USER MANUAL

THERMOCOOL™
PC/104 ENVIRONMENTAL FAN CARD II

2/11/2005

MNL-0493C-01

[This page left blank intentionally]

PC/104 ENVIRONMENTAL FAN CARD II
Table of Contents

INTRODUCTION

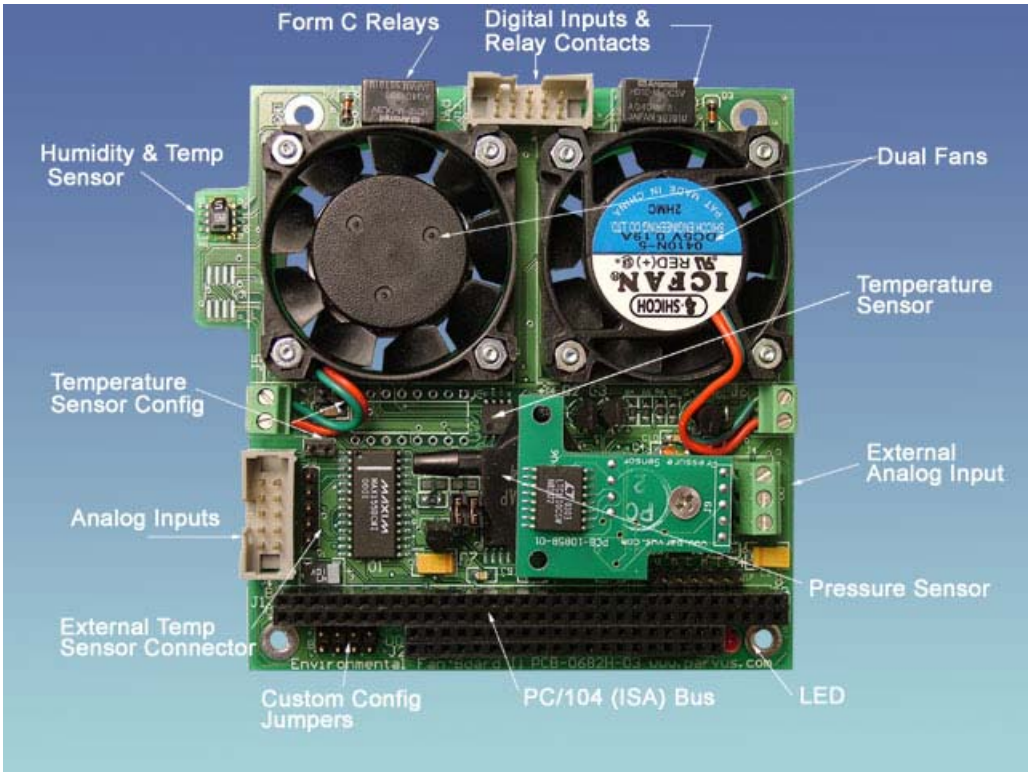
About PC/104 Environmental Fan Card II

The **PC/104 Environmental Fan Card II** intelligently monitors and controls the environmental characteristics of an enclosed PC/104 computer system through a series of pressure, humidity and temperature sensors working in concert with onboard fans to circulate internal air.

In a thermostatic function, the board's two ball-bearing fans turn on and off, responding to changing temperature conditions while conserving power when not in use. In a sealed enclosure, the board can reduce or eliminate the need for outside air exchange. Air is driven in a push-pull circular cooling pattern around the board and adjacent PC/104 modules. Each fan can alternatively be reversed (flipped over) to create a push-push or pull-pull movement of air.

Besides providing thermal management, the Environmental Fan Card II integrates capacitive humidity and atmospheric pressure sensors, which can be used as alarm indicators of moisture and pressure changes within an enclosed PC/104 system. The board's humidity sensor measures from 1-99% relative humidity with instantaneous desaturation. Similarly, the pressure sensor delivers a highly accurate, linear voltage output directly proportional to applied pressure (0-29 PSI) - with precise span and offset calibration, as well as temperature compensation. For applications not requiring a pressure sensor, the board is also available without its pressure sensor mezzanine.

Other features include two Form C relays, two digital inputs, four analog I/O inputs, elapsed time counter (ETC) counter, LED indicator and configuration jumpers to enable the module to control external devices and integrate into a variety of complex environmental management schemes. Four (4) kb of onboard EEPROM provides the board with a unique serial number (for identification in a



system) and permanent storage for configuration or maintenance information. The hour counter records the number of times the unit has been turned on and off (power cycles).

Features

16-bit PC/104 Bus
Two Form C Relays
Digital Thermometer Thermostat
Relative Humidity Sensor
(PRV-0509x-06 Only)
Air Pressure Sensor
(PRV-0509x-06 Only)
Configuration Jumpers
Dual Fans
LED Indicator
Two 10-pin Headers (8 Analog Inputs, 2 Digital Inputs
3-pin Screw Clamp Terminal (for Remote Management)
Xilinx® Programmable Logic

About PC/104

The PC/104 specification is characterized by its small form-factor (3.550" x 3.775"), stackable 104-pin/socket ISA bus connector, and reduced bus signal drive, making PC/104's size, durability, expandability, reliability, quality, and power consumption ideal for embedded computing. PC/104 technology leverages the same readily available development tools used with personal desktop computers to dramatically improve time-to-market for embedded systems development. The full PC/104 specification can be found at the PC/104 Consortium Web site:
http://www.pc104.org/technology/pc104_tech.html

