Stack.

This bit may be cleared by writing a one to this bit. This bit may also be cleared by reading the first word of the Result Stack. When cleared, this bit will not be set to one again until the result of the next command is ready to be read (after reading the six words of the current result).

The default value of this bit is zero.

### 6.10.2 Command Ready

This bit is set to one when the Command Stack is ready to accept a new command.

This bit may be cleared by writing a one to this bit. This bit may also be cleared by writing the first word to the Command Stack. When cleared, this bit will not be set to one again until the chip is ready to accept another new command (after all six writes of the current command have been written).

The default value of this bit is one.

### 6.10.3 Source FIFO Ready

This bit is set to one when the available space in the Source FIFO exceeds the threshold programmed in the FIFO Configuration register (when the DREQ# signal of the source FIFO port is active).

This bit is set to zero when the source FIFO is full (when the DREQ#signal is inactive). See the *Source FIFO Data Flow* section for more information regarding this bit. This bit must be set before the FIFO Status Register will contain a valid value in the SOURCE FIFO field.

If the CPU is the source port, then the DREQ# signal is not used. But the functional timing of the SOURCE FIFO READY bit remains the same (as if there were a DREQ# signal).

This bit may not be cleared by writing to this bit. The default value of this bit is zero.

### 6.10.4 Dest FIFO Ready

This bit is set to one when the number of bytes in the Dest FIFO exceeds the threshold programmed in the FIFO Configuration register (when the DREQ# signal of the dest FIFO port is active).

This bit is set to zero when the dest FIFO is empty (when the DREQ#signal is inactive). See the *Dest FIFO Data Flow* section for more information regarding this bit. This bit must be set before the FIFO Status Register will contain a valid value in the DEST FIFO field.

If the CPU is the dest port, then the DREQ# signal is not used. But the functional timing of the DEST FIFO READY bit remains the same (as if there were a DREQ# signal).

This bit may not be cleared by writing to this bit. The default value of this bit is zero.

### 6.10.5 Command/Result Overrun

This bit is set to one if the Command Stack is written when it is not ready, or the Result Stack is read when it is not ready.
This error condition will not affect any command in operation, or any command successfully queued. However, no new commands may be written until this status bit is cleared.

This bit may be cleared by writing a one to this bit. The default value of this bit is zero.

### 6.10.6 Data Error

This bit is set to one if the Source FIFO is written when it is not ready, or the Dest FIFO is read when it is not ready. This applies to any Port (Port A, Port B, or the CPU Port).

This error condition will cause the automatic execution of a Stop command. The current command will terminate as soon as the source FIFO becomes empty. See the Stop command description in the Command Stack for additional information. Also, any queued commands will be cleared. No new commands may be written until this status bit is cleared.

When this bit is set, the DATA ERROR bit in the Result Stack will also be set to one.

This bit may be cleared by writing a one to this bit. The default value of this bit is zero.

### 6.10.7 Dest Overrun

This bit is set to one if the command produced more data than expected. The dest counter attempted to decrement below zero, and the IGNORE DEST COUNT bit was set to zero. When the dest counter attempted to decrement below zero, the command continue to process incoming data, but no additional data was output to the dest FIFO. This did not cause the command to terminate. The dest counter will remain at zero.

This bit may be cleared by writing a one to this bit. The default value of this bit is zero.

### 6.10.8 Check Error

This bit is set to one If the check value calculated for the current block of data does not match the check field embedded in the data stream. This error is also reported in the CHECK ERROR bit in the Result Stack register

This bit may be cleared by writing a one to this bit. The default value of this bit is zero.

### 6.10.9 Parity Error

This bit is set to one If a parity error on a DMA bus is detected. This error is also reported in the PARITY ERROR bit in the Result Stack register.

This bit may be cleared by writing a one to this bit. The default value of this bit is zero.

### 6.10.10 Command In Progress

This 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 Stack register. This bit returns to zero after the last (sixth) access to the Command Stack register.

The default value of this bit is zero.

### 6.10.11 Result In Progress

This 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 Stack register. This bit returns to zero after the last (sixth) read from the Result Stack register.

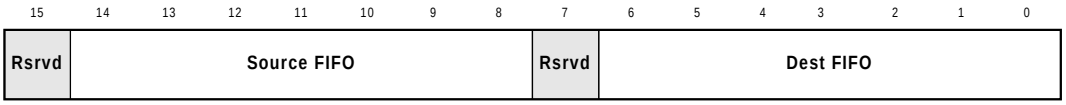
The default value of this bit is zero.

### 6.10.12 Chip ID

When the ChipID bit (bit 5) in the Configuration register is set, this register returns the Chip ID value 0x02XX. The upper 8 bits are defined as the product ID code and the lower 8 bits are reserved.

Setting the ChipID bit allows only one read of this register. If the ChipID value is to be read again the ChipID bit in the Configuration register must be set to one again.

## 6.11 FIFO Status (6)




This is a read-only register. It is used to determine the number of bytes residing in the source FIFO and dest FIFO.

