Product Specification

AHA371 / AHA372

PCI Express® Compression and Decompression Accelerator Card

This product is protected by U.S. patents 5,532,693 and 7,609,182 and other patents pending owned by Comtech EF Data Corporation.

WARNING – This document contains U.S.-origin data, the re-transfer of which is restricted by the Export Administration Act and Export Administration Regulations (15 C.F.R. 770, et seq.).

aha371-372_spec_rev_a

AHA
aha products group

Comtech EF Data Corporation
1126 Alturas Drive, Moscow ID 83843
tel: 208.892.5600 - fax: 208.892.5601 - www.aha.com

table of contents

1.0 INTRODUCTION

The AHA371 and AHA372 products are PCI Express® (PCIe) add-in cards that accelerate lossless compression/decompression for networking and storage systems. These cards implement the open standard Deflate compression and decompression algorithm and have support for the GZIP and ZLIB file formats. Deflate is the ideal compression algorithm for networking and storage, since compatible software is readily available on client machines and is included in Internet browser software. Lempel-Ziv-Stac (LZS) compression and decompression are also supported for compatibility with existing systems.

Software compression is inadequate for many systems either because the throughput is too slow or the CPU loading is too high. To remedy this, the AHA371 provides throughput of 10Gbps compression and 10Gbps decompression and the AHA372 provides throughput of 20Gbps compression and 20Gbps decompression. A highly efficient Scatter/Gather DMA engine performs the transfers with low CPU overhead. The cards are PCIe Gen 2 compliant with x8 interface in a low-profile form factor.

1.1 FEATURES

FEATURE	***BENEFIT***
Open Standard Algorithm	• Implements Deflate Algorithm • Supports GZIP file Format • Supports ZLIB format
PCI Express Gen 2 x8 Interface	• Standardized High Speed Serial Interface • Scatter/Gather DMA engine provides low host CPU overhead
Low Profile Form Factor	• Fits in smallest servers and appliances.
High Compression Ratios	• Deflate outperforms LZS, Adaptive Lossless Data Compression (ALDC), and Streaming Lossless Data Compression (SLDC)
Minimal expansion of uncompressible data	• Expansion is usually limited to 5 bytes per Deflate block so uncompressible data does not adversely affect bandwidth or storage requirements.
10 or 20Gbps sustained compression throughput	• The AHA371 uses the AHA3641 Compression Accelerator IC to provide 10Gbps of sustained compression throughput. • The AHA372 uses the AHA3642 Compression Accelerator IC to provide 20Gbps of sustained compression throughput.
10 or 20Gbps sustained decompression throughput	• The AHA371 uses the AHA3641 Compression Accelerator IC to provide 10Gbps of sustained decompression throughput. • The AHA372 uses the AHA3642 Compression Accelerator IC to provide 20Gbps of sustained decompression throughput.
Full Duplex Operation	• Supports simultaneous compression and decompression
Multiple Files	• Supports Compression of intermixed blocks from different files
Integrity check mode	• The decompression cores in the AHA3641 or AHA3642 can optionally be used to decompress and check the output of the compression core for added data integrity protection.

1.2 PERFORMANCE

The AHA371 uses a single AHA3641 compression coprocessor IC to provide a compression throughput of 10Gbps. Each AHA3641 contains 3 high performance compression engines and 3 high performance decompression engines, each operating at 3.33Gbps sustained throughput rate.

The AHA372 uses a single AHA3642 compression coprocessor IC to provide a compression throughput of 20Gbps. Each AHA3642 contains 6 high performance compression engines and 6 high performance decompression engines, each operating at 3.33Gbps sustained throughput rate.

The AHA371 and AHA372 have compression ratios derived from the coprocessor IC. Table 1 shows that the deflate compression ratios of the AHA3641 and AHA3642 are, on average, 83 percent better than LZS with HTML data.