INSTALLATION

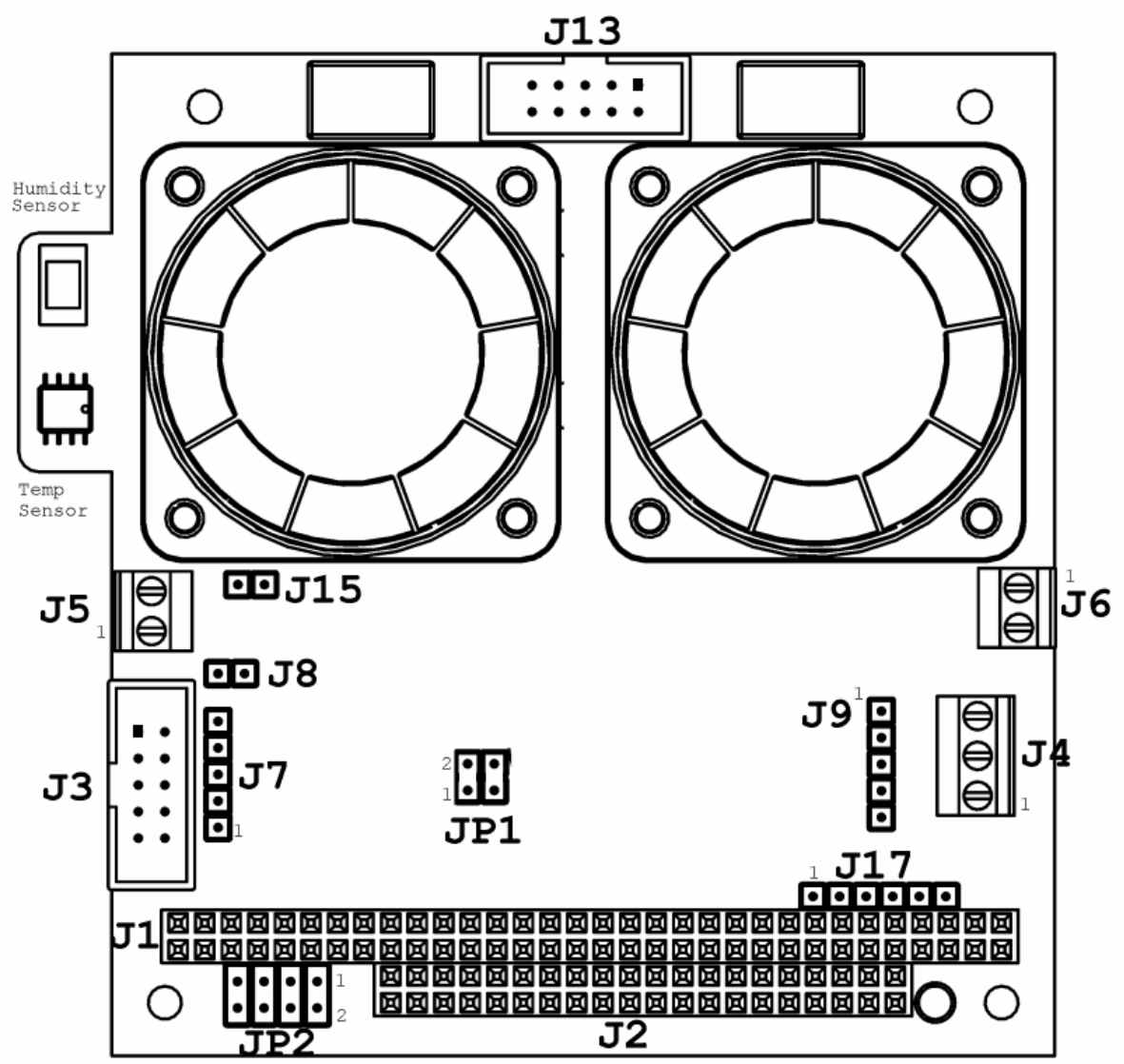
Hardware Installation

Remove the card from the anti-static bag. Make any desired connections such as Analog input (remote temperature sensing), Digital input/output, or add an external temperature sensor. The I/O address for the card selected by JP1 should not be set to conflict with anything else in the system, and should be recorded if the software is going to be used. Install the card onto the PC/104 stack. Ensure that JP2 is open across all eight pins. This is a jumper used in the manufacturing of the module.

Software Installation

Use the included software to set the high and low temperatures for fan activation, which are then stored in an on-board EEPROM. Customization of the software is easily achieved for applications requiring it. Contact Parvus for more information in this area.

CONNECTION



Component and Connector Placement

Note: All dimensions are in inches

Con #	Function	Con #	Function	Con #	Function
J1,J2	PC/104 Bus	J3	Analog Input	J4	External Analog Input
J5/J6	Fan Power	J7	External Temp Sensor	J8	Temperature Sensor Configuration
J9	Pressure Sensor Interface	J13	Digital I/O & Relay Contacts	J15	
J17	FPGA Programming Port	JP1	I/O Address Select	JP2	Custom Configuration Jumpers

Connector Pinouts

J1/J2: PC/104 Bus

Pin	Row A	Row B	Row C	Row D
0	--	--	GND	GND
1	/IOck	GND	/SBHE	/MCS16
2	SD7	Rstdrv	LA23	/IOCS16
3	SD6	+5v	LA22	IRQ10
4	SD5	IRQ9	LA21	IRQ11
5	SD4	-5v	LA20	IRQ12
6	SD3	DRQ2	LA19	IRQ15
7	SD2	-12v	LA18	IRQ14
8	SD1	/Exfer	LA17	/DA0
9	SD0	+12v	/MR	DRQ0
10	IOrdy	(key)	/MW	/DA5
11	AEN	/SMW	SD8	DRQ5
12	SA19	/SMR	SD9	/DA6
13	SA18	/IOW	SD10	DRQ6
14	SA17	/IOR	SD11	/DA7
15	SA16	/DA3	SD12	DRQ7
16	SA15	DRQ3	SD13	+5v
17	SA14	/DA1	SD14	/MSTR
18	SA13	DRQ1	SD15	GND
19	SA12	/Rfrsh	(key)	GND
20	SA11	SCLK	--	--
21	SA10	IRQ7	--	--
22	SA9	IRQ6	--	--
23	SA8	IRQ5	--	--
24	SA7	IRQ4	--	--
25	SA6	IRQ3	--	--
26	SA5	/DA2	--	--
27	SA4	TC	--	--
28	SA3	BALE	--	--
29	SA2	+5v	--	--
30	SA1	OSC	--	--
31	SA0	GND	--	--
32	GND	GND	--	--

The PC/104 bus always uses Row A and B, while Row C and D are for 16 bit systems. B10 and C19 are keyed locations.

J3: Analog Input
These analog inputs are converted to a 12-bit signal from 0-5V DC.

Pin	Function	Pin	Function
1	+5V	6	Analog Input 4
2	Analog Input 0	7	Analog Input 5
3	Analog Input 1	8	Analog Input 6
4	Analog Input 2	9	Analog Input 7
5	Analog Input 3	10	Ground

J4: External Analog Input	
Pin	Function
1	+5V
2	Analog Input 7
3	Ground

J5: Fan Power	
Pin	Function
1	+5V
2	Fan Control

J6: Fan Power	
Pin	Function
1	Fan Control
2	+5V

J7: External Temperature Sensor (DS1620 Only)			
Pin	Function	Pin	Function
1	Ground	4	Clock
2	+5V	5	Reset
3	DQ		

J9: Pressure Sensor Interface			
Pin	Function	Pin	Function
1	+5V	4	SPR2
2	SPR1	5	Ground
3	Analog Input 3		

J13: Digital I/O & Relay Contacts			
Pin	Function	Pin	Function
1	+5V	6	Normally Closed 1
2	Digital 0	7	Normally Open 2
3	Digital 1	8	Common 2
4	Normally Open 1	9	Normally Closed 2
5	Common 1	10	Ground

J17: FPGA Programming Port			
Pin	Function	Pin	Function
1	+5V	4	TMS
2	TD_OUT	5	TD_IN
3	TCK	6	Ground

CONFIGURATION

JP1 – I/O Address Select

This jumper block selects I/O space of the board. All registers are offsets from this base address.

Pins 1-2	Pins 3-4	Address
Short	Open	320-327
Open	Short	340-347
Short	Short	360-367

JP2 – Custom Configuration Jumpers

These four jumpers may be used as custom configuration jumpers and may be read through register +5. The inputs read 0 with a jumper installed. See the programming section for more detail.

J8/J15 – Temperature Sensor Configuration

An external temperature sensor can be applied to the environmental board. This sensor must be a DS1620 part. Two jumpers are used to indicate the presence of a secondary, external sensor. J8 is installed for no external sensor and removed if an external sensor is present. J15 is only installed if the primary sensor is not installed on the PCB but is remote.