During normal operation, the DMA interface should be used to transfer data to the source FIFO and from the dest FIFO. Therefore, this register should only be used under special conditions, or for testing.

Settling time is required between source or dest FIFO transfers and reading this register. There must be 12 clocks between the last write to the source FIFO and the read of the SOURCE FIFO field of this register, and there must be 12 clocks between the last write to the dest FIFO and the read of the DEST FIFO field of this register. These clocks must be the slowest of CLK, CLKC, and CLKA or CLKB.

The value of this register is latched on each access to the 9610. The value read is always the previously latched value. Therefore, if it is unknown when the previous access to the 9610 took place, the user should read the value of this register twice, and use only the second value.

Except for the last transfer, the actual number of bytes transferred must be 4 bytes less than the value read in the register. This will avoid completely filling or emptying the FIFOs, which may negate the SOURCE FIFO READY or DEST FIFO READY bits and generate a data error condition.

The Source FIFO and Dest FIFO are each 64 bytes in size.

### 6.11.1 Source FIFO

This field indicates the number of bytes available to be written to the source FIFO. For example, if the source FIFO is empty, this field will indicate 64. The default value for this field is undefined.

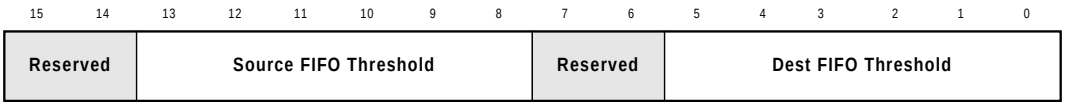
The SOURCE FIFO READY bit in the Status register must be set for this register to be valid.

### 6.11.2 Dest FIFO

This field indicates the number of bytes residing in the dest FIFO. For example, if the dest FIFO is empty, this field will indicate 0. The default value for this field is undefined.

The DEST FIFO READY bit in the Status register must be set for this register to be valid.

## 6.12 FIFO Configuration (7)




This register may be read or written. It is used to configure the FIFO thresholds at which a FIFO will begin to request a data transfer. This threshold is used to determine when the FIFO ready status bits in the Status register will be set to one. This may optionally assert an interrupt if configured in the Configuration register.

The threshold is also used to determine when the DMA Interface logic will begin to request for a block of DMA transfers. See the *Product Overview* section for more information.

### 6.12.1 Source FIFO Threshold

This field sets the source FIFO Threshold. When the number of bytes available to be written to the source FIFO is greater than the value set in this field, the SOURCE FIFO READY bit in the Status register will be set to one and the DREQ# signal of the source DMA interface will become active. See the *Source FIFO Data Flow* section for more information regarding this field.

The 9610 assumes a minimum value for the source threshold field of 3 bytes for a 16-bit bus, and 7 bytes for a 32-bit bus. This assures that there are at least 2 bus transfers that can take place before source FIFO becomes full.

### 6.12.2 Dest FIFO Threshold

This field sets the dest FIFO Threshold. When the number of bytes residing in the dest FIFO is greater than the value set in this field, the DEST FIFO READY bit in the Status register will be set to one and the DREQ# signal of the dest DMA interface will become active. See the *Dest FIFO Data Flow* section for more information regarding this field.
The 9610 will never assert DEST FIFO READY bit 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 a full bus width). In addition, the dest FIFO always keeps at least one bus transaction in the chip until the very end of the command. This may give the appearance that the DEST FIFO READY bit is asserted later than expected.

# 7 Register Summary

### 0 - Data

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data (D15-D0) | | | | | | | | | | | | | | | |

### 1 - Command

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Command | | | | Reserved | | | | | | | | | | | |
| Clear Hist | Ignore Source Cnt | Ignore Dest Cnt | Rsrvd | Check Enable | CRC/ LCB | Reserved | | Source Align | | Dest Align | | Source Port | | Dest Port | |
| Source Count (D31-D16) | | | | | | | | | | | | | | | |
| Source Count (D15-D0) | | | | | | | | | | | | | | | |
| Dest Count (D31-D16) | | | | | | | | | | | | | | | |
| Dest Count (D15-D0) | | | | | | | | | | | | | | | |

### 2 - Result

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | Data Error | Dest Overun | Check Error | Parity Error | End Marker | Source Zero | Reserved | | | | |
| Check Value | | | | | | | | | | | | | | | |
| Source Count (D31-D16) | | | | | | | | | | | | | | | |
| Source Count (D15-D0) | | | | | | | | | | | | | | | |
| Dest Count (D31-D16) | | | | | | | | | | | | | | | |
| Dest Count (D15-D0) | | | | | | | | | | | | | | | |

### 3 - Configuration

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | Port A Bus Width | | Port B Bus Width | | Port A Mode | Port B Mode | Big Endian | Reserved | ChipID | Parity | Reserved | | | |