Table 1 Compression Ratios

METHOD	HTML CORPUS[A]	CALGARY CORPUS[B]	CANTERBURY CORPUS[C]
AHA3641/AHA3642	6.25:1	2.85:1	3.63:1
LZS	3.41:1	2.24:1	2.75:1
ALDC	3.20:1	2.10:1	2.68:1

A. *The HTML Corpus is a collection of 1600 "typical" HTML pages*

B. *The Calgray Corpus is a standard file set and can be located and downloaded from the Internet*

C. *The Canterbury Corpus is a standard file set and can be located and downloaded from the internet*

1.3 RELATED DOCUMENTS

- AHA3641/AHA3642 Product Specification
- AHA3XX Series Linux Device Driver API Application Note
- RFC 1950: ZLIB Compressed Data Format Specification version 3.3
- RFC 1951: DEFLATE Compression Data Format Specification version 1.3
- RFC 1952: GZIP File Format Specification version 4.3

1.4 NOTATIONS:

- Hex values are represented with a prefix of "0x", such as Register "0x00".
- Gigabits per second is referred to as Gpbs or Gbits/sec.
- Reserved bits are referred to as "RES" and, in the case of a field within a writeable register, should always be written to 0.
- Signals that end with a "#" symbol are active low.

2.0 DEVICE OPERATION

The AHA371 and AHA372 are PCI Express devices that provide high-speed data compression and decompression to a host system. They are low-profile Generation 2.0 compliant add-in cards that provide full-duplex sustained data rates of up to 10 or 20Gbps respectively. The AHA371 uses the AHA3641 IC which contains three compression/decompression engines, each operating at 3.33Gbps, to achieve an aggregate throughput of 10Gbps for compression and decompression. The AHA372 uses the AHA3642, which contains six 3.33Gbps compression/decompression engines for an aggregate throughput of 20Gbps for each operation.

For full details of how the AHA3641 and AHA3642 ICs operate, refer to the AHA3641/AHA3642 Product Specification.

The AHA Products Group provides the AHA3XX Linux Device Driver, which is a reference device driver suitable for a variety of compression and decompression applications. Source code is available so that it can be modified for specific applications as needed.   For more information on interfacing with the device driver, refer to the AHA3XX Linux Device Driver API Application Note.

2.1 LEDS

There are 6 LEDs on the board that provide status information.

LEDS

LED Name	D#	DESCRIPTION
STATUS_ACT	D3	PCI Express Activity Indicator LED. When illuminated, data is transferring on the PCI Express Interface or the card is actively processing a compression or decompression operation.
STATUS_CON	D2	PCI Express Connection Indicator LED. When illuminated, a PCI Express connection to a host has been established.
STATUS_WIDTH	D1	PCI Express Width Indicator Signal. Valid when the STATUS_CON LED is illuminated. When connected with a PCIe x8 bus, STATUS_WITDH is solidly illuminated. When connected with a PCIe x1 bus, solidly inactive. When connected with a PCIe x4 bus, toggling between active and inactive at 1 Hz.
STATUS_SPEED	D4	PCI Express link speed indicator. Valid when the STATUS_CON LED is illuminated. When STATUS_SPEED is illuminated, the PCIe bus is connected at Gen 2 speed (5GTs). When it is not illuminated, the PCIe bus is connected at Gen 1 speed.
LED_A	D6	Reserved for future use.
LED_B	D5	This RED LED is controllable by software. See Section 2.5 for details.

2.2 DEVICE CONFIGURATION REGISTERS

The AHA371 and AHA372 devices will appear to the host system as an AHA3641 and an AHA3642, respectively, on the PCI-Express bus.   Either device uses a 64-bit PCI Base Address Register (BAR0) to define a memory space in which registers controlling the compression and decompression operations are located. These registers are the configuration registers, and start at the memory location pointed to by Base Address Register 0 + 0x8000. All configuration registers are 32-bits wide. Writes to these registers must write the entire 32-bit register at once. Partial writes less than 32-bits will be ignored.

Configuring the device for operation is discussed in detail in the AHA3641/AHA3642 Product Specification. Certain of the registers have a specific purpose or are used for a specific function on the AHA371 and AHA372 boards that are further covered in sections 0 to 2.5.

2.3 ID REGISTER

The AHA3641 and AHA3642 contain an external ID register which on the AHA371 and AHA372 boards is used to identify board type and hardware revision.

The external ID register is defined for the AHA371 and AHA372 as follows:

2.3.1 ORIGINAL AHA3641/AHA3642 REGISTER: ID REGISTER

Address: BA + 0x9028
Type: READ ONLY

31…12	11	10…8	7…4	3…0
RESERVED	BRD_FAMILY	364x_ID	BRD_TYPE	HW_REV

BRD_FAMILY Board Family – Indicates the family of board. Always 0 for AHA371 and AHA372 boards.

