HM104 Manual

Health Monitor
For
PC104 BUS

Manufactured by
TRI-M ENGINEERING
Engineered Solutions for Embedded Applications

Technical Manual

P/N: HM104-MAN
Revision: 25 April 2006

TRI-M ENGINEERING
1407 Kebet Way, Unit 100
Port Coquitlam, BC V3C 6L3
Canada
http://www.Tri-M.com
Tel 604.945.9565
North America 800.665.5600
Fax 604.945.9566

PREFACE

This manual is for integrators of applications of embedded systems. It contains information on hardware requirements and interconnection to other embedded electronics.

DISCLAIMER

Tri-M Engineering makes no representations or warranties with respect to the contents of this manual, and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Tri-M Engineering shall under no circumstances be liable for incidental or consequential damages or related expenses resulting from the use of this product, even if it has been notified of the possibility of such damages. Tri-M Engineering reserves the right to revise this publication from time to time without obligation to notify any person of such revisions. If errors are found, please contact Tri-M Engineering at the address listed on the title page of this document.

CHAPTER 1: INTRODUCTION

1.1 *GENERAL DESCRIPTION*

The HM104 is a compact board design to monitor the overall health of a PC/104 stack. It can simultaneously monitor four internal analog voltage input, five external analog voltage inputs, one on board temperature sensor, 2 external temperature sensors and 2 fan tachometer inputs. It has pulse width modulation outputs to control the speed rotation of two fans. The HM104 also provides visual and audio warning when any measurement goes out of a programmable range.

1.2 Specifications

1.2.1 Analog voltage inputs

- Four internal : +5V, +12V, -12V, -5V.
- Five external : Vcore, Vin, +3.3V, Vbat, 5V stby.

1.2.2 Temperature sensors

- One on board temperature sensor using a 10K NTC thermistor (optional 2N3904 NPN-type transistor).
- Two external sensor inputs using thermistor, 2N3904 NPN-type transistor or CPU thermal diode

1.2.3 Tachometer inputs

- 2 Fan speed monitoring inputs.

1.2.4 Fan control outputs

- 2 PWM outputs for fan speed control.

1.2.5 Alarm

- LED + Beep tone warning.
- SMI# and OVT# signals to active system protection.

1.2.6 Programming and control

- PC/104 bus

1.2.7 Mechanical/Environmental

- PC/104 form factor compliant, 3.55" x 1.5" x 0.9" (90mm x 38mm x 23mm)
- Standard PC/104 16-bit stackthrough connector for PC/104-compliant modules
- Operating temperature -40° to 185°F (0° to 70°C)
- Storage temperature: -67° to 185°F (-55° to 150°C)
- WEIGHT: 0.1 LB. (45G)

CHAPTER 2: EMBEDDED FEATURES

2.1 Customizable External Analog Inputs

- VCORE: default set to +3V (max +4V), can be adjusted with R20.
- VBAT: default set to +3V (max +4V), can be adjusted with R24.
- +3V: default set to +3V (max +4V), can be adjusted with R23.
- 5VSB: default set to +5V (max +6.5V), can be adjusted with R29 and R25.
- VIN: default set to +120V (max +130V), can be adjusted with R3 and R22.

2.2 Customizable internal Temperature sensor

- 10K NTC thermistor or 2N3904 NPN-type transistor.

2.3 Selectable Interrupt

- None, IRQ5 or IRQ7.

2.4 Visual and Audio Alarm

- Programmable Out of range activates LED and Speaker.

2.5 Software Compatibility

- Linux (kernel 2.6, lm_sensor).
- Windows (Motherboard Monitor, Hardware Doctor, Hardware monitor).

Note: Normally every software supporting the Winbond W83782D monitoring IC on ISA/PC104 bus