### 4 - Interrupt Enable

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Result Ready Int En. | Cmnd Ready Int En. | Source FIFO Ready Int En. | Dest FIFO Ready Int En. | Cmnd/ Result Overrn Int En. | Data Error Int En. | Dest Overun Int En. | Check Error Int En. | Parity Error | Reserved | | | | | | |

### 5 - Status

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Result Ready | Cmnd Ready | Source FIFO Ready | Dest FIFO Ready | Cmnd/ Result Overun | Data Error | Dest Overun | Check Error | Parity Error | Reserved | | | Cmnd In Prog. | Result In Prog. | Reserved | |

### 6 - FIFO Status

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rsrvd | Source FIFO | | | | | | | Rsrvd | Dest FIFO | | | | | | |

### 7 - FIFO Configuration

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | Source FIFO Threshold | | | | | | Reserved | | Dest FIFO Threshold | | | | | |

# 8 Timing Description

## 8.1 CPU and DMA Bus Interface

The CPU and DMA busses may be configured for Asynchronous or Synchronous timing modes. This is configured by the use of the CPUMODE signal for the CPU bus, and the PORT A MODE and PORT B MODE bits in the Configuration register.

In the following discussions (including the timing section), the signals DREQ# and DACK# are used in place of the signals DREQ0, DREQ1, DACK0, and DACK1. Furthermore the signal TC# is used in place of the signals TC0# and TC1#. Generic terms are used, because the actual signals are configured by the use of the SOURCE PORT and DEST PORT bits in the Command Stack.

### 8.1.1 Asynchronous Mode on CPU Interface

If configured for Asynchronous operation, the bus interface is driven internally by the frequency of the CLK input pin.

A typical CPU access consists of setting up the R/W and ADDR signals prior to the assertion of the CS# signal. The R/W and ADDR signals must remain stable for a short time after the trailing edge of CS#.

On a read cycle, the 9610 will drive the data bus with valid data after the leading edge of the CS# signal. The data bus will become tri-state after the trailing edge of CS#.

On a write cycle, the data bus must be valid before the trailing edge of the CS# signal. The data bus must remain valid after the trailing edge CS#.

### 8.1.2 Asynchronous Mode on DMA Interface

If configured for Asynchronous operation, the bus interface is driven internally by the frequency of the CLK input pin.

A typical DMA transfer consists of DREQ# being asserted by the 9610, and then one or more assertions of the DACK# signal by the system bus to initiate one or more DMA transfers. The minimum DMA transfer time is five CLK cycles.

The start of a DMA cycle is controlled by the DMA Controller with the assertion of the DACK# signal. The DREQ# signal must be asserted previous to DACK# being asserted.

On a read cycle, the 9610 will drive the data bus with valid data after the leading edge of the DACK# signal. The data bus will become tri-state after the trailing edge of DACK# .

On a write cycle, the data bus must be valid before the trailing edge of the DACK# signal. The data bus must remain valid after the trailing edge DACK# .
The 9610 will deassert the DREQ# signal after the leading edge of the DACK# signal on the last DMA transfer of a burst.

The source TC# signal indicates the last DMA transfer to the Source FIFO for the current command. The 9610 will assert the TC# signal after the leading edge of

DACK# for the last DMA transfer. The 9610 will deassert the TC# signal after the trailing edge of the DACK# on the last DMA transfer.

The dest TC# signal indicates the last DMA transfer from the Dest FIFO for the current command. The 9610 will assert the TC# signal after the leading edge of DACK# for the last DMA transfer. The 9610 will deassert the TC# signal after the trailing edge of the DACK# on the last DMA transfer. The NOALIGN and BYTEALIGN signals will be valid during the DACK# of the last DMA transfer.

### 8.1.3 Synchronous Mode on CPU Interface

If configured for Synchronous operation, the bus interface is driven by the CLKC signal. All bus signals are relative to the rising edge of the CLKC.

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 activity is identified as Ti.

Synchronous mode supports a minimum bus transfer 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 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.

### 8.1.4 Synchronous Mode on DMA Interface

If configured for Synchronous operation, the bus interface is driven by the CLKA or CLKB signals. All bus signals are relative to the rising edge of the CLKA or CLKB.

Synchronous mode will support a minimum bus transfer time of two clock cycles.

A typical DMA transfer 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 activity is identified as Ti.

The start of a DMA cycle is controlled by the DMA Controller with the assertion of the DACK# signal. The DREQ# signal must be valid on a previous clock (or on the same clock) that the DACK# signal is asserted. There can be any number of clock cycles (including zero) between the assertion of the DREQ# signal and the assertion of the DACK# signal. These will be Ti clock cycles.

A T1 cycle is identified by the assertion of the DACK# signal by the end of a clock. The following clock cycle will be either T2 or Tw based on the value of the DACK# signal at the end of the following cycle. If DACK# is active, then this cycle will be Tw. If DACK# 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 following clock cycle will be either T2 or Tw based on the value of the DACK# signal at the end of the following cycle. If DACK# is active, then this cycle will be Tw. If DACK# 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 DACK# signal at the end of the following cycle. If DACK# is active then this cycle will be T1. If DACK# is inactive then this cycle will be Ti.