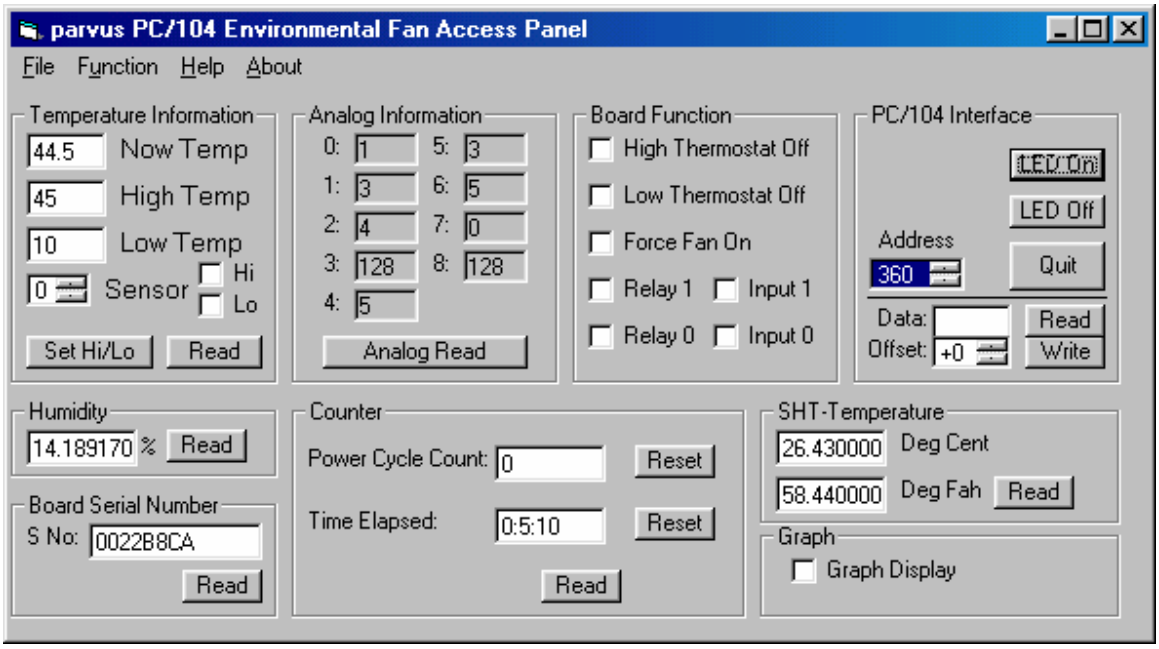
One Primary Sensor, On-board J8-Installed, J15-Removed
One Primary Sensor, Off-board J8-Removed, J15-Installed
Two Sensors, Primary and Secondary J8-Removed, J15-Removed

SOFTWARE UTILITY

This software utility is provided to give the user a functional tool that will operate with the Environmental Fan card. This is the same program used by Parvus to test the product during manufacturing. This tool allows the user to verify the functionality of the board prior to, and anytime during, development of custom software. In most case the implementation of these cards into a system will require the user to create their own custom application software. This software utility and source code can be downloaded from http://tech.parvus.com/

With this utility, the user can read the data from all the sensors (temperature, humidity and pressure), read the analog inputs, control the fans and relays, and graph the sensor inputs.

A screen shot of the running program is shown below.



The program window is divided into four sections: Temperature Information, Analog Information, Board Information, and PC/104 Interface.

Temperature Information

The temperature information is displayed in Celsius.

Now Temp – Displays the current temperature reading of the selected temperature sensor.

High Temp – Displays the high temperature limit of the selected sensor. Above this temperature the fans come on. To update this setting, make the change in the text box and click on 'Set Hi/Lo'. When this temperature is reached, a checkbox appears in the box next to 'Hi'.

Low Temp – Displays the low temperature limit of the selected sensor. Below this temperature the fans come on. To update this setting, make the change in the text box and click on 'Set Hi/Lo'. When this temperature is reached, a checkbox appears in the box next to 'Lo'.

Sensor – This menu selects which of the two temperature sensor data is displayed. '0' selects the onboard sensor and '1' selects the optional external sensor.

Set Hi/Lo – Used to save the High/Low temperature to the temperature sensor.

Read – Queries the temperature sensor and display its data. To continuously read the information, select 'Continuous Scan' from the 'Function' menu.

Analog Information

Displays the analog value of the inputs of the corresponding channel. The fields can be updated by pressing 'Analog Read' or by selecting 'Continuous Scan' from the 'Function' menu.
Note: Channel 3 or Channel 8 is used for the pressure sensor (if installed).

Board Function

This interface allows direct control over the fans, relays and digital I/O. Clicking 'Humidity' brings up a new window that displays the relative humidity.

PC/104 Interface

This interface is used to change the base address of the registers and to manipulate the contents of the registers directly.

Humidity

The relative humidity is displayed as a percentage. The range is 0 to 100 percent. Press ‘Read’ to display the relative humidty or select ‘Continuous Scan’ from the ‘Function’ menu. Refer to the SHT1x/SHT7x datasheet for more information.

Board Serial Number

Each Environment Fan board has a unique 6 byte serial number. The board’s serial number is displayed only after ‘Read’ is pressed.