CHAPTER 3: INSTALLATION

3.1 Locating the Connectors & JUMPERS

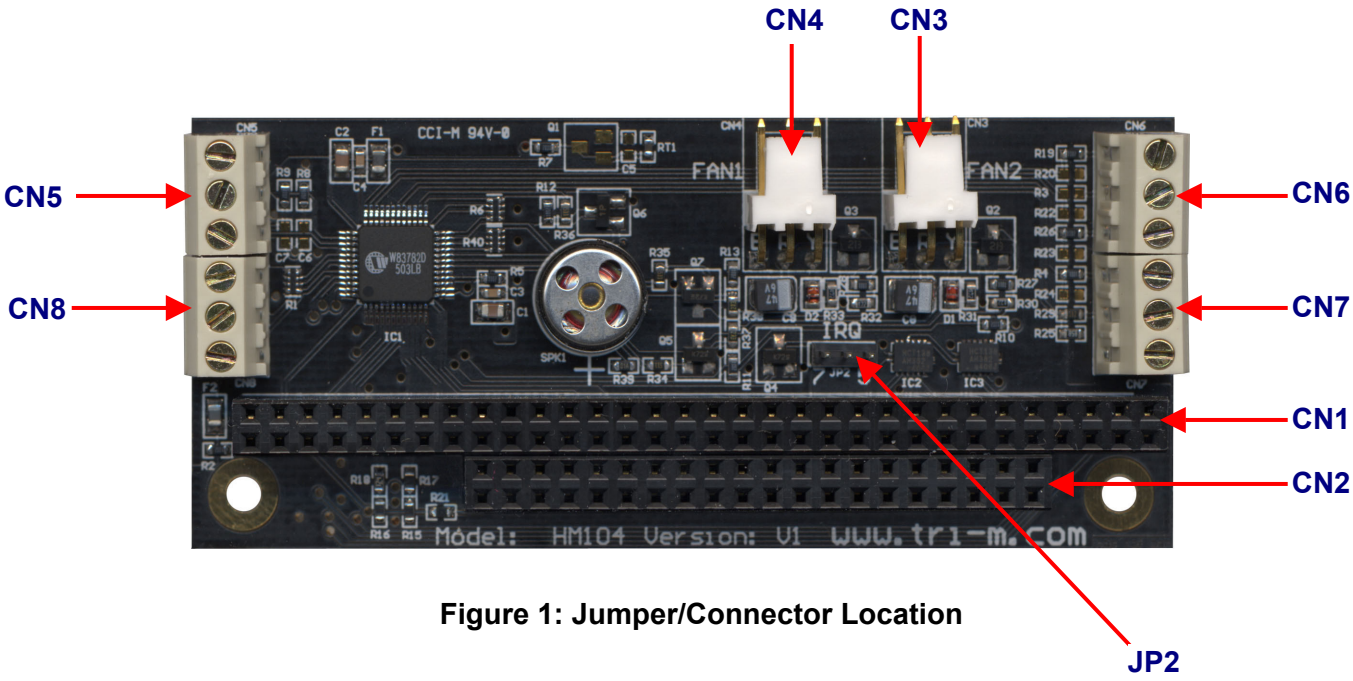


Figure 1: Jumper/Connector Location

CHAPTER 4: JUMPERS

Jumper is provided on the HM104 to select the PC/104 IRQ line.

Jumpers	
Label	**Function**
JP2	IRQ select

Table 1: List of Jumpers

4.1 IRQ output select (JP2)

The SMI# output of the Winbond W83782D monitoring IC is inverted and wired to the PC/104 IRQ5 or IRQ7 through the jumper JP2. When a jumper is populated an interrupt will be generated on the PC/104 bus when an alarm condition occurs. The alarm condition is programmable and can be disabled independently for each input. The IRQ will allow the software to take action on specific events.

IRQ	JP2
None	OFF
5	2-3
7	1-2

Table 2: IRQ Select

CHAPTER 5: CONNECTORS

Connectors on the HM104 are provided to communicate with the board, measure bus/external inputs, generate external events and control the fans.

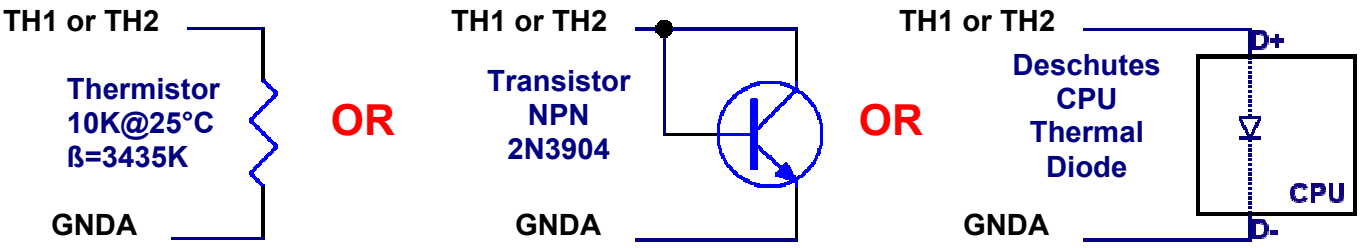
HM104 Connector List	
Connector Label	Function
CN1	PC/104 8 bits BUS
CN2	PC/104 16 bits BUS
CN3	FAN 1
CN4	FAN 2
CN5	External Temperature sensor input TH1 and TH2
CN6	VCORE, VIN and +3V inputs
CN7	VBAT, +5VSB inputs and Analog ground
CN8	External Event output SMI# and OVT#

Table 3: HM104 Connector List

Note: CN5 to CN8 are screw terminal connectors, each input is individually named on the back side of the board.

5.1 External Temperature sensor input connectors (CN5)

The remote temperature sensing can be performed by thermistors, or 2N3904 NPN-type transistors, or directly from Deschutes CPU thermal diode output.



Temperature sensor connector (CN5)			
Pin #	**Marking**	**Type**	**Description**
1	GNDA	Power	Analog Ground
2	TH2	Input	Temperature sensor input 2
3	TH1	Input	Temperature sensor input 1

Table 4: Temperature sensor connector

5.2 Fan connectors (CN3 and CN4)

Use to connect of the shelve 12VDC PC fans, they allow to measure and control the fan speed rotation.

Fan connector (CN3 and CN4)			
Pin #	**Marking**	**Type**	**Description**
1	B	Power	Power Ground
2	R	Output	Programmed Voltage output
3	Y	Input	Tachometer input

Table 5: Fan connector

5.3 Analog Voltage input connectors (CN6 and CN7)

These connectors allow to measure up to five external voltage input. Each input is connected to the Winbond monitoring IC through a resistor divider to lower the input voltage in a 0 to 4V range.

Analog Voltage input connector (CN6 and CN7)				
Pin	**Type**	**Range**	**Step**	**Description**
VCORE	Input	0 to 4V	0.016V	Vcore dedicated input
VIN	Input	0 to 130V	0.51 V	High voltage dedicated input
3V	Input	0 to 4V	0.016V	+3.3V dedicated input
VBAT	Input	0 to 4V	0.016V	Battery dedicated input
5VSB	Input	0 to 6.5V	0.027	5V standby dedicated input
GNDA	Power	/	/	Analog Ground

Table 6: Analog Voltage inputs connectors

5.4 External event connector (CN8)

The HM104 provides two output activated on programmable events.