There can be any number of Ti cycles (including zero). During a Ti cycle, the data bus will be inactive. The 9610 will deassert the DREQ# signal during the next clock after T1 of the last DMA transfer of a burst.

The source TC# signal indicates the last DMA transfer to the Source FIFO for the current command. The 9610 will assert the TC# signal during the clock following the T1 cycle of the last DMA transfer. TC# will remain asserted through one cycle after the T2 cycle of the last DMA transfer.

The destination TC# signal indicates the last DMA transfer from the Dest FIFO for the current command. The 9610 will assert the TC# signal during the clock following the T1 cycle of the last DMA transfer. TC# will remain asserted through one cycle after the T2 cycle of the last DMA transfer. The NOALIGN and BYTEALIGN signals will be valid whenever the dest TC# signal is active, but not beyond the last T2 cycle

# 9 Electrical Specifications

| | |
|---|---|
| DC Supply Voltage ($V_{DD}$) | -0.3V to +6.V |
| DC Input Voltage | -0.6 to $V_{DD}$ + 0.75V |
| 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 11. Absolute Maximum Ratings**

| | |
|---|---|
| DC Supply Voltage | +4.75V to +5.25V |
| Operating Temperature | 0°C to +70°C |

**Figure 12. Recommended Operating Conditions**

| Symbol | Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| $V_{IL}$ | Low level input voltage | | | | | |
| | TTL | | | | 0.8 | V |
| | Schmitt negative threshold | | 0.8 | 1.0 | | V |
| $V_{IH}$ | High level input voltage | | | | | |
| | TTL | | 2.0 | | | V |
| | Schmitt positive threshold | | | 1.8 | 2.0 | V |
| $V_H$ | Schmitt hysteresis | | | 0.8 | | V |
| $V_{IL(CLK)}$ | Low level CMOS input voltage | $V_{DD}$ = 4.75V | | | 1.4 | V |
| $V_{Ih(CLK)}$ | High level CMOS input voltage | $V_{DD}$ = 5.25V | 3.7 | | | V |
| $I_{IL}$ | Low level input current | $V_{IN} = V_{SS}$ $V_{DD}$ = 5.25V | -10 | | | µA |
| $I_{IH}$ | High level input current | $V_{IN} = V_{DD}$ $V_{DD}$ = 5.25V | | | 10 | µA |
| $V_{OL}$ | Low level output voltage | $I_{OL}$ = 4mA | | 0.2 | 0.4 | V |
| $V_{OH}$ | High level output voltage | $I_{OL}$ = -4mA | 3.3 | 4.2 | | V |
| $I_{OZ}$ | High impedance leakage current | $V_O = V_{SS}$ $V_{DD}$ = 5.25V | -10 | | 10 | µA |
| $I_{DD}$ | Quiescent supply current | | | 10 | | µA |
| $C_{IN}$ | Input capacitance | $V_{DD}$ = 5V | | 3 | | pF |
| $C_{OUT}$ | Output capacitance | $V_{DD}$ = 5V | | 6 | | pF |
| $C_{I/O}$ | Input/Output capacitance | $V_{DD}$ = 5V | | 7 | | pF |
| $P_A$ | Power dissipation | $V_{DD}$ = 5.25V CLK =50MHz | | 0.75 | 1.0 | W |
| $P_A$ | Power dissipation | $V_{DD}$ = 5.25V CLK =20MHz | | 0.3 | 0.4 | W |

**Figure 13. DC Electrical Characteristics**

| Symbol | Parameter | Conditions |
|---|---|---|
| $C_{L1}$ | Load on CPU interface | 50 pF |
| $C_{L2}$ | Load on A bus and B bus | 50 pF, 20* pF |
| $C_{L3}$ | Load on other port pins | 20 pF |
| $V_{DD}$ | Supply voltage | 5V ± 5% |
| $V_{SS}$ | Ground potential | 0V |
| $T_A$ | Ambient operating temperature | 0°C to +70°C |

Alternate timings, see Figure 18 and Figure 20.

**Figure 14. AC Specification Definition**

# 10 Timing Specifications

| Number | Description | Min | Max | Units |
|---|---|---|---|---|
| 1 | Reset width | 10 $t_{CLOCK}$* | | ns |
| 2 | First access after Reset | 12 $t_{CLOCK}$* | | ns |

* $t_{CLOCK}$ represents the slowest clock frequency of CLK, CLKC, CLKA, or CLKB

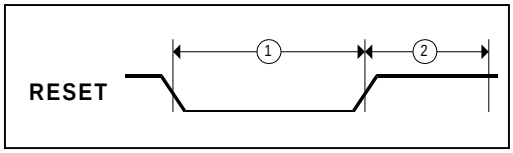



**Figure 15. Reset Timing**

| Number | Description | Min | Max | Units |
|---|---|---|---|---|
| | Oscillator frequency | | 50 | MHz |
| 1 | Clock period (see note 1) | 20 | | ns |
| 2 | Clock width high (see note 2) | 8 | | ns |
| 3 | Clock width low (see note 2) | 8 | | 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 |

Note 1: CLK period is referred to as $t_{CLK}$. CLKC period is referred to as $t_{CLKC}$. CLKA period is referred to as $t_{CLKA}$. CLKB period is referred to as $t_{CLKB}$. Any bus in asynchronous mode will use the CLK signal.