0 An AHA37x series board.

1 A non-AHA37x series board.

364x_ID 364x ID – Used to uniquely identify each 364x IC on boards containing multiple 364x devices. Always 0 for AHA371 and AHA372 boards.

BRD_TYPE Board Type. Represents the 3rd digit in the product ID. The AHA371 BRD_TYPE is 0x1 and the AHA372 BRD_TYPE is 0x2.

HW_REV Hardware Revision. Identifies the hardware revision of the board. 0x0 – A, 0x1 – B, etc. See Ordering Info for more details.

2.4 BOARD MONITORING

Certain environmental parameters can be monitored in real time on the AHA371 and AHA372. The board contains voltage, current, and temperature sensors that can provide the following telemetry in during operation.

PARAMETER	SENSOR	LOCATION
12V External Supply Voltage	LTM4676	
12V External Supply Current	LTM4676	
3.3V External Supply Voltage	LTC2991	
0.95V Regulated Supply Voltage	LTM4676	
0.95V Regulated Supply Current	LTM4676	
0.95V Regulated Supply Power	LTM4676	
2.5V Regulated Supply Voltage	LTM4676	
2.5V Regulated Supply Current	LTM4676	
2.5V Regulated Supply Power	LTM4676	
1.0V Analog Regulated Supply Voltage	LTC2991	
1.0V Analog Regulated Supply Current	LTC2991	
AHA3641/AHA3642 Temperature	LTC2991	PCB SURFACE TEMP 2MM ABOVE AHA IC
Power Supply internal Temperature	LTM4676	LEFT SIDE OF CARD
Ambient Temperature	LTC2991	TOP RIGHT CORNER OF CARD

LTC2991 and LTM4676 devices on the board contain the sensors. They are connected via an SMBus to an SMBus controller in an onboard CPLD. The GPIO0_DATA bus from the AHA3641/AHA3642 device is connected to a register stack in the CPLD in a simple address/data scheme. The register stack allows for control of the SMBus controller and access to the telemetry data.

2.4.1.1 CPLD REGISTER STACK

The SMBus controller must be reset before use.

REGISTER ADDR	REGISTER NAME
0	RESET
1	SMB_CTRL
2	SMB_ADDR
3	SMB_CMD
4	SMB_DATAL
5	SMB_DATAH
6	VERSION

2.4.1.2 RESET REGISTER

Address: 0
READ/WRITE
Reset Value: 0x00

7…1	0
RESERVED	RST_N

RST_N is the active low reset for the SMBus controller. The power-up default will be to hold the controller in reset. RST_N must be written to '1' prior to performing any SMBus operations.

2.4.1.3 SMB_CTRL REGISTER

Address: 1
READ/WRITE
Reset Value: 0x00

7	6…4	3	2	1	0
SMB_ERR	RESERVED	SMB_DONE	WORD_MODE	RD_WRN	SMB_REQ

SMB_ERR SMBus Error Flag – Read-only, valid after an SMBus operation when SMB_DONE is '1'. If the controller encounters and error during the SMBus operation then SMB_ERR will be '1'.

SMB_DONE SMBus Operation Done – Read-only. The SMBus controller will set this to '1' when an SMBus operation is complete. After setting SMB_REQ, the application needs to poll this bit to determine when the operation is complete.

WORD_MODE Word Mode Select. The application must set this before starting an SMBus operation to indicate the data size for the operation. Set to '1' for a 16 bit read or write operation or '0' for an 8 bit read or write operation.

RD_WRN Read or Write Mode Select. The application must set this before starting an SMBus operation to indicate the direction of data transfer. Set to '1' for a read operation or '0' for a write operation

SMB_REQ SMBus Request. The application must set this to begin an SMBus operation. When the operation is complete, the SMBus controller will set the SMB_DONE bit.

2.4.1.4 SMB_ADDR REGISTER

Address: 2
READ/WRITE
Reset Value: N/A

7	6…0
RESERVED	ADDR

The application must set ADDR to the SMBus slave device address before starting an SMBus operation.

