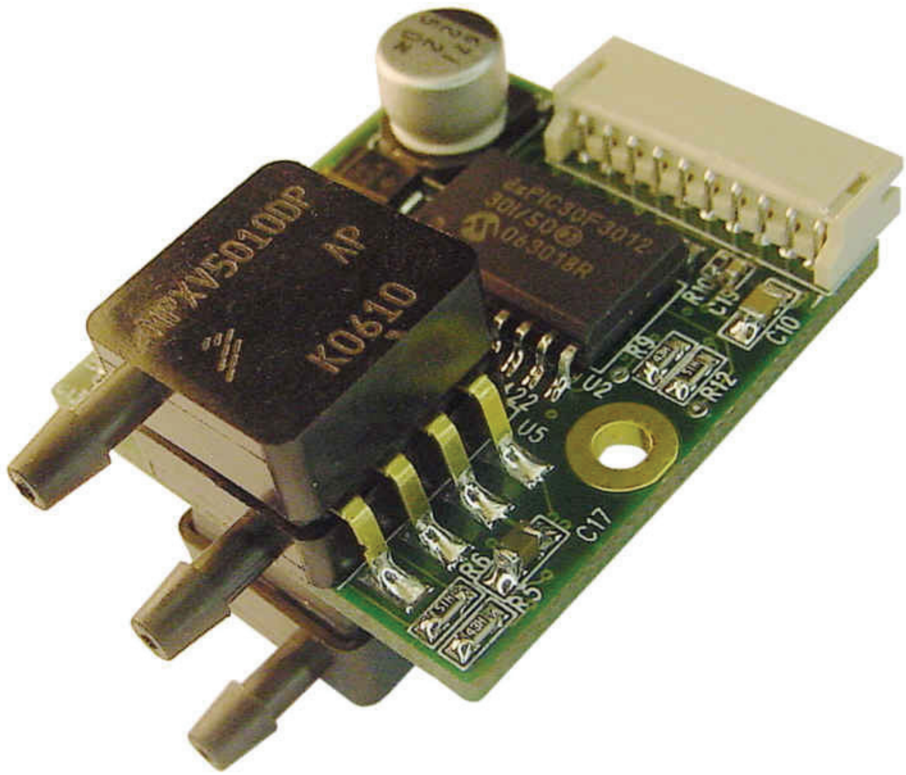
# SPA20422
## AIR DATA SYSTEM




**USER MANUAL**
**V1.0.0**
**October 2008**


MICROBOTICS, INC.
28 Research Drive, Suite G
Hampton, Virginia 23666
+1-757-865-7728
Innovative Navigation and Controls
www.microboticsinc.com

**1 Introduction**. The Microbotics SPA20422 Air Data System is a microprocessor controlled system meant to provide basic altitude and airspeed information for unmanned vehicles. It consists of an absolute barometer (for static pressure), a differential barometer (for pitot pressure), and temperature sensors (both on-board and external) to assist in the airspeed determinations. An on-board microprocessor reads the sensor values, calibrates the readings, and then calculates the target parameters (altitude, airspeed, etc.). A non-volatile EEPROM is available to maintain user settings during power-down. This capability is attained on a circuit board only 1.05 x 1.15 inches (.75 inches thick).