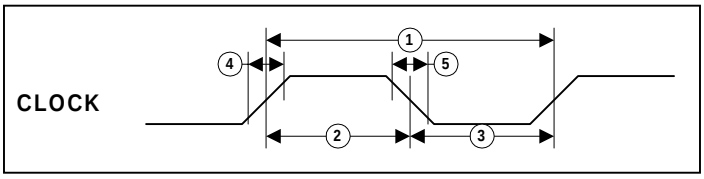



**Figure 16. External Clock**

| Number | Description | Min | Max | Units |
|---|---|---|---|---|
| 1 | Addr setup before CS# active | 3 | | ns |
| 2 | Addr hold after CS# inactive | 5 | | ns |
| 3 | R/W# setup before CS# active | 3 | | ns |
| 4 | CS# active width | $t_{CLKC}$ +7 | | ns |
| 5 | CS# inactive width | 3 $t_{CLKC}$ +7 | | ns |
| 6 | R/W# hold after CS# inactive | 5 | | ns |
| 7 | Data valid after CS# active (read) | | 18 | ns |
| 8 | Data hold after CS# inactive (read) | 5 | 16 | ns |
| 9 | Data setup before CS# inactive (write) | 3 | | ns |
| 10 | Data hold after CS# inactive (write) | 5 | | ns |

**Figure 17. Asynchronous CPU Timing**

**(Diagram on the next page)**

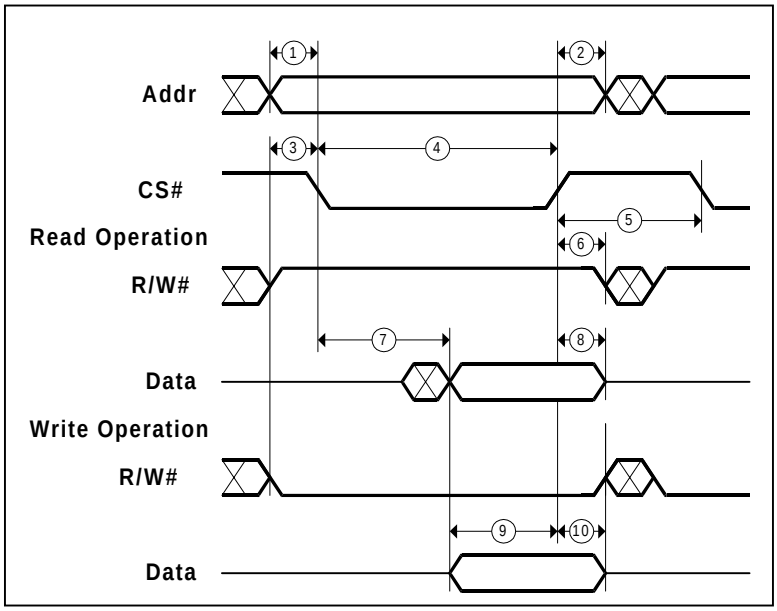



**Figure 17. Asynchronous CPU Timing**

| Number | Description | Min | Max | Units |
|---|---|---|---|---|
| 1 | DACK# active width | $t_{CLKA}$ +7 | | ns |
| 2 | DACK# inactive width | 3 $t_{CLKA}$ +7 | | ns |
| 3 | DREQ# inactive after DACK# active | | 10 | ns |
| 4 | TC# inactive after DACK# inactive | | 4 $t_{CLKA}$ +7 | ns |
| 5 | TC# active after DACK# active | | 10 | ns |
| 6 | Data valid after DACK# active (read) @ 50pF load | | 18 | ns |
| 7 | Data hold after DACK# inactive (read) | 5 | 16 | ns |
| 8 | Data setup before DACK# inactive (write) | 3 | | ns |
| 9 | Data hold after DACK# inactive (write) | 5 | | ns |

Note: References to $t_{CLKA}$ apply to Port A DMA timing. Use $t_{CLKB}$ for Port B timing.