External event connector (CN8)			
Pin #	**Marking**	**Type**	**Description**
1	GNDD	Power	Digital Ground
2	OVT#	Output	Over temperature flag
3	SMI#	Output	Out of Range alarm flag

Table 7: External Event connector

5.5 PC/104 connector (CN1 and CN2)

These connectors interface the HM104 with a computer to configure the board and retrieve the measurement. They are also used the measure the bus power input +5V, +12V, -12V and –5V.

PC/104 8-bit Connector (CN1)			
Pin #	**Signal**	**Pin #**	**Signal**
A1	/IOCHCK	B1	GND
A2	SD7	B2	RESETDRV
A3	SD6	B3	+5V
A4	SD5	B4	IRQ9
A5	SD4	B5	-5V
A6	SD3	B6	DRQ2
A7	SD2	B7	-12V
A8	SD1	B8	/0WS
A9	SD0	B9	+12V
A10	IOCHRDY	B10	GND(*)
A11	AEN	B11	/SMEMW
A12	SA19	B12	/SMEMR
A13	SA18	B13	/IOW
A14	SA17	B14	/IOR
A15	SA16	B15	/DACK3
A16	SA15	B16	DRQ3
A17	SA14	B17	/DACK1
A18	SA13	B18	DRQ1
A19	SA12	B19	/REFRESH
A20	SA11	B20	SYSCLK
A21	SA10	B21	IRQ7
A22	SA9	B22	N/A
A23	SA8	B23	IRQ5
A24	SA7	B24	IRQ4
A25	SA6	B25	IRQ3
A26	SA5	B26	/DACK2
A27	SA4	B27	TC
A28	SA3	B28	BALE
A29	SA2	B29	+5V
A30	SA1	B30	OSC
A31	SA0	B31	GND
A32	GND	B32	GND

Table 8: PC/104 8-bit Interface

PC/104 16-bit Connector (CN2)			
Pin #	**Signal**	**Pin #**	**Signal**
C0	GND	D0	GND
C1	/SBHE	D1	/MEMCS16
C2	LA23	D2	/IOCS16
C3	LA22	D3	IRQ10
C4	LA21	D4	IRQ11
C5	LA20	D5	IRQ12
C6	LA19	D6	IRQ15
C7	LA18	D7	IRQ14
C8	LA17	D8	/DACK0
C9	/MEMR	D9	DRQ0
C10	/MEMW	D10	/DACK5
C11	SD8	D11	DRQ5
C12	SD9	D12	/DACK6
C13	SD10	D13	DRQ6
C14	SD11	D14	/DACK7
C15	SD12	D15	DRQ7
C16	SD13	D16	+5V
C17	SD14	D17	/MASTER
C18	SD15	D18	GND
C19	GND(*)	D19	GND

Table 9: PC/104 16bit Interface

CHAPTER 6: INTERFACE

The HM104 is interfaced through the PC/104 bus. The board uses two I/O ports to provide access to all the internal registers. The Port 295_h is used to select the register to be accessed and the port 296_h is used to read or write the register data.

6.1 Index Register Port (295_h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BUSY	A6	A5	A4	A3	A2	A1	A0

BUSY : Indicate if there is a bus transaction in progress.
The bit is set when the Index Register Port 295_h is accessed.
The bit is reset when the Data Register Port 296_h is accessed.
This bit allows multiple programs to access the ports without interfering.

A6-A0: Address of the internal register to be read or write.

The power on default value of this register is 00_h.

6.2 Data Register Port (296_h)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

D7-D0: 8 bits data to be read from or to be written to the internal register.

Note: Some registers use auto-increment, accessing the Data Register Port will automatically increment the Index Register Port. See the next chapter for more details.

CHAPTER 7: REGISTERS

7.1 Base registers

Register	Address	Auto-incremental Address	Power on default
VCORE Reading	20h	60h	/
VIN Reading	21h	61h	/
3V Reading	22h	62h	/
+5V Reading	23h	63h	/
+12V Reading	24h	64h	/
-12V Reading	25h	65h	/
-5V Reading	26h	66h	/
On board Temperature sensor Reading	27h	67h	/
FAN1 Reading	28h	68h	/
FAN2 Reading	29h	69h	/
VCORE limit high	2Bh	6Bh	FFh
VCORE limit low	2Ch	6Ch	00h
VIN limit high	2Dh	6Dh	FFh
VIN limit low	2Eh	6Eh	00h
3V limit high	2Fh	6Fh	EDh
3V limit low	30h	70h	B0h
+5V limit high	31h	71h	01h
+5V limit low	32h	72h	0Ah
+12V limit high	33h	73h	00h
+12V limit low	34h	74h	0Ch
-12V limit high	35h	75h	04h
-12V limit low	36h	76h	10h
-5V limit high	37h	77h	04h
-5V limit low	38h	78h	44h
On board Temperature sensor limit low	39h	79h	00h
On board Temperature sensor hysteresis	3Ah	7Ah	00h
FAN1 count limit low	3Bh	7Bh	41h
FAN2 count limit low	3Ch	7Ch	20h
Configuration register	40h	/	01h
Interrupt Status register 1	/	41h	00h
Interrupt Status register 2	42h	/	00h
Interrupt Mask register 1	/	43h	00h
Interrupt Mask register 2	44h	/	00h
VID/FAN divisor register	47h	/	50h
Voltage ID (VID4) and Device ID	49h	/	01h
IRQ#/OVT# property select	4Ch	/	01h
FAN, BEEP and GPO control register	4Dh	/	15h
Register 50h to 5Fh bank select	4Eh	/	80h
Winbond Vendor ID	4Fh	/	5Ch

