Pike **A**erospace **R**esearch **C**orporation
42 Silver Aspen Crescent
Sudbury, Ontario
Canada, N2N-1J1

Phone: (705) 586-2255

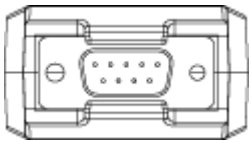
email info@pikeaero.com
web: http://www.pikeaero.com

Model PA1102

Serial Temperature / Humidity Sensor

User Manual

Firmware Revision: **3.1**



PA1102 Specifications

Temperature Range: -40 º C to +85 º C (-40 º F to +185 º F)
Dimensions: 9.1 cm (3.6") x 4.2 cm (1.7") x 2 cm (0.8")
Housing Material: injection molded ABS plastic
Housing Colour: black (bone white available)
Power source: Data Cable (DTR/RTS) or (5VDC to 24VDC)
Current Consumption: ~10 mA
Communications Interface: RS232.
Maximum RS232 extension cable length: 152.5 m (950 ft.).
Communications Protocol: 8-bit ASCII with 16-bit CRC or Checksum error detection.

Communication

The PA1102 sensor communicates at **2400 baud, 8 data bits, no parity, one stop bit (8N1)** in the default settings state.

For data-lline powered operation, both DTR (Data Terminal Ready) and RTS (Request To Send) shall be asserted by the host at all times while communications are taking place. The PA1102 requires approximately 500 microseconds (0.5 milliseconds) from the time that DTR+RTS are asserted, until the time it is ready for receiving data.

The communication between the host and the PA1102 is a simple master/slave style of protocol. The host (computer or device server) register queries, and the PA1102 responds. The fields in each query and response packet are separated by a ':' (colon) character, and the packet is terminated with a carriage-return (Hex 0D) character, or a carriage-return/line-feed pair (Hex 0D/0A).

REGISTER	**READ/ WRITE**	**VARIABLE**	**DESCRIPTION**
R0	R	VARS	Number of Variables.
R1	R	MODEL	Product Model NUmber.
R2	R	SERIAL	Unit Serial Number.
R3	W	VENDOR	Vendor Identification Data.
R4	R	REV	Firmware Revision.
R5	R	TEMPC	Celcius Reading.
R6	R	TEMPF	Fahrenheit Reading.
R7	R	HR	Relative Humidity
R8	R	DEWPOINTC	Calculated DP in Celcius
R9	R	DEWPOINTF	Calculated DP in Fahrenheit
R10	W	RHCAL	Relative Humidity Calibration
R11	W	TCAL	Temperature Calibration
R12	W	OPTION	Option Settings

Command Packet Format

rr<CR>

rr	A register name.
<CR>	The carriage-return character. (0D Hex). <CR><LF> is also acceptable.

Response Packet:

rr:t:a:xxxx:s:yyyy:zzzz<CR><LF>

rr	The register name.
t	The data type (I=integer,R=real,S=string,B=boolean).
a	Access Mode (R=read,W=read/write).
xxxx	Contains the value for the register.
s	Unit of measure.
yyyy	Register name.

zzzz	16-Bit hexadecimal 2's compliment checksum.

RESPONSE PACKET EXAMPLES:

R0:I:R:13:*:VARS:FBBC

R1:S:R:PA1102:*:MODEL:FA8B

R2:S:W:12345678:*:SN:FB06

R3:S:W:www.pikeaero.com:*:VENDOR:F52C

R4:S:R:3.0:*:REV:FBD0

R5:R:R:22.8:C:TEMPC:FAF2

R6:R:R:73.0:F:TEMPF:FAED

R7:R:R:43.2:%:RH:FBF0

R8:R:R:9.6:C:DEWPOINTC:F9E8

R9:R:R:49.0:F:DEWPOINTF:F9B3

R10:I:W:-25:*:RHCAL:FB28

R11:I:W:4050:*:TCAL:FB38

R12:I:W:0x10:*:OPTION:FA42

CRC CALCULATION

The 16-bit CRC calculation is the hexadecimal representation of the the 2's compliment of the sum of all of the data up to and including the last field separator (:).

```
/**
 * Example PA110x sensor response packet validation using CRC.
 */
#include <stdio.h>
#include <string.h>

#define SEPCHAR         ':'     /** The field separation character */
#define MAX_SEPCHARS    6       /** Number of significant SEPCHARs */
#define P_16            0xA001  /** 16 bit polynomial for CRC gen */

unsigned short crc_tab16( unsigned char ch )
{
    unsigned short crc, c;
    unsigned char i=0;
    unsigned char j;
    do {
        crc = 0;
        c   = (unsigned short) i;
        for (j=0; j<8; j++)
        {
            if ( (crc ^ c) & 0x0001 )
                crc = ( crc >> 1 ) ^ P_16;
            else
                crc >>= 1;
            c >>= 1;
        }
    } while ( i++ != ch );
    return crc;
}

void update_crc16( unsigned short* crc, char c )
{
    unsigned short tmp, short_c;
    short_c = 0x00ff & (unsigned short) c;
    tmp = (*crc) ^ short_c;
    *crc = ((*crc) >> 8) ^ crc_tab16( tmp & 0xff );
}

unsigned short xtoi(const char* str)
{
    unsigned short rc=0;
    while ( *str && isxdigit(*str ) )
    {
        char ch = toupper(*str++);
        char nibble=0;
        if ( ch >= 'A' && ch <= 'F' )
            nibble = (ch-'A')+10;
        else
            nibble = (ch-'0');
        rc <<= 4;
        rc |= nibble;
    }
    return rc;
}

int validate(const char* pa110x_response)
{
```

```
    unsigned short crc=0;
    int nsepchars=0;
    int n;
    for(n=0; nsepchars < MAX_SEPCHARS && n < strlen(pa110x_response); n++)
    {
        update_crc16(&crc,pa110x_response[n]);
        if ( pa110x_response[n] == SEPCHAR )
            ++nsepchars;
    }
    return crc == xtoi(&pa110x_response[n]);
}

int main(int argc,char* argv[])
{
    if ( argc == 2 )
    {
        printf( "%s\n", validate(argv[1]) ? "valid" : "invalid" );
    }
}
```