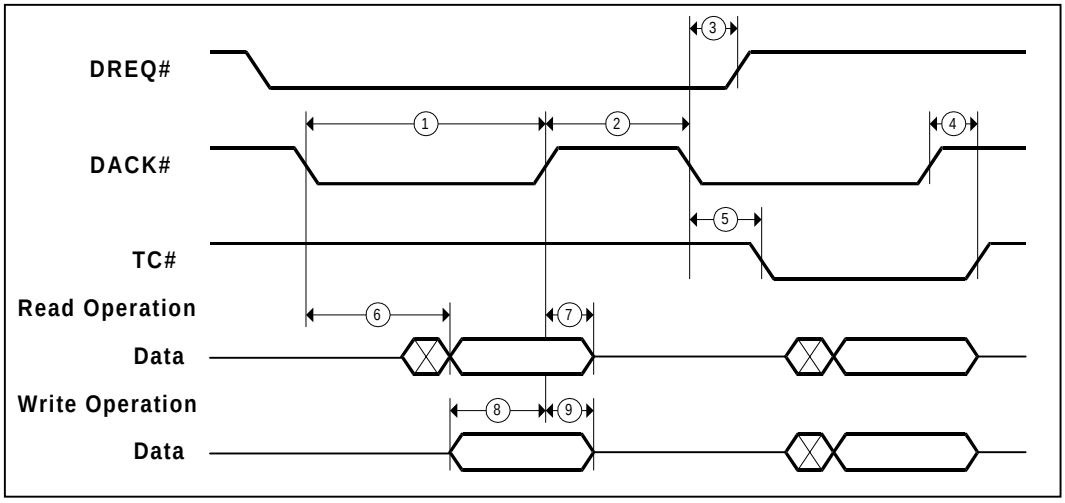



**Figure 18. Asynchronous DMA Timing**

| Number | Description | Min | Max | Units |
|---|---|---|---|---|
| 1 | Addr setup | 5 | | ns |
| 2 | Addr hold | 3 | | ns |
| 3 | CS# setup | 3 | | ns |
| 4 | CS# hold | 3 | | ns |
| 5 | R/w# setup | 5 | | ns |
| 6 | R/w# hold | 3 | | ns |
| 7 | Data output valid delay (read) | | 20* | ns |
| 8 | Data hold (read) | 5 | | ns |
| 9 | Data setup (write) | 3 | | ns |
| 10 | Data hold (write) | 3 | | ns |

* Requires additional CPU wait state at 50MHz CPU clock

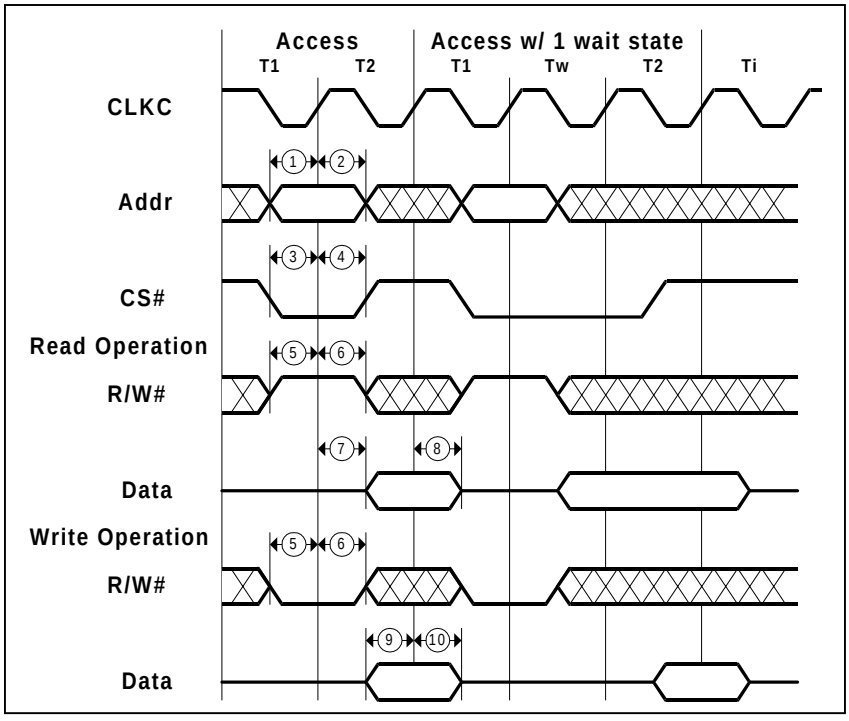



**Figure 19. Synchronous CPU Timing**

| Number | Description | Min | Max | Units |
|---|---|---|---|---|
| 1 | DREQ# output valid delay | | 15 | ns |
| 2 | DREQ# hold | 5 | | ns |
| 3 | DACK# setup | 3 | | ns |
| 4 | DACK# hold | 3 | | ns |
| 5 | TC# output valid delay | | 16 | ns |
| 6 | TC# hold | 5 | | ns |
| 7a | Data output valid delay (read) @ 50pF load | | 18 | ns |
| 7b | Data output valid delay (read) @ 20pF load | | 15 | ns |
| 8 | Data hold (read) | 5 | | ns |
| 9 | Data setup (write) | 3 | | ns |
| 10 | Data hold (write) | 3 | | ns |