Counter

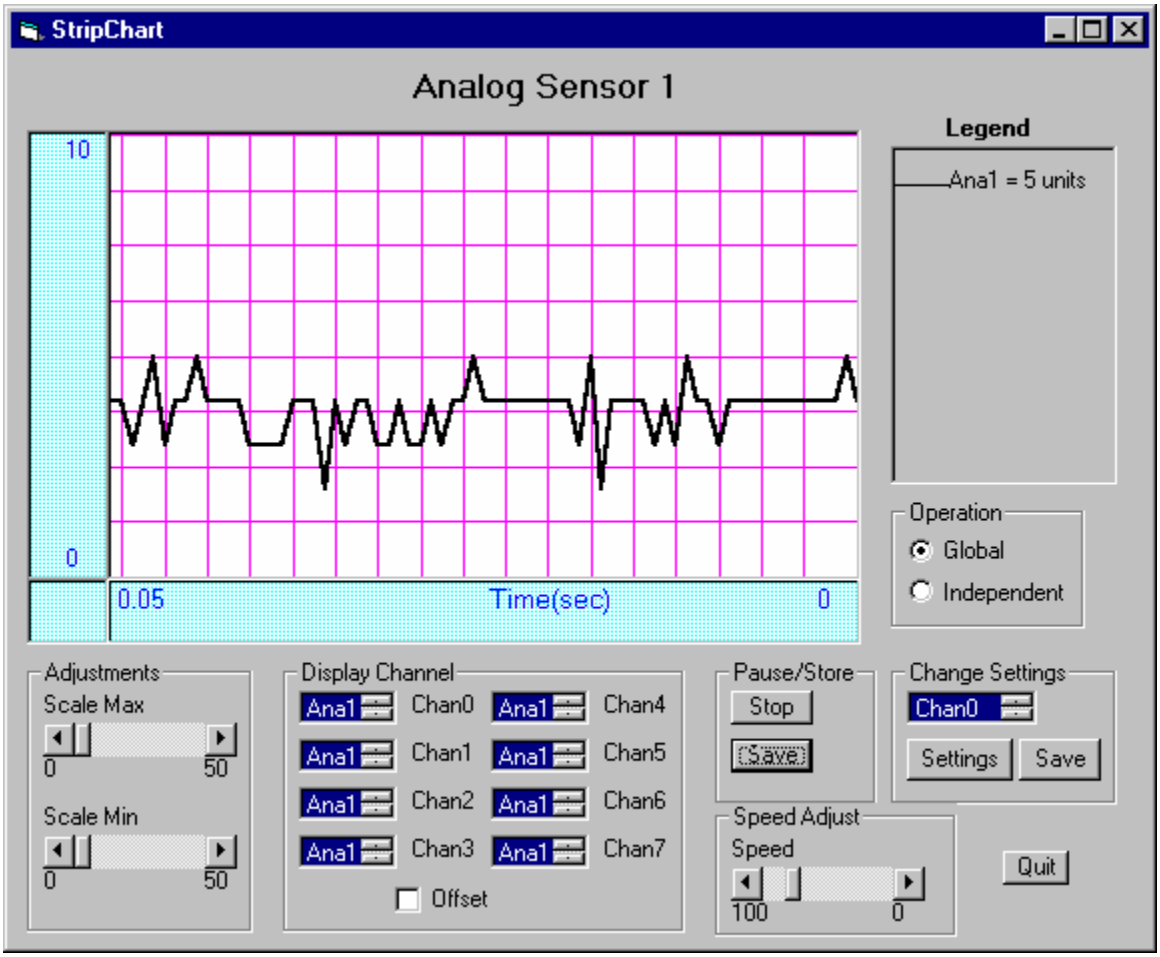
The Environmental Fan board can track how many power cycles have occurred. ‘Reset’ sets this value to zero. The Environmental Fan board can also track the elapsed time since it was last reset. Time is displayed in seconds through the software but is stored in quarter second intervals. Range is from 0 to 1073741823 seconds. Select ‘Read’ to read the current values for each or select ‘Continuous Scan’ form the ‘Function’ menu. Refer to the DS1682 documentation for more information.

SHT-Temperaute

The humidity sensor also has a temperature sensor. The temperature is displayed here in both Celsius and Fahrenheit. Select ‘Read’ to read the current temperature or select ‘Continuous Scan’ form the ‘Function’ menu. Refer to the SHT1x/SHT7x datasheet for more information.

Graph

Select ‘Graph Display’ to display the graph dialog (ss shown below).



The graph window is divided into eight sections: Graph, Legend, Operation, Adjustments, Display Channel, Pause/Store, Change Settings, and Speed Adjust.

Legend
Displays the signal names with the corresponding color.

Operation
Used to set the min and max range of the y-axis. This can be done in the following two ways.

Global — Makes the coordinates of signal ana1 the default y-axis coordinates. So no matter which channel is chosen from the list box, “Settings” brings up the form for the signal ana1. Any changes made to the coordinates of ana1 will become the y-axis coordinates.

Independent — The y-axis min coordinate becomes the y-min (setting) of all the signals selected and y-axis max becomes the y-max (setting) of all the signals selected. Individual settings such as color, coordinates, Title, Abbreviation can be changed using this option and selecting the corresponding channel.

Adjustments
This feature is useful to locate the graph if a signal is chosen and is not visible on the screen.

Scale Max - Increases the scale for the max range of y-axis up to 50units
Scale Min - Decreases the scale for the min range of y-axis up to 50units.

Note: To increase or decrease the scales at the same time, so that the graph shrinks or expands use “ Ctrl+L ” (lock) and decrease or increase either scale max or scale min. Unlock it using “ Ctrl+U ” to get the normal display.

Display Channel
Channels - One can choose signals from 8 channels whose graph is to be displayed on the screen.
Offset - If the graphs overlap each other checking this option on will separate them and display them on top of each other.

Pause/Store
Stop/Start - Use to pause and start the display.
Save - Used to save the snapshot of the graphs on the screen to files. These files are stored in .txt files and can be imported by Excel spreadsheets to get the graph for print out.

Change Settings
Settings - To change the individual settings.
Save - Save the settings to the file for next time.

Speed Adjust
The fastest that the graphs are displayed is every 100ms. This can be decreased, by using the adjustments. It depends on the speed of the processor.

PROGRAMMING

Humidity Sensor Communication

The humidity sensor (SHT1x from Sensirion) has two lines: Data and Clock.

The humidity sensor clock is controlled by writing either a 1 (High) or a 0 (Low) to the Clk bit of register 0.

The Data line can be accessed as input or output using the Din and Dout bits of Register 0.To write a bit to a data line, set the DDR bit to 1 and then write the corresponding bit to Dout.To read a bit from the data line, set the DDR bit to 0 and then read the Din bit.

Basic steps to communicate with the humidity sensor:

1. Issue the “Connection Reset Sequence”
2. Issue the “Transmission Start” sequence
3. Write Command
4. Read the data

Command	**Code**
Reserved	0000x
Measure Temperature	00011
Measure Humidity	00101
Read Status Register	00111
Write Status Register	00110

Please see the Sensirion User’s Manual and supplied Visual Basic source code for more details.

Register Description

The offsets shown are from the selected base I/O address.

+0 Humidity Read/Write Register
- Bit Mapped
- 0 - Humidity Data Direction (DDR) (0 – Input, 1 – Output)
- 1 - Humidity Data Output (Dout)
- 2 - Humidity Clock (Clk)
- 3 - Humidity Data Input (Din)
- 4 – Not Used
- 5 - IRQ Select A, IRQ2/9=0, IRQ3=1, IRQ4=2
- 6 - IRQ Select B, IRQ5=3, IRQ6=4, IRQ7=5
- 7 - IRQ Select C, IRQ10=6, IRQ11=7

+1 Digital Output Register
- Bit Mapped, Read/Write
- 0 - Unused
- 1 - LED Output, XORed with Fan Output
- 2 - Force Fan On, 1=On
- 3 - Spare Output, CPLD Pin 34
- 4 - Relay Output #0, Active = 1
- 5 - Relay Output #1, Active = 1
- 6 - Pressure Sensor Output #0, 1=Active

7 - Pressure Sensor Output #1, 1=Active

+2 Data Direction Register

Bit Mapped, Read/Write

0 - Output=0, Input=1 Pin 85, CPU PA0
1 - Output=0, Input=1 Pin 78, CPU PA1
2 - Output=0, Input=1 Pin 89, CPU PA2
3 - Output=0, Input=1 Pin 82, CPU PA7
4 - Output=0, Input=1 Pin 74, CPU IRQ
5 - Output=0, Input=1 CPLD Pin 43
6 - Output=0, Input=1 CPLD Pin 46
7 - Output=0, Input=1 CPLD Pin 73