2.4.1.5 SMB_CMD REGISTER

Address: 3
READ/WRITE
Reset Value: N/A

7…0
CMD

The application must set CMD to the command to be sent to the SMBus slave device before starting an SMBus operation.

2.4.1.6 SMB_DATAL REGISTER

Address: 4
READ/WRITE
Reset Value: N/A

7…0
DATAL

For a write operation, the application must write the first 8 bits of data to be written into DATAL before starting a write operation. For a read operation, DATAL will contain the first 8 bits of data read from the slave device after an SMBus read operation has completed.

2.4.1.7 SMB_DATAH REGISTER

Address: 5
READ/WRITE
Reset Value: N/A

7…0
DATAH

Only used for 16 bit SMBus operation (SMB_CTRL.WORD_MODE = ‘1’). For a write operation, the application must write the second 8 bits of data to be written into DATAH before starting a write operation. For a read operation, DATAH will contain the second 8 bits of data read from the slave device after an SMBus read operation has completed.

2.4.1.8 VERSION

Address: 5
READ ONLY
Reset Value: VER

7…0
VER

Contains an 8-bit Firmware Version.

2.4.2 CPLD READ OPERATION

1) Write GPIO0_DATA Register to 0x0000E<XXX>, where <XXX> is the Address of the CPLD register to be read.
2) Write GPIO0_DIR Register to 0x0000FFFF.
3) Write GPIO0_DATA Register to 0x00006<XXX>, where <XXX> is the Address of the CPLD register to be read.
4) Write GPIO0_DIR Register to 0x0000FF00.
5) Write GPIO0_DATA Register to 0x00002000.
6) Read GPIO0_DATA Register. The value in 7:0 is the 8-bit CPLD register value.
7) Write GPIO0_DATA Register to 0x00006000.

2.4.3 CPLD WRITE OPERATION

1) Write GPIO0_DATA Register to 0x0000E<XXX>, where <XXX> is the Address of the CPLD register to be written.
2) Write GPIO0_DIR Register to 0x0000FFFF.
3) Write GPIO0_DATA Register to 0x00006<XXX>, where <XXX> is the Address of the CPLD register to be written.
4) Write GPIO0_DATA Register to 0x000020<XX>, where <XX> is the 8-bit data to be written to the CPLD register.
5) Write GPIO0_DATA Register to 0x000060<XX>, where <XX> is the 8-bit data to be written to the CPLD register.
6) Write GPIO0_DATA Register to 0x00006000.
7) Write GPIO0_DIR Register to 0x0000FF00.

2.4.4 SMBUS READ OPERATION

1) Write CPLD register SMB_ADDR with the SMBus slave device address to read from.
2) Write CPLD register SMB_CMD with the command to send to the slave device.
3) Write CPLD register SMB_CTRL with 0x03 for an 8-bit read, or 0x07 for a 16-bit read.
4) Poll bit 3 (SMB_DONE) of the CPLD register SMB_CTRL and wait for it to be set to ‘1’.
5) Check bit 7 (SMB_ERR) of the CPLD register SMB_CTRL. If set to ‘1’, then read operation failed.
6) Read the first byte of data from CPLD register SMB_DATAL.
7) If the operation was a 16 bit read, then read the second byte of data from the CPLD register SMB_DATAH.

2.4.5 SMBUS WRITE OPERATION

1) Write CPLD register SMB_ADDR with the SMBus slave device address to write to.
2) Write CPLD register SMB_CMD with the command to send to the slave device.
3) Write CPLD register SMB_DATAL with the first byte of data to write.
4) If the operation is a 16 bit write, then write CPLD register SMB_DATAH with the second byte of data to write.
5) Write CPLD register SMB_CTRL with 0x01 for an 8-bit write, or 0x05 for a 16-bit write.
6) Poll bit 3 (SMB_DONE) of the CPLD register SMB_CTRL and wait for it to be set to ‘1’.
7) Check bit 7 (SMB_ERR) of the CPLD register SMB_CTRL. If set to ‘1’, then write operation failed.