CHECKSUM CALCULATION

The checksum is the hexadecimal representation of the the 2's compliment of the sum of all of the data up to and including the last field separator (:).

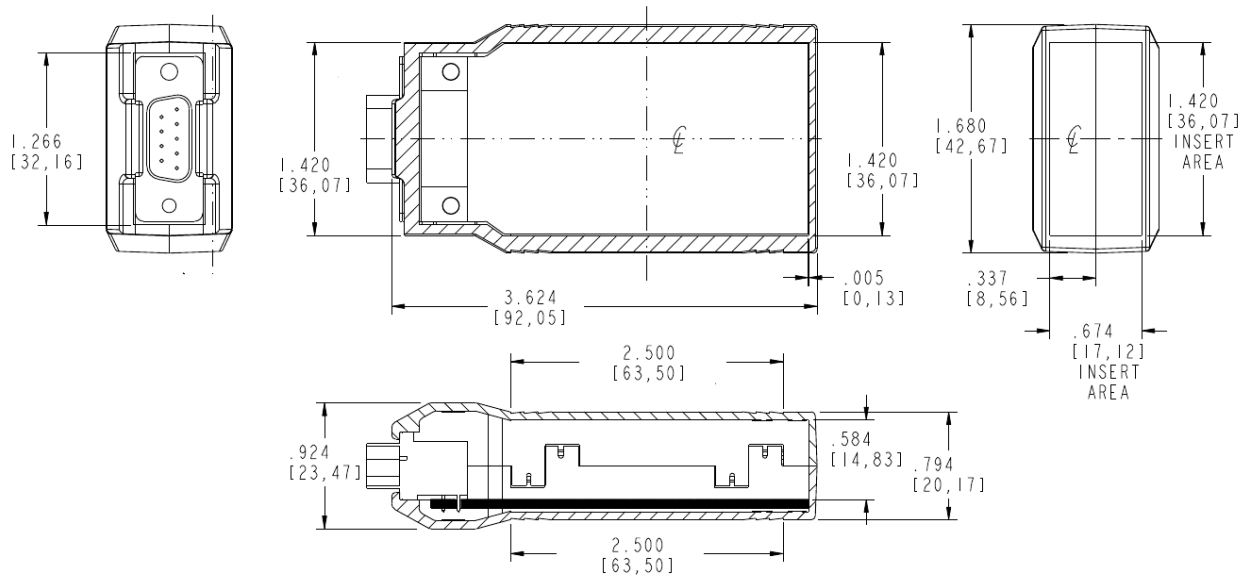
```
/**
 * Example PA110x sensor response packet validation based on checksum.
 */
#include <stdio.h>
#include <string.h>

#define SEPCHAR         ':'
#define MAX_SEPCHARS    6

unsigned short xtoi(const char* str)
{
    unsigned short rc=0;
    while ( *str && isxdigit(*str ) )
    {
        char ch = toupper(*str++);
        char nibble=0;
        if ( ch >= 'A' && ch <= 'F' )
            nibble = (ch-'A')+10;
        else
            nibble = (ch-'0');
        rc <<= 4;
        rc |= nibble;
    }
    return rc;
}

int validate(const char* pa110x_response)
{
    unsigned short checksum=0;
    int nsepchars=0;
    int n;
    for(n=0; nsepchars < MAX_SEPCHARS && n < strlen(pa110x_response); n++)
    {
        checksum += pa110x_response[n];
        if ( pa110x_response[n] == SEPCHAR )
            ++nsepchars;
    }
    checksum = ~checksum;
    return checksum == xtoi(&pa110x_response[n]);
}

int main(int argc,char* argv[])
{
    if ( argc == 2 )
    {
        printf( "%s\n", validate(argv[1]) ? "valid" : "invalid" );
    }
}
```

DB9F RS232 Pinout

PIN	SIGNAL	IN/OUT *	DESCRIPTION
1	DCD	N/C	Data Carrier Detect
2	RxD	OUT	Receive Data
3	TxD	IN	Transmit Data
4	DTR **	IN	Data Terminal Ready
5	GND	-	Signal Ground
6	DSR ***	OUT	Data Set Ready
7	RTS **	IN	Request To Send
8	CTS ***	OUT	Clear To Send
9	RI	N/C	Ring Indicator

- * IN/OUT Relative to the PA1102 device.
- ** DTR and RTS are required as a power source and must be asserted for data-line powered operation.

DIMENSIONS


1.266
[32,16]
1.420
[36,07]
1.420
[36,07]
3.624
[92,05]
.005
[0,13]
1.680
[42,67]
1.420
[36,07]
INSERT
AREA
.337
[8,56]
.674
[17,12]
INSERT
AREA
2.500
[63,50]
.924
[23,47]
.584
[14,83]
.794
[20,17]
2.500
[63,50]