+3 Digital Input Register

Bit Mapped, Read Only

0 - Temperature #1, Low Limit, Active High
1 - Temperature #1, High Limit, Active High
2 - Temperature #2, Low Limit, Active High
3 - Temperature #2, High Limit, Active High
4 - Digital Input #0, Active Low, 10K pull-up to 5vdc
5 - Digital Input #1, Active Low, 10K pull-up to 5vdc
6 - Busy from A/D, MAX156
7 - Unused

+4 Data Register

Bit Mapped, Read/Write

0 - I/O, CPLD Pin 85, CPU PA0
1 - I/O, CPLD Pin 78, CPU PA1
2 - I/O, CPLD Pin 89, CPU PA2
3 - I/O, CPLD Pin 82, CPU PA7
4 - I/O, CPLD Pin 74, CPU IRQ
5 - I/O, CPLD Pin 43
6 - I/O, CPLD Pin 46
7 - I/O, CPLD Pin 73

+5 Bit Mapped, Read Only

0 - Configuration Jumper #1, Installed = 0
1 - Configuration Jumper #2, Installed = 0
2 - Configuration Jumper #3, Installed = 0
3 - Configuration Jumper #4, Installed = 0
4 - Unused
5 - Unused
6 - Unused
7 - Unused

+6 Temperature Output Register

Bit Mapped, Read/Write

0 - Temp DQ Input/Output, DS1620
1 - Temp Data Direction, Input = 0
2 - Temp TCLK, DS1620
3 - Temp #0, TRST, DS1620
4 - Temp #1, TRST, DS1620, External or by Humidity Sensor
5 - Low Temperature Threshold Disable, Disable=1
6 - High Temperature Threshold Disable, Disable=1
7 - Interrupt Enable, Enable=1, Active when Fan On

+7 Unused Register

Not Implemented

DLL Functions Description

The Parvus Environmental Fan Card II has a number of I/O features that can be accessed through this Windows® 9x/NT/2000/XP driver. The Parvus DLL drivers are based on WinDriver technology from Jungo® (www.jungo.com). A small direct access kernel is loaded and registered with the operating system and then the specific board DLL can then be used.

The WinDriver must first be installed and registered with the operating system. This can be accomplished through the install wizard. On systems that are integrated by Parvus and then delivered, the driver is already installed.

The application software may be written to call the DLL dynamically or load it statically. The dynamic load simply identifies the DLL routines by name or ordinal such as with Visual Basic 6 or it can be linked in statically with a library such as with Visual C++ 6.

A channel must be opened to the board through the DLL by calling an OPEN command. This sets up the DLL for direct access to the hardware through the WinDriver kernel. Calls to the DLL can then occur including closing and reopening the channel to change base address of the board. Please note that the jumper settings on the board must match the selected base address in the DLL in order for proper access to occur.

This calling sequence will open the channel to the board using base address range 0 turn on the LED and then close the channel.

```
vbapp_Open(0)
vbapp_LedOn
vbapp_Close
```

It is possible to determine the address range sets by calling the vbapp_CardRangesL at the beginning of the application by using a function similar to this:

```
Const MAX_IO = 7   ' 8 - 1 for BASIC arrays upperbound
Const MAX_MEM = 7
Type PRVDRV_RANGES
  nRanges As Long
  nIo As Long
  nMem As Long
  ioBase(MAX_IO) As Long
  ioBytes(MAX_IO) As Long
  ioOffset(MAX_IO) As Long
  memBase(MAX_MEM) As Long
  memBytes(MAX_MEM) As Long
  memOffset(MAX_MEM) As Long
End Type
Public range As PRVDRV_RANGES

Call vbapp_CardRangesL( _
  range.nRanges, _
  range.nIo, _
  range.nMem, _
  range.ioBase(0), _
  range.ioBytes(0), _
  range.ioOffset(0), _
  range.memBase(0), _
```

```
        range.memBytes(0), _
        range.memOffset(0))

    For J = 0 To range.nIo - 1
        Base_IO(J).Text = Hex$(range.ioBase(J)) ‘ load array with values
        brdAddrList.AddItem Hex$(range.ioBase(J)) ‘ add to list box
        Extent_IO(J).Text = range.ioBytes(J)
        Next

    For J = 0 To range.nMem - 1
        Base_MEM(J).Text = Hex$(range.memBase(J))
        Extent_Mem(J).Text = range.memBytes(J)
        Next
```

This sets up the arrays to receive the information of the available address ranges for the cards I/O and memory addressing.

DLL Routines

The DLL file ‘prvapp_EnvFan.dll’ has a number of functions that directly access Environmental Fan board. These functions use the low-level access system created by WinDriver. Since the board driver is at a fairly high level it can be used on many operating systems. The only portion that changes from OS to OS is the kernel.

The following routines are available for usage. The address range of the card is fixed in hardware and adjustable by the user with jumper. These address ranges are also available in the DLL for selection. These routines access the bit and byte descriptions of I/O address space occupied by the PC/104 board. Some routines are designed to be single purpose such as LED_On while others are more general purpose like SetDig.

The normal usage of the DLL is to open the communications channel to the board when the application starts and the close it when the application ends. When opening the communications channel, the address range is selected.

To maintain compatibility with Visual Basic all passed parameters and returned results are longs or pointers to parameters. VB applications primarily pass parameters by value.

The source code for these routines are available with a specific signed NDA from Parvus. This source code can be ported to other operating systems that have C compilers. The parameter and variable types may be different in the available source code.

Common Routines:

Name: **vbapp_Open**
Entry: Channel - Address Group Base, 0-320, 1-340, 2-360
Exit: Status of Open
    0 = Already Open
    1 = Fault Upon Open
    2 = Successful Open
Globals: None
Prototype: long WINAPI vbapp_Open(long Channel);
Description: Performs and open of the channel between the board and the application software. Checks for a proper installation of the DLL and WinDriver components. This routine must be called to start the channel. The address group base changes with each card type.

Name: **vbapp_Close**
Entry: None
Exit: None
Globals: None
Prototype: void WINAPI vbapp_Close(void);
Description: Closes and open channel. Performs the close only if the channel is open, performs a no-op if the channel was already closed or not open.

Name: **vbapp_LedOn**
Entry: None
Exit: None
Globals: None
Prototype: void WINAPI vbapp_LedOn(void);
Description: Turns the diagnostic LED located on the PC/104 board on. The diagnostic LED is used as a simple test to see that the board is operational and that the DLL is attached to it. The normal state of the LED is on.

Name: **vbapp_LedOff**
Entry: None
Exit: None
Globals: None
Prototype: void WINAPI vbapp_LedOff(void);
Description: Turns the diagnostic LED located on the PC/104 board off

Name: **vbapp_Read**
Entry: Offset from base address of byte to read in I/O space
Exit: Value read from supplied offset
Globals: None
Prototype: long WINAPI vbapp_Read(long offset);
Description: This is a general-purpose routine for direct access to the board. It works by reading from the base address plus the supplied offset. The returned value is a byte read from the board. This can be used for creating a uniform interface set, diagnostic purposes, or for accessing features potentially missing from the DLL command set.