2.4.6 SMBUS MONITORING DEVICE REGISTER STACK

REGISTER NAME	R/W	SMB_ADDR	SMB_CMD	WORD/BYTE	PAGE
LTC2991_EN	W	0x48	0x01	B	N/A
LTC2991_CTRL1	W	0x48	0x06	B	N/A
LTC2991_CTRL2	W	0x48	0x07	B	N/A
LTC2991_CTRL3	W	0x48	0x08	B	N/A
TU1	R	0x48	0x0a	W	N/A
V10	R	0x48	0x16	W	N/A
V10_SENSE	R	0x48	0x18	W	N/A
TBOARD	R	0x48	0x1a	W	N/A
V33	R	0x48	0x1c	W	N/A
LTM4676_PAGE	W	0x4f	0x00	B	N/A
V12	R	0x4f	0x88	W	N/A
I12	R	0x4f	0x89	W	N/A
V095	R	0x4f	0x8b	W	0
V25	R	0x4f	0x8b	W	1
I095	R	0x4f	0x8c	W	0
I25	R	0x4f	0x8c	W	1
LTM4676_TEMP	R	0x4f	0x8e	W	N/A
P095	R	0x4f	0x96	W	0
P25	R	0x4f	0x96	W	1

2.4.7 INITIALIZING MONITORING DEVICES

The monitoring devices must be initialized before monitoring data can be read.

1) Write SMBus register LTC2991_CTRL1 with 0x02.
2) Write SMBus register LTC2991_CTRL2 with 0x00.
3) Write SMBus register LTC2991_CTRL3 with 0x10.
4) Write SMBus register LTC2991_EN with 0x98.

2.4.8 SETTING PAGE NUMBER

Before a value can be read from a register with a page number specified, the page number must be written into the SMBus register LTM4676_PAGE.

2.4.9 TEMPERATURE MONITORING

The application can monitor the following temperatures:

Board temp Temperature near the right side of the board. Read from the TBOARD register.

AHA364x temp Temperature of the board near the AHA3641 or AHA3642 device. Read from the TU1 register.

Power supply temp Internal temperature of the main voltage regulator, P3. Read from the LTM4676_TEMP register.

The registers TBOARD and TU1 have the following format:

DATAL

7	6…5	4…0
DV	RESERVED	D[12…8]

DATAH

7…0
D[7…0]

DV Data valid. Set to '1' when a new temperature value has been written into the register, reset to '0' after the register has been read.

D Temperature data in two's compliment format. The temperature in Celsius is D/16.

The register LTM4676_TEMP has the following format:

DATAL

7…0
Y[7…0]

DATAH

7…3	2…0
N	Y[10…8]

N and Y are both two's compliment integers and the temperature in Celsius is $Y*2^N$.

2.4.10 VOLTAGE MONITORING

The application can monitor the following voltages:

PCIe12V 12V power from PCIe bus. Read from the V12 register.

PCIe3.3V 3.3V power from PCIe bus. Read from the V33 register.

VDD2.5V 2.5V output from on-board regulator. Read from the V25 register.

VDD1.0V 1.0V output from on-board regulator. Read from the V10 register.

VDD1.0V_SENSE 1.0V supply on load side of current sensing resistor, used to calculate the current draw on the 1.0V supply. Read from the V10_SENSE register.

VDD0.95V 0.95V output from on-board regulator. Read from the V095 register.

The registers V33, V10, and V10_SENSE have the following format:

DATAL

7	**4…0**
DV	D[14…8]

DATAH

7…0
D[7…0]

DV Data valid. Set to ‘1’ when a new value has been written into the register, reset to ‘0’ after the register has been read.

D Voltage data in two’s compliment format. The measured voltages for VDD1.0V and VDD1.0V_SENSE are $D*2.5/2^{13}$. The measured voltage for PCIe3.3V is $2.5+D*2.5/2^{13}$.

The register V12 has the following format:

DATAL

7…0
Y[7…0]

DATAH

7…3	**2…0**
N	Y[10…8]

N and Y are both two’s compliment integers and the measured voltage is $Y*2^{N}$.

The registers V25 and V095 have the following format:

DATAL

7…0
Y[7…0]

DATAH

7…0
Y[15…8]

Y is a two’s compliment integer and the measured voltage is $Y*2^{-12}$.

2.4.11 CURRENT MONITORING

The application can monitor current on the following power rails:

PCIe12V 12V power from PCIe bus. Current can be read from the I12 register.

VDD2.5V 2.5V output from on-board regulator. Current can be read from the I25 register.