Table 10: W83782D Base Registers

7.2 BANK 0

Register	Address	Auto-incremental Address	Power on default
BEEP control register 1	56h	/	00h
BEEP control register 2	57h	/	80h
Chip ID	58h	/	30h
Diode selection register	59h	/	70h
PWMOUT 2 control	5Ah	/	FFh
PWMOUT 1 control	5Bh	/	FFh
PWMOUT 1 and 2 clock select	5Ch	/	11h
VBAT monitor control register	5Dh	/	00h

Table 11: W83782D Bank 0 Registers

7.3 BANK 1

Register	Address	Auto-incremental Address	Power on default
Temperature sensor 2 high byte	50h	/	/
Temperature sensor 2 low byte	51h	/	/
Temperature sensor 2 configuration register	52h	/	00h
Temperature sensor 2 hysteresis high byte	53h	/	4Bh
Temperature sensor 2 hysteresis low byte	54h	/	00h
Temperature sensor 2 over-temperature high byte	55h	/	50h
Temperature sensor 2 over-temperature low byte	56h	/	00h

Table 12: W83782D Bank 1 Registers

7.4 BANK 2

Register	Address	Auto-incremental Address	Power on default
Temperature sensor 3 high byte	50h	/	/
Temperature sensor 3 low byte	51h	/	/
Temperature sensor 3 configuration register	52h	/	00h
Temperature sensor 3 hysteresis high byte	53h	/	4Bh
Temperature sensor 3 hysteresis low byte	54h	/	00h
Temperature sensor 3 over-temperature high byte	55h	/	50h
Temperature sensor 3 over-temperature low byte	56h	/	00h

Table 13: W83782D Bank 2 Registers

7.5 BANK 4

Register	Address	Auto-incremental Address	Power on default
Interrupt status register 3	50h	/	00h
Interrupt mask register 3	51h	/	00h
BEEP control register 3	53h	/	00h
Real Time Hardware status register 1	59h	/	00h
Real Time Hardware status register 2	5Ah	/	00h
Real Time Hardware status register 3	5Bh	/	00h
PWMOUT 3 & 4 clock select	5Ch	/	11h

Table 14: W83782D Bank 4 Registers

7.6 BANK 5

Register	Address	Auto-incremental Address	Power on default
5VSB Reading	50h	/	/
VBAT Reading	51h	/	/
5VSB limit high	54h	/	10h
5VSB limit low	55h	/	00h
VBAT limit high	56h	/	00h
VBAT limit low	57h	/	00h

Table 15: W83782D Bank 5 Registers

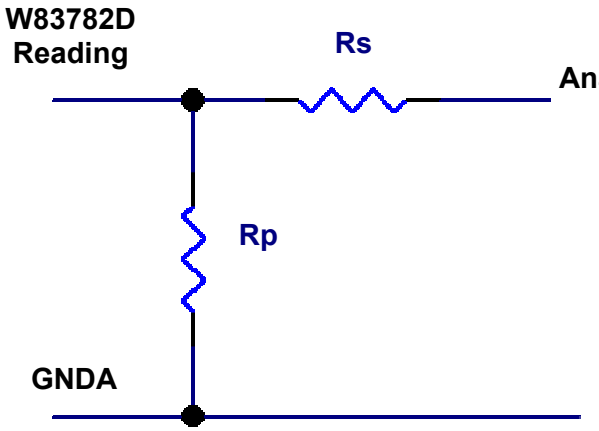
Note: BANK 3 is not used by the Winbond W83783D monitoring IC.

More detailed information about these registers is available in the W83782D Winbond H/W Monitoring IC datasheet.

CHAPTER 8: MEASUREMENT & CONTROL

8.1 Positive Analog Voltage Inputs

The positive analog voltage inputs reading reports the voltage measured on the input resistor divider and not the real input voltage. The resistor values of the input divider will have to be taken in account to calculate the real input voltage.



$$INPUT = \frac{READING \times 4.08 \times (Rs + Rp)}{255 \times Rp}$$

Analog Voltage inputs measurement			
Input	**Rp Value**	**Rs Value**	**Input calculation (V)**
+5V	50K	34K	READING x 4.08 x 84 / 255 / 50
+12V	10K	28K	READING x 4.08 x 38 / 255 / 10
VCORE	NL	10K	READING x 4.08 / 255
VIN	7K5	232K	READING x 4.08 x 239.5 / 255 / 7.5
3V	NL	10K	READING x 4.08 / 255
VBAT	NL	1K	READING x 4.08 / 255
5VSB	7K5	5K1	READING x 4.08 x 12.6 / 255 / 7.5