Name: **vbapp_Write**
Entry: Offset from base address of byte to read in I/O space
Value to write to board
Exit: None
Globals: None
Prototype: void WINAPI vbapp_Write(long offset, long value);
Description: This is a general-purpose routine for direct access to the board. It works by writing to the base address plus the supplied offset. The value is a byte written to the board. This can be used for creating a uniform interface set, diagnostic purposes, or for accessing features potentially missing from the DLL command set.

Name: **vbapp_Test**
Entry: A test value, 0-255
Exit: The test value times 2
Globals: None
Prototype: long WINAPI vbapp_Test(long value);
Description: This routine is used to assure that the DLL is properly attached and that parameters can be passed back and forth. It simply multiplies the supplied value by 2 and returns it.

Name: **vbapp_Version**
Entry: None
Exit: Returns a long value of version, 10102 translates to 1.01.02
Globals: None
Prototype: long WINAPI vbapp_Version(void);
Description: This is used to version lock DLLs to critical applications as well as to simply return the version of the DLL during test.

Name: vbapp_**CardRangesL**
Entry: These are all pointers to values

int* nRanges	-
int* nIo	- Number of available I/O Ranges
int* nMem	- Number of available Memory Ranges
int* ioBase	- First element in I/O Range Array
int* ioBytes	- First element in I/O Range Size Array
int* ioOffset	- Reserved, empty pointer required
int* memBase	- First element in Memory Range Array
int* memBytes	- First element in Memory Range Size Array
int* memOffset	- reserved, empty pointer required

Exit: All pointer values are filled
Globals: None
Prototype: void WINAPI vbapp_CardRangesL (unsigned int* nRanges,
unsigned int* nIo,
unsigned int* nMem,
unsigned int* ioBase,
unsigned int* ioBytes,
unsigned int* ioOffset,
unsigned int* memBase,
unsigned int* memBytes,
unsigned int* memOffset)
Description: By calling this routine at the beginning of the application it is possible to get all of the I/O and memory ranges available for the board.

Application Specific Routines
These functions and routines are unique to the function of the board. They are all prefixed with the same value for easy identification.

Name: **envDiagLED**
Entry: LED State, 1=On, 0=Off, -1=No change
Exit: Resultant LED state
Globals: None
Prototype: long envDiagLED(long state);
Description: Allows the set and clear of the diagnostic LED located on the PC/104 board. A parameter of -1 simply return the state of the LED

Name: **envReadCfg**
Entry: None
Exit: Bit mapped Status of configuration jumpers
0 - Configuration Jumper #1, Installed = 0
1 - Configuration Jumper #2, Installed = 0
2 - Configuration Jumper #3, Installed = 0
3 - Configuration Jumper #4, Installed = 0
4 - Unused
5 - Unused
6 - Unused

7 - Unused
Globals: None
Prototype: long envReadCfg(void);
Description: Returns the state of the configuration jumpers. These jumpers have no effect on the function of the board and are supplied for application specific usage.

Name: **envReadDig**
Entry: None
Exit: Bit mapped status of the digital inputs
- 0 - Temperature #1, Low Limit, Active High
- 1 - Temperature #1, High Limit, Active High
- 2 - Temperature #2, Low Limit, Active High
- 3 - Temperature #1, High Limit, Active High
- 4 - Digital Input #0, Active Low, 10K pull-up to 5vdc
- 5 - Digital Input #1, Active Low, 10K pull-up to 5vdc
- 6 - Busy from A/D, MAX156
- 7 - Unused

Globals: None
Prototype: long envReadDig(void);
Description: Returns the status of the various digital inputs on the board

Name: **envSetDig**
Entry: bit mapped value 0-255, -1 simply returns state
- 0 - Relay Output #0, Active = 1
- 1 - Relay Output #1, Active = 1
- 2 - Pressure Sensor Output #0, 1=Active
- 3 - Pressure Sensor Output #1, 1=Active
- 4 - Unused
- 5 - Unused
- 6 - Unused
- 7 - Unused

Exit: Resultant state of the digital output register
- 0 - Unused
- 1 - LED Output, XORed with Fan Output
- 2 - Force Fan On, 1=On
- 3 - Spare Output, CPLD Pin 34
- 4 - Relay Output #0, Active = 1
- 5 - Relay Output #1, Active = 1
- 6 - Pressure Sensor Output #0, 1=Active
- 7 - Pressure Sensor Output #1, 1=Active

Globals: None
Prototype: long envSetDig(long val);
Description: Sets the various digital values of the board. A passed parameter of -1 will only return the state of the digital outputs and not change them. Note that the return value is the full register and the passed parameter is only the digital outputs.

Name: **envReadAna**
Entry: Channel of analog value to convert and return, 0-7
Exit: 8 bit analog value from selected channel
Globals: None
Prototype: long envReadAna(long channel);
Description: The analog value is converted and read based on the supplied channel.

Name: **envSetEnabled**
Entry:
Exit:
Globals: None
Prototype: long envSetEnabled(long value);
Description:

Name: **envFan**
Entry: State of Fan
Exit: Resultant state of fan
Globals: None
Prototype: long envFan(long value);
Description: Force the fan to turn on. There are no provisions to for the fan to turn off other than to disable both high and low thermostat enables.

Name: **envInitTemp**
Entry: Selected temperature measurement channel.
Exit: None
Globals: None
Prototype: void envInitTemp(long ch);
Description: The thermal component is initialized. This function is generally not used because this initialization occurs when the communications channel to the board has been opened.

Name: **envReadTemp**
Entry: Selected temperature measurement channel.
Long Pointers to Long variables for current temp, high setpoint, low setpoint.
Exit: The variables pointed to by the supplied parameters are loaded with the actual values from the digital temperature component.
Globals: None
Prototype: void envReadTemp(long *TempNow, long *TempHi,
long *TempLo, long *TempCfg, long ch);
Description: Read the actual temperature of the digital temperature component. Also read the high and low thresholds as well as current configuration. Refer to the DS1620 documentation for details on the component. All temperatures are 9 bit signed values in 0.5 degree increments. The range is +125 to –55 degree C in .5 degree C steps.

Name: **envWriteTemp**
Entry: Values for high and low thresholds
Exit: None
Globals: None
Prototype: void envWriteTemp (long TempHi, long TempLo, long ch);
Description: Write high and low thresholds that control the digital thermostat high and low outputs. Refer to the DS1620 documentation for details on the component. All temperatures are 9 bit signed values in .5 degree increments. The range is +125 to -55 degree C in .5 degree C steps.

Name: **Sn_GetSerialNumber**
Entry: The selected byte offset.
0) The 0th byte (MSB).
1) The 1st byte.
2) The 2nd byte.
3) The 3rd byte.
4) The 4th byte.
5) The 5th byte (LSB).
Exit: The byte value pointed by the argument. Returns -1 if unsuccessful.

Globals: None
Prototype: int Sn_GetSerialNumber (int byte_num)
Description: Read the 6 bytes unique board serial number from the EEprom component. Refer to the 24LCS61/62 documentation for details. The range is 0 to 255 for each byte returned. -1 is returned if the read fails, that is if the component is not loaded or a wrong base address is chosen, or if the component does not work.