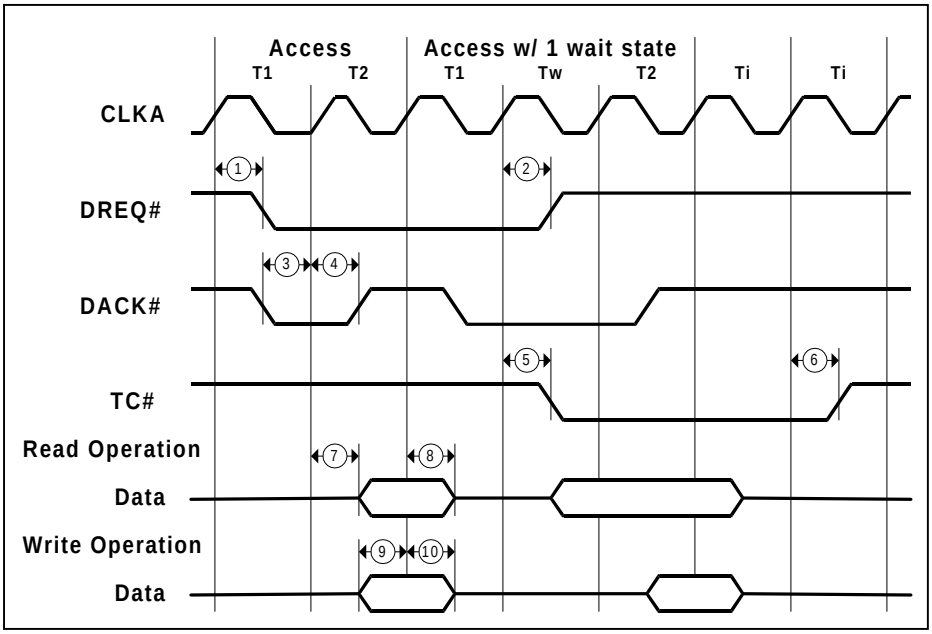



**Figure 20. Synchronous DMA Timing**

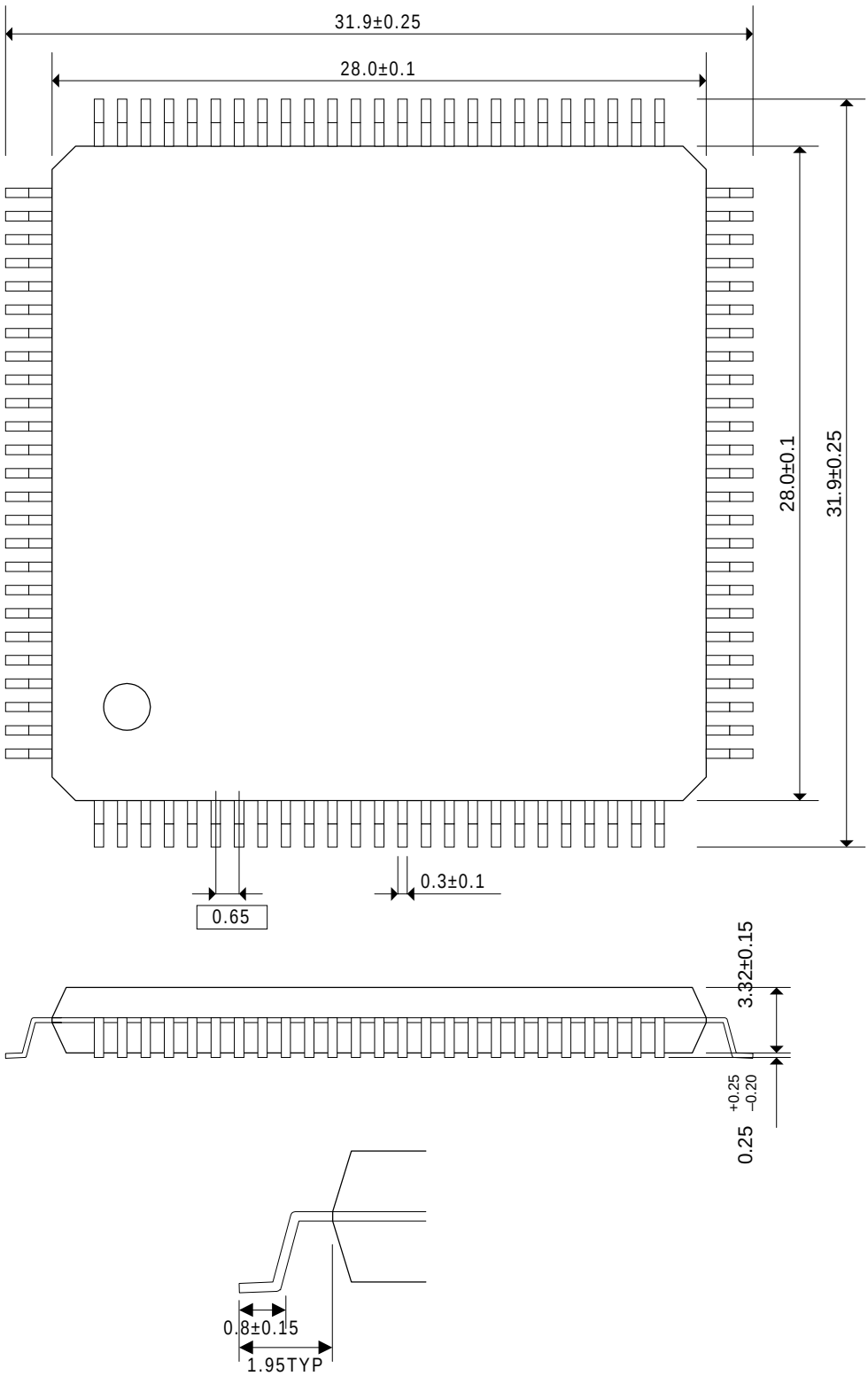