Table 16: Positive Analog Voltage input measurement

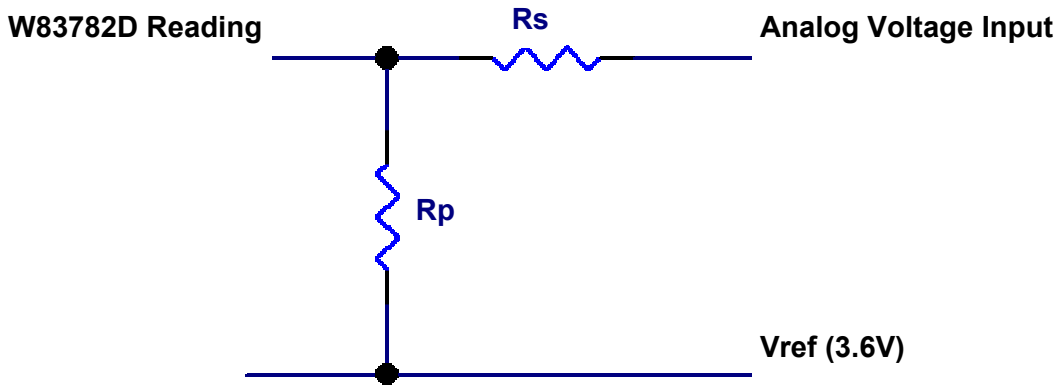
Note: NL indicate a resistor not populated on the board in the default configuration

Analog Voltage Reading Examples					
Input	**Reading**	**Rp Value**	**Rs Value**	**Input calculation (V)**	**Real input value**
3V	C7h	NL	10K	**199** x 4.08 / 255	3.18V
+5V	B8h	50K	34K	**184** x 4.08 x 84 / 255 / 50	4.95V
+12V	C6h	10K	28K	**198** x 4.08 x 38 / 255 / 10	12.04V

Table 17: Positive Analog Voltage Reading Example

8.2 Negative Analog Voltage Inputs

The negative analog voltage inputs reading reports the voltage measured on the input resistor divider referenced to 3.6V and not the real input voltage. The resistor values of the input divider will have to be taken in account to calculate the real input voltage.



$$INPUT = Vref - \left(Vref - \frac{READING \times 4.08}{255} \right) \left(\frac{Rs + Rp}{Rp} \right)$$

Analog Voltage inputs measurement			
Input	**Rp Value**	**Rs Value**	**Input calculation (V)**
-5V	56K	120K	3.6 - (3.6 - READING x 0.016) x (176 / 56)
-12V	56K	232K	3.6 - (3.6 - READING x 0.016) x (288 / 56)

Table 18: Negative Analog Voltage input measurement

Analog Voltage Reading Examples					
Input	**Reading**	**Rp Value**	**Rs Value**	**Input calculation (V)**	**Real input value**
-5V	2Ch	56K	120K	3.6 – (3.6 – **44** x 0.016) x (176 / 56)	-5.5V
-12V	23h	56K	232K	3.6 – (3.6 – **35** x 0.016) x (288 / 56)	-12.03V

Table 19: Negative Analog Voltage Reading Example

8.3 Temperature sensor inputs

The temperature date format is 8 bits two-complement for the internal sensor and 9 bits two-complement for both external sensors. The temperature reading register of the internal sensor reports the temperature in degree Celsius. The temperature reading registers of the external sensors report the temperature in ½ degree Celsius.

Temperature input readings		
Temperature	**Internal sensor**	**External sensors**
+125 °C	$0111.1101_B : 7D_H : 125_D$	$0.1111.1010_B : 0FA_H : 250_D$
+25 °C	$0001.1001_B : 19_H : 25_D$	$0.0011.0010_B : 032_H : 50_D$
+1 °C	$0000.0001_B : 01_H : 1_D$	$0.0000.0010_B : 002_H : 2_D$
+0.5 °C	/	$0.0000.0001_B : 001_H : 1_D$
0 °C	$0000.0000_B : 00_H : 0_D$	$0.0000.0000_B : 000_H : 0_D$
-0.5 °C	/	$1.1111.1111_B : 1FF_H : -1_D$
-1 °C	$1111.1111_B : FF_H : –1_D$	$1.1111.1110_B : 1FE_H : -2_D$
-25 °C	$1110.1110_B : E7_H : -25_D$	$1.1100.1110_B : 1CE_H : -50_D$
-55 °C	$1100.1001_B : C9_H : -55_D$	$1.1001.0010_B : 192_H : -110_D$

Table 20: Temperature input readings

8.4 FAN tachometer inputs

The HM104 allow fan speed measurement when the connected fan provides a tachometer output.
The value recorded in the fan reading register will be invert proportional to the fan speed and the programmed divisor. The value FFh represents a reading out of range. Increasing the divider will allow to measure lower speed, and decreasing it will allow to measure higher speed. The Divider can be programmed from 1 to 128.

$$READING = \frac{1{,}350{,}000}{RPM \times DIVIDER}$$

Tachometer input reading examples		
RPM	**Divider**	**Reading**
4440	2	98h
4440	16	13h

Table 21: Tachometer input reading examples

8.5 FAN speed control outputs

The HM104 can control the speed of two fan using two PWM outputs. The PWM output duty-cycle can be programmed by a 8 bits register. Modifying the duty-cycle allows the HM104 to drive the fans with a specific voltage. Both fans can be programmed from off condition to full speed.

$$Duty - Cycle(\%) = \frac{PROGRAMMED\_VALUE}{255} \times 100$$

PWM output examples		
Programmed value	**Output voltage**	**RPM**
00h	10mV	0
10h	4.5V	1300
20h	7.7V	2700
30h	9.3V	3400
40h	10V	3750
50h	10.6V	4000
60h	11V	4100
80h	11.3V	4220
A0h	11.6V	4330
FFh	11.9V	4380