Name: **PowerCycleCountReset**
Entry: None
Exit: Returns 1 if upon successful reset else 0 is returned.
Globals: None
Prototype: int PowerCycleCountReset(void)
Description: Reset the power cycle (event) counter. Refer to the DS1682 documentation for details. 0 is returned if the RESET fails, that is if the component is not loaded or a wrong base address is chosen, or if the component does not work.

Name: **ElapsedTimerReset**
Entry: None
Exit: Returns 1 if upon successful reset else 0 is returned.
Globals: None
Prototype: int ElapsedTimerReset(void)
Description: Reset the total elapsed time counter. Refer to the DS1682 documentation for details. 0 is returned if the RESET fails, that is if the component is not loaded or a wrong base address is chosen, or if the component does not work.

Name: **Evnt_GetPowerCycleCount**
Entry: None
Exit: The total number of times the board has been turned on and off. Returns -1 if unsuccessful.
Globals: None
Prototype: long Evnt_GetPowerCycleCount(void);
Description: Read the power cycle count of the event component. Refer to the DS1682 documentation for details on the component. The range is 0 to 65535(FFFFh). -1 is returned if the read fails, that is if the component is not loaded or a wrong base address is chosen, or if the component does not work.

Name: **Evnt_GetTimeInSecs**
Entry: None
Exit: Total Time elapsed in seconds. Returns -1 if unsuccessful.
Globals: None
Prototype: long Evnt_GetTimeInSecs(void);
Description: Returns the total time elapsed in seconds from the time elapsed counter. Refer to the DS1682 documentation for details. The range is 0 to 1073741823(3FFFFFFFh) seconds. -1 is returned if the read fails, that is if the component is not loaded or a wrong base address is chosen, or if the component does not work.

Name: **envTemperature**
Entry: Option for Centigrade of Fahrenheit.
0) Centigrade
1) Fahrenheit
Exit: The Temperature value in Centigrade of Fahrenheit. Returns -1 if unsuccessful.
Globals: None
Prototype: double envTemperature (int tmp);
Description: Read the actual temperature of the digital temperature component. Refer to the SHT1x/SHT7x documentation for details. The temperature value returned is in Centigrade or Fahrenheit. -1 is returned if the read fails, that is if the component is not loaded or a wrong base address is chosen, or if the component does not work.

Name: **envHumidity**
Entry: None
Exit: The relative Humidity in Percentage. Returns -1 if unsuccessful.
Globals: None
Prototype: double envHumidity (void);
Description: Read the Relative Humidity of the digital Humidity component. Refer to the SHT1x/SHT7x documentation for details. The Humidity value is in percentage. The range is +0 to 100 percent. -1 is returned if the read fails, that is if the component is not loaded or a wrong base address is chosen, or if the component does not work.

SPECIFICATIONS

Environmental

	Operating	Storage
Temperature	-5°C to +60°C Extended Temperature Qualification Available	

Electrical

Requirements	+5VDC

Mechanical

Dimensions	3.550” x 3.775”