**Figure 21. 144-pin PQFP package**

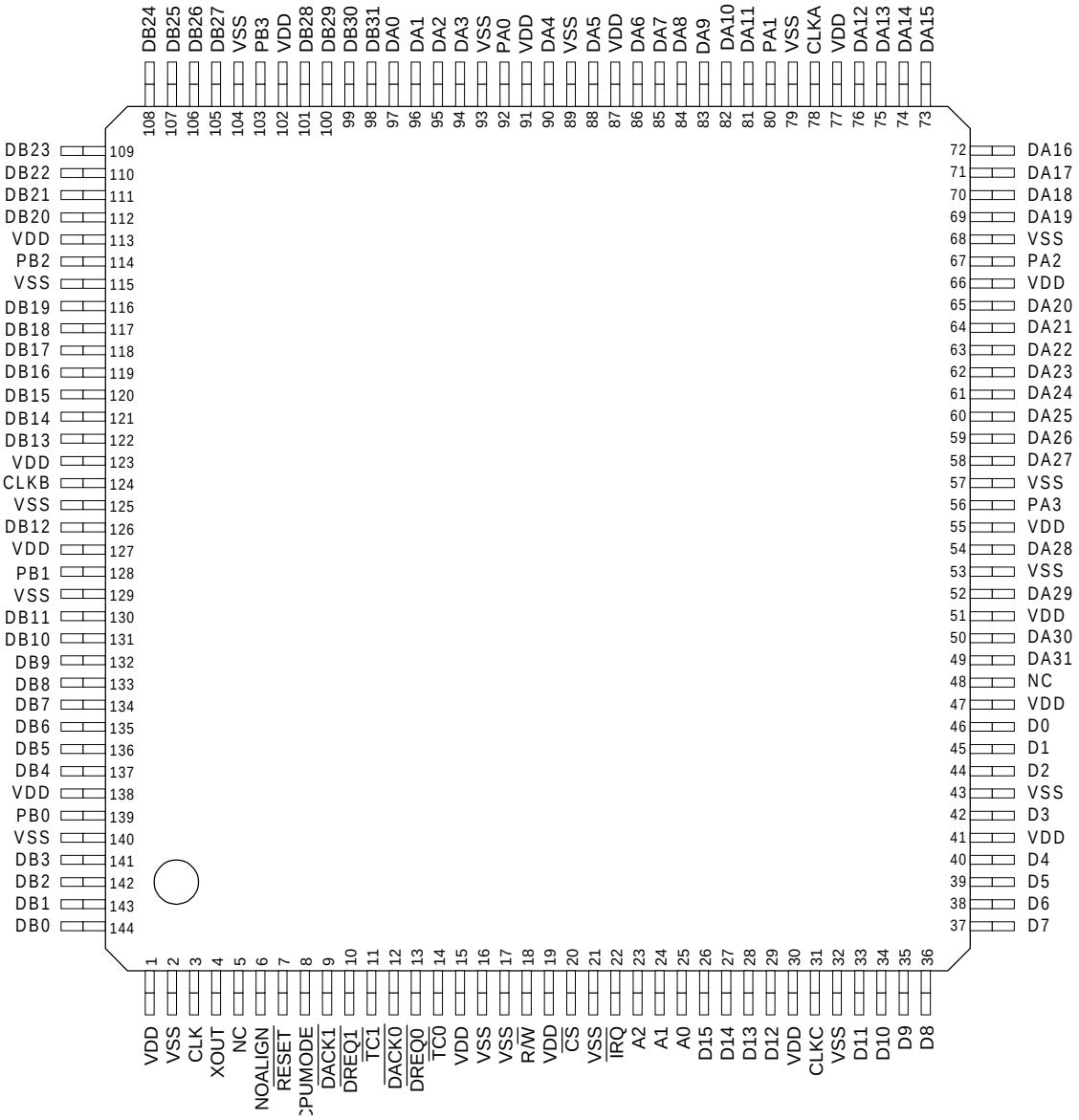



**Figure 22. 144-pin PQFP pinout**