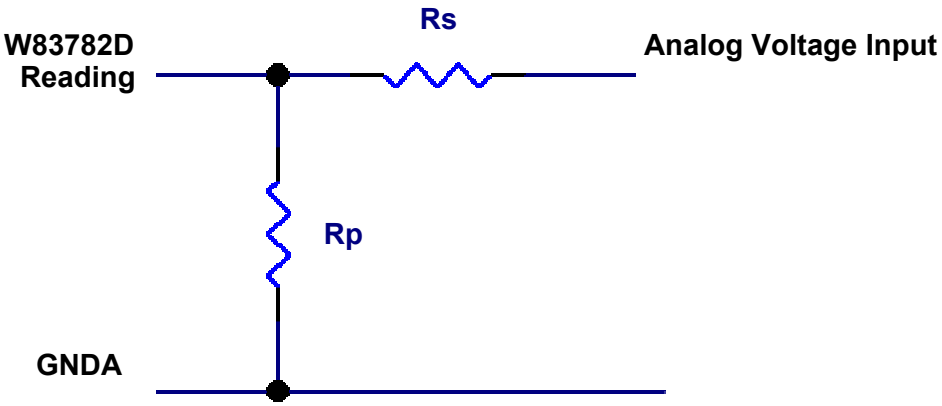
Table 22: PWM output examples

The measurements were performed using a 12V, 0.28A fan, part number F08G-12B2S1.

CHAPTER 9: CUSTOMISATION

9.1 Analog Voltage inputs

All analog voltage inputs of the Winbond monitor IC allow to measure a voltage from 0 to 4.096V with a precision of 0.016V. The HM104 provides a resistor divider to every input to enlarge the range of measurement.The divider can be modified to match a specific measurement need in range and precision.



Analog Voltage Maximum inputs							
Input	**Rp**	**Rs**	**Rp Value**	**Rs Value**	**Vmax calculation (V)**	**Vmax (V)**	**Precision (mV) (Vmax / 255)**
VCORE	R20	R19	NL	10K	4.08/ R20 * (R20 + R19)	4.08	16
VIN	R22	R3	7K5	232K	4.08 / R22 * (R22 + R3)	130.8	511
3V	R23	R26	NL	10K	4.08 / R23 * (R26 + R23)	4.08	16
VBAT	R24	R4	NL	1K	4.08 / R24 * (R24 + R4)	4.08	16
5VSB	R25	R29	7K5	5K1	4.08 / R25 * (R25 + R29)	6.88	27

Table 23: Analog Voltage Maximum inputs

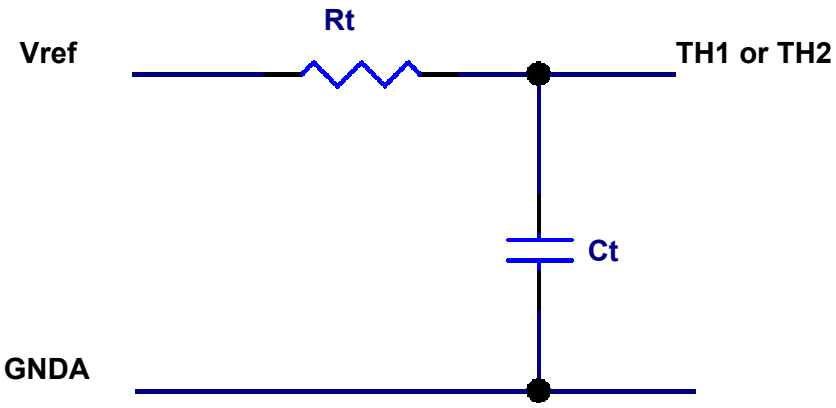
Note: NL indicate a resistor not populated on the board in the default configuration

Analog Voltage Example inputs				
Rp Value	**Rs Value**	**Vmax calculation (V)**	**Vmax (V)**	**Precision (mV) (Vmax / 256)**
10K	15K	4.08/ 10 * (10 + 15)	10	40
12K	47K	4.08 / 12 * (12 + 47)	20	79
12K	105K	4.08 / 12 * (12 + 105)	40	156

Table 24: Analog Voltage Example inputs

9.2 Temperature inputs

The HM104 provides two external connections for remote temperature sensors. These connections allows use of Thermistors (10K@25°C - ß=3435K) or 2N3904 NPN-type transistors or directly from Deschutes CPU thermal diode output. Each input has a circuit composed of a resistor and a capacitor that need to match the temperature sensor used. The HM104 is populated by default with the necessary components to use thermistor on both inputs.



Temperature inputs default					
Input	**Rt**	**Ct**	**Rt Value**	**Ct Value**	**Sensor**
TH1	R8	C6	10K	NL	Thermistors
TH2	R9	C7	10K	NL	Thermistors

Table 25: Temperature inputs default

Note: NL indicate a resistor not populated on the board in the default configuration

Temperature input components match		
Sensor	**Rt Value**	**Ct Value**
Thermistors (10K@25°C - ß=3435K)	10K	NL
2N3904 NPN-type transistors	30K	3.3nF
Deschutes CPU thermal diode output	30K	3.3nF