Humidity Sensor

Measurement Range	1-99%RH
Accuracy	±2%RH

Pressure Sensor

Temperature Compensation	from 0°C - 85°C
Rating	0-29 PSI
Sensitivity	0.2 mV/kPa

Temperature Sensor

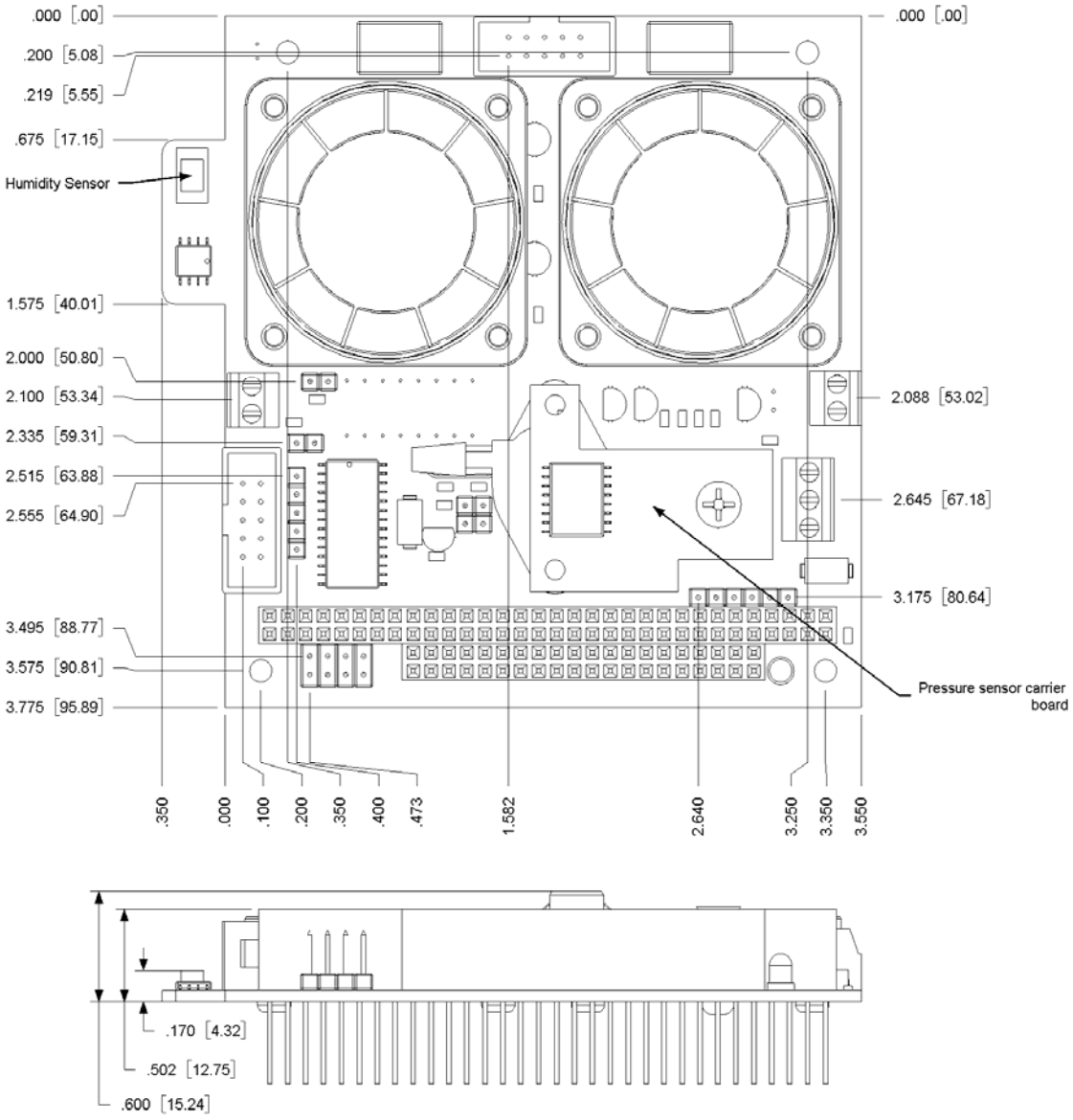
Measurement	0.5° Increments

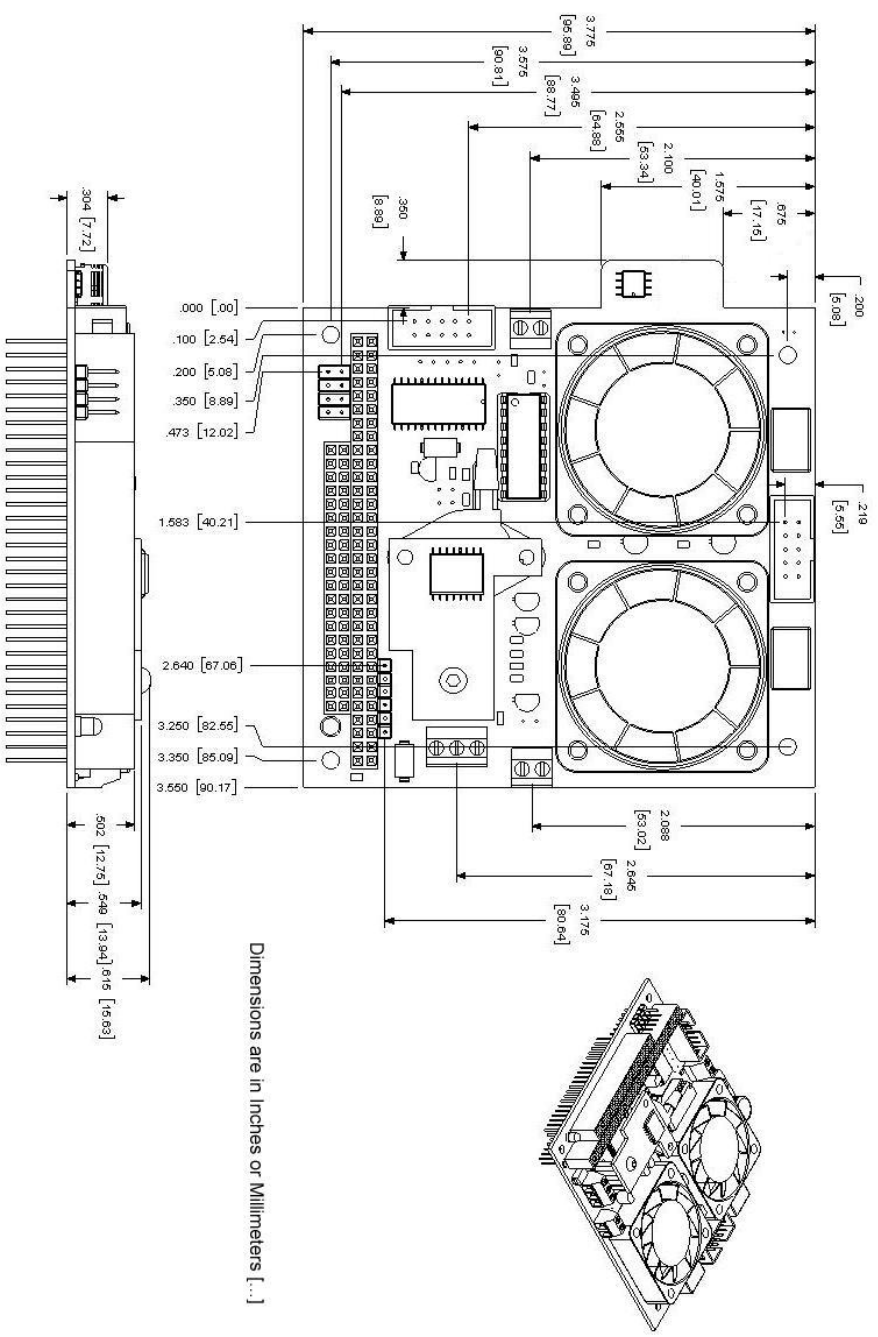
Fan Specs

Speed	6000 RPM
Noise @ 1 meter	25dB
Air Flow Rating	5.6 CFM

APPENDIX

Dimensional Drawings




3.775 [95.89]
3.575 [90.81]
3.485 [88.77]
2.555 [64.88]
2.100 [53.34]
1.575 [40.01]
.675 [17.15]
.350 [8.89]
.200 [5.08]
.219 [5.55]
.304 [7.72]
.000 [.00]
.100 [2.54]
.200 [5.08]
.350 [8.89]
.473 [12.02]
1.583 [40.21]
2.640 [67.06]
3.250 [82.55]
3.350 [85.09]
3.550 [90.17]
.502 [12.75]
.549 [13.94]
.615 [15.63]
2.088 [53.02]
2.646 [67.18]
3.175 [80.64]
Dimensions are in Inches or Millimeters [...]

For more information about this or other products in the Parvus line of embedded systems development tools and control systems call (801) 483-1533 from 8:00AM to 5:00PM Mountain Time, E-mail us at parvus@parvus.com or visit our web-site at: http://www.parvus.com

LIMITED WARRANTY

Parvus Corporation warrants this product to be free of defects in materials and workmanship, and that the product meets or exceeds the current specifications published by Parvus. This Warranty is valid for a period of one (1) year from the date of purchase. Parvus reserves the right to repair or replace any Warranted products at its sole discretion. Any product returned to Parvus for repair or replacement under the provisions of this warranty must be accompanied by a valid Return Material Authorization (RMA) number issued by the Parvus Customer Service Department.

Parvus Corporation makes no warranty not expressly set forth in this document. Parvus disclaims and excludes all implied warranties of merchantability and fitness for a particular purpose. The aggregate liability of Parvus arising from or relating to (regardless of the form of action or claim) is limited to the total of all payments made to purchase the product. Parvus shall not in any case be liable for any special, incidental, consequential, indirect or punitive damages, even if Parvus has been advised of the possibility of such damages. Parvus is not responsible for lost profits or revenue, loss of the use of software, loss of data, costs of recreating lost data, or the cost of any substitute equipment or program.

This Warranty shall be governed by the laws of the United States of America and the State of Utah, and any claim brought under this Warranty may only be brought in state or federal court located in Salt Lake County, State of Utah, and purchaser hereby consents to personal jurisdiction in such courts.

FEDERAL COMMUNICATIONS COMMISSION RADIO FREQUENCY INTERFERENCE STATEMENT

Warning: The equipment described herein has been designed to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC rules. Only peripherals (computer input/output devices, terminals, printers, etc.) designed to comply with the Class B limits may be attached to this computer.

INSTRUCTIONS TO USER: This equipment generates and uses radio frequency energy and if not installed and used properly, i.e., in strict accordance with the operating instructions, reference manuals, and the service manual, may cause interference to radio or TV reception. It has been designed to comply with the limits for a Class B computing device pursuant to Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference when operated in a residential installation.

For further information contact:

Parvus® Corporation
3222 S. Washington St.
Salt Lake City, Utah, USA 84115
(801) 483-1533, FAX (801) 483-1523
Web-site: http://www.parvus.com
Email: parvus@parvus.com