VDD1.0V 1.0V output from on-board regulator. Current can be calculated using (VDD1.0V – VDD1.0V_SENSE)/0.025.

VDD0.95V 0.95V output from on-board regulator. Current can be read from the I095 register.

The registers I12, I25, and I095 have the following format:

DATAL

7…0
Y[7…0]

DATAH

7…3	2…0
N	Y[10…8]

N and Y are both two's compliment integers and the measured current is $Y*2^N$.

2.4.12 POWER MONITORING

The application can calculate current on each power rail by multiplying measured voltage times current, or power in watts can be read directly from the hardware for the following power rails:

VDD2.5V 2.5V output from on-board regulator. Power can be read from the P25 register.

VDD0.95V 0.95V output from on-board regulator. Power can be read from the P095 register.

The registers P25 and P095 have the following format:

DATAL

7…0
Y[7…0]

DATAH

7…3	2…0
N	Y[10…8]

N and Y are both two's compliment integers and the measured power in watts is $Y*2^N$.

2.4.13 ORIGINAL AHA3641/AHA3642 REGISTER: GPIO0_DATA

Address: BA + 0x9008
READ/WRITE
Reset Value: 0x00000000

31…16	15	14	13	12	11:8	7:0
RES	ALE	RD#	WR#	RES	A	AD

ALE Address Latch Enable. Active High. When active, the address presented on the combined A and AD busses is captured by the address latch in the monitoring device.

RD# Read Enable. Active Low. When active, the monitoring device drives the value stored in the register location specified by the address latch on to the AD bus.

WR# Write Enable. Active Low. When active, the monitoring device writes the value from the AD bus into the register at the address specified by the address latch.

A Address Bus. The upper 5 bits of the address bus to the address latch.

AD Address/Data Bus. When ALE is active, the lower 8 bits of the address bus to the address latch. Otherwise, the data bus.

2.4.14 ORIGINAL AHA3641/AHA3642 REGISTER: GPIO0_DIR

Address: (BA + 0x900C)
READ/WRITE
Reset Value: 0x00000000

31…16	**15…0**
RES	DIR[15:0]

DIR[x] GPIO Pin x Direction – Selects GPIO pin direction: input (0) or output (1). Bits 15:8 must always be set as output. Bits 7:0 should be configured as output when writing an address or when writing data to the monitoring device. Bits 7:0 should be configured as input when reading data from the monitoring device.

2.5 LED_B

LED_B provides a software controllable status indicator. LED_B is connected to GPIO1[15]. When GPIO1[15] is configured as an output with a value of '0', then LED_B is illuminated. When GPIO1[15] is configured as an input or as an output with a value of '1', then LED_B is off.

2.5.1 ORIGINAL AHA3641/AHA3642 REGISTER: GPIO1_DATA

Address: (BA + 0x90010)
READ/WRITE
Reset Value: 0x00000000

31…16	**15**	**14…0**
RES	LED	RES

LED Illuminate LED. Active Low. When active and configured as an ouput, LED_B will be illuminated on the board.