Table 26: Temperature inputs components match

CHAPTER 10: EXAMPLE PROGRAMS

10.1 Program 1 : read specified register

```
/*
br.c is a small program to access the registers of the HM104.
the program display the content of the specified HM104 register.

usage : br REGISTER_NUMBER

br.c can be compiled with the command : gcc br.c -o br

br.c was tested on a TMZ104 running Linux Slackware 7.0
*/
#include <stdio.h>
#include <stdlib.h>
#include <sys/io.h>

#define IOPORT        0x290 //address of the WINBOND W83782D

int main(int argc, char *argv[])
{
 unsigned char bInByte,bOutByte;
 unsigned int unAddress=IOPORT,unRegister;

 // Parameter test
 if (argc != 2)
 {
  printf ("\nCommand 'br' needs the register number as parameter\n");
  printf ("br REG\n\n");
  return -1;
 }

 sscanf(argv[1],"%i",&unRegister);
 bOutByte = (unsigned char)(unRegister & 0xFF);

 // Request access to selected port
 if (ioperm(unAddress + 0x05,2,1) != 0)
 {
  printf("***** Access to address %04X denied *****\n\n",unAddress);
  return -2;
 }

 outb(bOutByte, unAddress + 0x05); // select the register to read
 bInByte = inb(unAddress + 0x06); // read the data present in the register

 printf("Reading %02X from %02X\n\n",bInByte,bOutByte);
 return 0;
}
```

10.2 Program 2 : write specified register

```
/*
bw.c is a small program to modify the registers of the HM104.
the program display the read back content of the specified HM104
register.

usage : br REGISTER_NUMBER

bw.c can be compiled with the command : gcc bw.c -o bw

bw.c was tested on a TMZ104 running Linux Slackware 7.0
*/
#include <stdio.h>
#include <stdlib.h>
#include <sys/io.h>

#define IOPORT        0x290 //address of the WINBOND W83782D

int main(int argc, char *argv[])
{
 unsigned char bInByte,bOutByte;
 unsigned int unAddress=IOPORT,unRegister,unValue;

 // Parameter test
 if (argc != 3)
 {
  printf ("\nCommand 'bw' needs the register number and the value to store as parameters\n");
  printf ("bw REG VAL\n\n");
  return -1;
 }

 sscanf(argv[1],"%i",&unRegister);
 bOutByte = (unsigned char)(unRegister & 0xFF);

sscanf(argv[2],"%i",&unValue);
 bInByte = (unsigned char)(unValue & 0xFF);

 // Request access to selected port
 if (ioperm(unAddress + 0x05,2,1) != 0)
 {
  printf("***** Access to address %04X denied *****\n\n",unAddress);
  return -2;
 }

 printf("writing %02X to %02X\n",bInByte,bOutByte);

 outb(bOutByte, unAddress + 0x05); // select the register to write
 outb(bInByte, unAddress + 0x06); // write the new value to the selected register
```

```
 outb(bOutByte, unAddress + 0x05); // select the register to read the value back
 bInByte = inb(unAddress + 0x06); // read the data present in the register

 printf("written %02X to %02X\n\n",bInByte,bOutByte);

 return 0;
}
```

10.3 Program 3 : monitoring example

```
/*
mon.c is a small program to access the memory of the HM104.
the program display the content of the HM104 memory.

usage : mon

mon.c can be compiled with the command : gcc mon.c -o mon

mon.c was tested on a TMZ104 running Linux Slackware 7.0

*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/io.h>

#define IOPORT        0x290 //address of the WINBOND W83782D


int main(int argc, char *argv[])
{
 unsigned char bInByte,bOutByte,bIndex;
 unsigned int unAddress=IOPORT,unRegister=0x60;
 unsigned char bMem[30];
 unsigned int unSMI1,unSMI2,unSMImask1,unSMImask2,unFan1Div,unFan2Div;
 float fValue,fLimitH,fLimitL;

 printf("\n");


 // Parameter test
 if (argc != 1)
 {
  printf ("Command 'mon' doesn´t need any parameter\n\n");
  return -1;
 }


 // Request access to selected port
 if (ioperm(unAddress + 0x05,2,1) != 0)
 {
  printf("***** Access to address %04X denied *****\n\n",unAddress);
  return -2;
 }
```

```
bOutByte = 0x60;
outb(bOutByte, unAddress + 0x05); // select the memory start index

for  (bIndex = 0;bIndex < 30;bIndex++)
{
        bMem[bIndex]  = inb(unAddress + 0x06); // read the data present in the memory
}
```

```
/* VCORE display */
fValue = (float)(bMem[0]) * 0.016;
fLimitH = (float)(bMem[11]) * 0.016;
fLimitL = (float)(bMem[12]) * 0.016;
printf("Vcore = %5.3f\t - Limit high = %5.3f\t - Limit low = %5.3f\n",fValue,fLimitH,fLimitL);

/* VIN display */
fValue = (float)(bMem[1]) * 0.016 * (232 + 7.5)  / 7.5;
fLimitH = (float)(bMem[13]) * 0.016 * (232 + 7.5)  / 7.5;
fLimitL = (float)(bMem[14]) * 0.016 * (232 + 7.5)  / 7.5;
printf("Vin   = %5.3f\t - Limit high = %5.3f\t - Limit low = %5.3f\n",fValue,fLimitH,fLimitL);

/* 3.3V display */
fValue = (float)(bMem[2]) * 0.016;
fLimitH = (float)(bMem[15]) * 0.016;
fLimitL = (float)(bMem[16]) * 0.016;
printf("3.3V  = %5.3f\t - Limit high = %5.3f\t - Limit low = %5.3f\n",fValue,fLimitH,fLimitL);

/* +5V display */
fValue = (float)(bMem[3]) * 0.016 * (34 + 50)  / 50;
fLimitH = (float)(bMem[17]) * 0.016 * (34 + 50)  / 50;
fLimitL = (float)(bMem[18]) * 0.016 * (34 + 50)  / 50;
printf("+5V   = %5.3f\t - Limit high = %5.3f\t - Limit low = %5.3f\n",fValue,fLimitH,fLimitL);

/* +12V display */
fValue = (float)(bMem[4]) * 0.016 * (28 + 10)  / 10;
fLimitH = (float)(bMem[19]) * 0.016 * (28 + 10)  / 10;
fLimitL = (float)(bMem[20]) * 0.016 * (28 + 10)  / 10;
printf("+12V  = %5.3f\t - Limit high = %5.3f\t - Limit low = %5.3f\n",fValue,fLimitH,fLimitL);

/* -12V display */
fValue = - ((3.6 - (float)(bMem[5]) * 0.016) / 56 *  (56 + 232)  - 3.6);
fLimitH = - ((3.6 - (float)(bMem[21]) * 0.016) / 56 *  (56 + 232)  - 3.6);
fLimitL = - ((3.6 - (float)(bMem[22]) * 0.016) / 56 *  (56 + 232)  - 3.6);
printf("-12V  = %5.3f\t - Limit high = %5.3f\t - Limit low = %5.3f\n",fValue,fLimitH,fLimitL);
```

```
/* -5V display */
fValue = - ((3.6 - (float)(bMem[6]) * 0.016) / 56 *  (56 + 120)  - 3.6);
fLimitH = - ((3.6 - (float)(bMem[23]) * 0.016) / 56 *  (56 + 120)  - 3.6);
fLimitL = - ((3.6 - (float)(bMem[24]) * 0.016) / 56 *  (56 + 120)  - 3.6);
printf("-5V   = %5.3f\t - Limit high = %5.3f\t - Limit low = %5.3f\n",fValue,fLimitH,fLimitL);

/* Temp1 display */
fValue = (float)(bMem[7]);
fLimitH = (float)(bMem[25]);
fLimitL = (float)(bMem[26]);
printf("Temp  = %5.1f\t - Limit high = %5.1f\t - Hysteresis= %5.1f\n",fValue,fLimitH,fLimitL);

outb(0x47, unAddress + 0x05); // select the FAN Divider low register
bInByte  = inb(unAddress + 0x06); // read the data present in the register

unFan1Div = (bInByte & 0x30) >> 4;
unFan2Div = (bInByte & 0xC0) >> 6;

outb(0x5D, unAddress + 0x05); // select the FAN Divider high register
bInByte  = inb(unAddress + 0x06); // read the data present in the register

unFan1Div = unFan1Div + ((bInByte & 0x20) >> 3);
unFan2Div = unFan2Div + ((bInByte & 0x40) >> 4);
unFan1Div = 1 << unFan1Div;
unFan2Div = 1 << unFan2Div;

/* FAN1 display */
fValue = 1350000 / (float)(bMem[8]) / (float)(unFan1Div);
fLimitL = 1350000 / (float)(bMem[27]) / (float)(unFan1Div);
if (bMem[8] == 0xFF)
 printf("FAN1  = ???????\t ",fValue);
else
 printf("FAN1  = %5.0f\t ",fValue);
printf("- Limit Low  = %5.0f\t ",fLimitL);
printf("- Divider = %d \n",unFan1Div);

/* FAN2 display */
fValue = 1350000 / (float)(bMem[9]) / (float)(unFan2Div);
fLimitL = 1350000 / (float)(bMem[28]) / (float)(unFan2Div);
if (bMem[9] == 0xFF)
 printf("FAN2  = ???????\t ",fValue);
else
 printf("FAN2  = %5.0f\t ",fValue);
printf("- Limit Low  = %5.0f\t ",fLimitL);
printf("- Divider = %d \n",unFan2Div);
```

```
outb(0x41, unAddress + 0x05); // select the SMI# status register 1 index
unSMI1  = inb(unAddress + 0x06); // read the data present in the register

outb(0x42, unAddress + 0x05); // select the SMI# status register 2 index
unSMI2  = inb(unAddress + 0x06); // read the data present in the register

outb(0x43, unAddress + 0x05); // select the SMI# mask register 1 index
unSMImask1  = inb(unAddress + 0x06); // read the data present in the register

outb(0x44, unAddress + 0x05); // select the SMI# mask register 2 index
unSMImask2 = inb(unAddress + 0x06); // read the data present in the register
```

```
 printf("\nInterrupts  = %02X%02X - mask  = %02X%02X \n\n",unSMI2,unSMI1,unSMImask2,unSMImask1);

 return 0;
}
```

CHAPTER 11: LITERATURE REFERENCES

The following references are for information about the PC/104 architecture, the PC DOS, and the PC BIOS.

11.1 ISA System Architecture

MindShare, Inc., Tom Shanley and Don Anderson
Internet: mindshar@interserv.com
CompuServe: 72507,1054
Published by Addison Wesley, Inc.

11.2 AT Bus Design

Edward Solari
Anabooks
12145 Alta Carmel Ct., Suite 250
San Diego, CA 92128
ISBN 0-929392-08-6

11.3 Personal Computer Bus Standard P996

Institute of Electrical and Electronic Engineers, Inc.
445 Hoes Lane
Piscataway, NJ 08854

11.4 PC Interrupts

PC Interrupts, Ralf Brown, Addison/Wesley.

11.5 PC/104 Consortium

809 B-175 Cuesta Drive,
Mountain View, CA 94040
Phone: 415 903-8304
FAX: 415 967-0995

11.6 W83782D Data Sheet

Winbond
4, Creation Rd,
Hsinchu, 300, Taiwan R.O.C.
Phone: 886-3-577-0066