2.5.2 ORIGINAL AHA3641/AHA3642 REGISTER: GPIO1_DIR

Address: (BA + 0x9014)
READ/WRITE
Reset Value: 0x00000000

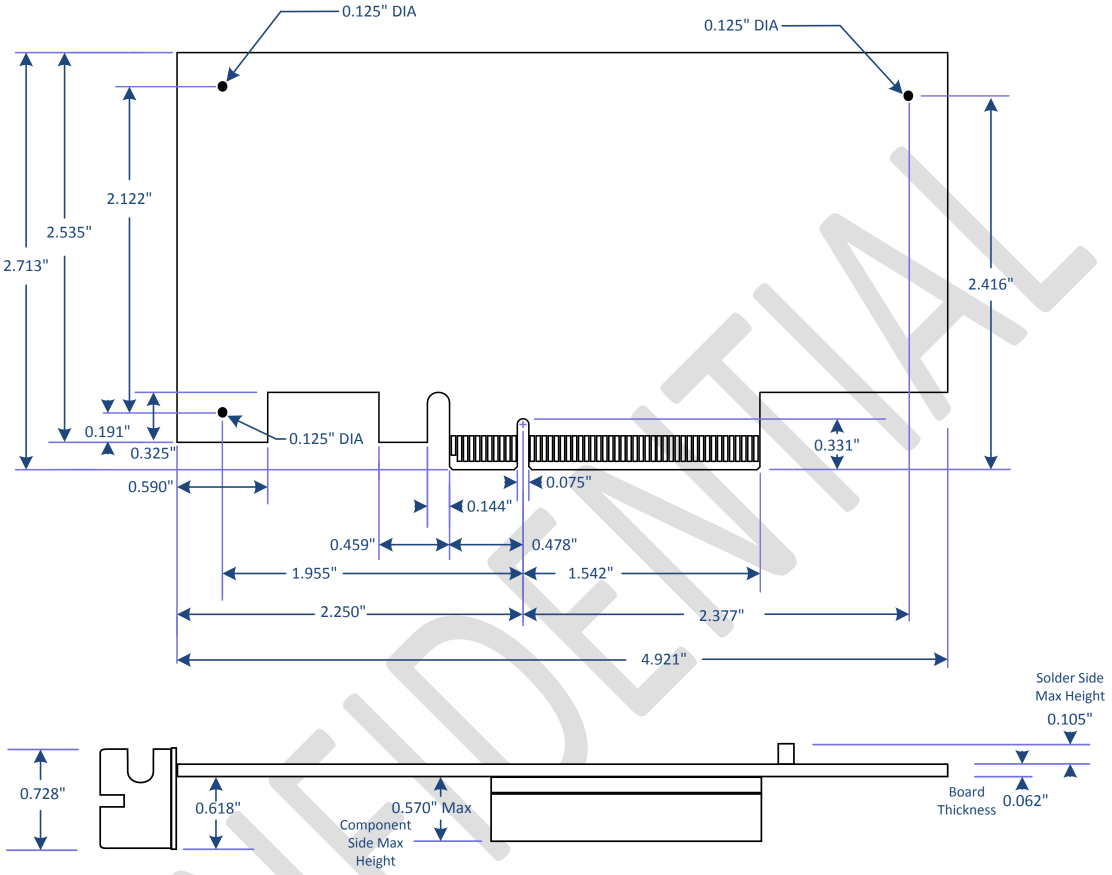
31…16	**15**	**14…0**
RES	LED_DIR	RES

LED_DIR LED Pin Direction – Selects pin direction: input (0) or output (1) Bit 15 should be configured as output when software control of LED_B is desired.

Note: Bits 31:16 and 14:0 are RESERVED and must always be set to 0.

3.0 MECHANICAL

The AHA371 and AHA372 cards comply with the PCI Express CEM 2.0 standard for a low-profile PCIe x8 add in card, though the cards length of 4.921” is shorter than the 6.600” allowed by the standard.



All Dimensional Tolerances: +/- 0.005"

4.0 OPERATING TEMPERATURE RANGE

The AHA371 and AHA372 will operate under the following ambient temperature conditions:

MIN	MAX
0^{o}C	50^{o}C

5.0 AIR FLOW REQUIREMENTS

The AHA371 and AHA372 require the following air flow over the top of the board. If not provided with the specified airflow while powered, the board can be permanently damaged.

MIN
100 LFM

6.0 POWER REQUIREMENTS

The AHA371 and AHA372 draws power from the 12V supply and 3.3V supply on the PCIe bus. The power dissipation is as follows:

IDLE	FULL THROUGHPUT
13 W	16 W

7.0 ENVIRONMENTAL

The AHA371 and AHA372 are fully RoHS compliant.

8.0 ORDERING INFORMATION

8.1 PARTS NUMBERING

MANUFACTURER	DEVICE NUMBER	HARDWARE REVISION	FIRMWARE REVISION
AHA	372	A	01
AHA	371	A	01

Device Number:
372 – 20Gbps
371 – 10Gbps

Example:
AHA371A01

9.0 ABOUT AHA

The AHA Products Group (AHA) of Comtech EF Data Corporation develops and markets superior integrated circuits, boards, and intellectual property cores for improving the efficiency of communications systems everywhere. AHA has been setting the standard in Forward Error Correction and Lossless Data Compression for many years and provides flexible and cost effective solutions for today's growing bandwidth and reliability challenges. Comtech EF Data is a wholly owned subsidiary of Comtech Telecommunications Corporation (NASDAQ: CMTL). For more information, visit: www.aha.com.