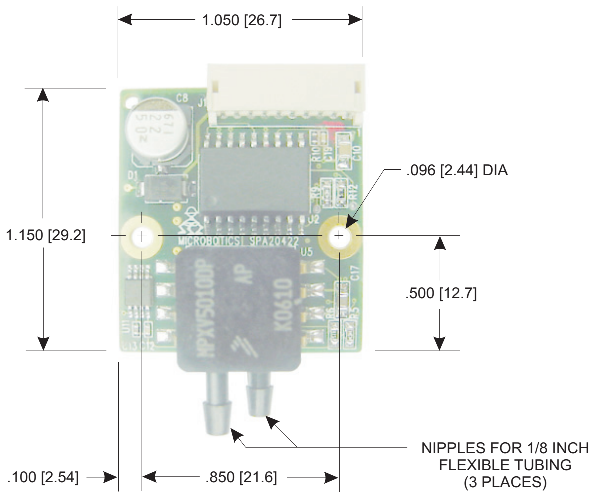



**Figure 1. SPA20422 Physical Dimensions.**

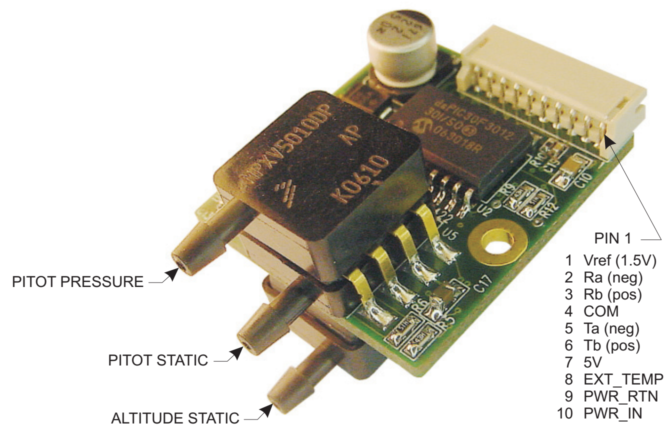



**Figure 2. Connector Pin-Out and Ports Locations.**

$^1/_8$ inch tubing is used for connections to Static Air (for altitude measurement), as well as for the pitot static and dynamic ports (for airspeed measurement). Optional Industry standard temperature probes employing the Analog Devices AD590 are supported for external air temperature measurements.

**2 Wiring the SPA20422 to the User System.** A single10-pin input/output connector is used to provide power, serial data communications, and external temperature probe connections to the SPA20422. The mating connector is a JST Part Number ZHR-10 (housing) using JST Part Number SZH-002T-P0.5 (pins). The pin-out for this connector (Figure 2, Pin 1 to the outside of the board) is:

| | | |
|---|---|---|
| Pin 1 | ***$V_{ref}$*** | Bias reference for RS-422 receiver when TTL communications are used |
| Pin 2 | ***$R_a$*** | RS-422 negative data ***receipt into SPA20422*** (*MARK* Low) |
| Pin 3 | ***$R_b$*** | RS-422 positive data ***receipt into SPA20422*** (*MARK* High) |
| Pin 4 | ***COM*** | RS-422 Ground return – tie to Ground of user serial port |
| Pin 5 | ***$T_a$*** | RS-422 negative data ***transmit from SPS20422*** (*MARK* Low) |
| Pin 6 | ***$T_b$*** | RS-422 positive data ***transmit from SPA20422*** (*MARK* High) |
| Pin 7 | ***5V*** | Power source for external AD590 temperature sensor (positive side of sensor) |
| Pin 8 | ***EXT_TEMP*** | Signal return from external AD590 temperature sensor (negative side of sensor) |
| Pin 9 | ***PWR_RTN*** | Power Ground (negative side) |
| Pin 10 | ***PWR_IN*** | Board Power (6V to 32V input) |

**2.1 Power Connection.** The SPA20422 requires a DC power source of 6-32 VDC, with the positive side of the power source connected to ***PWR_IN*** and the negative side of the power source connected to ***PWR_RTN***. The board uses a switch-mode power converter with a typical constant power dissipation of approximately 150 milliwatts. The ***PWR_RTN*** signal (Pin 9) must be used for the board power return (negative side of power source). While the ***PWR_RTN*** and ***COM*** lines, as well as the two mounting holes, are electrically connected to the board Ground, the ***COM*** line (Pin 4) is used to maintain signal Ground reference for the RS-422 communications, and is not designed nor sized for returning the power supply current. ***DO NOT USE THE COM SIGNAL LINE FOR POWER RETURN.***

**2.2 RS-422 Connections**. The SPA20422 uses an RS-422 differential-mode asynchronous serial data stream for communications to and from the user system. Five signal lines are used for serial communications: ***$T_a$***, ***$T_b$***, ***$R_a$***, ***$R_b$***, and ***COM***. The ***$T_a$*** and ***$T_b$*** pair are the signals ***transmitted from the SPA20422*** to the user system, while the ***$R_a$*** and ***$R_b$*** pair are the signals ***received by the SPA20422*** from the user system. The ***COM*** line is a signal Ground return for the RS-422 signals, and ***must*** be connected to the Ground of the user

RS-422 port. Figure 3 shows a typical connection for the RS-422 signals. ***FAILURE TO CONNECT THE COM LINE TO THE USER PORT GROUND CAN CAUSE ELECTRICAL FAILURE OF THE SPA20422 BOARD AND VOIDS THE BOARD WARRANTY.***

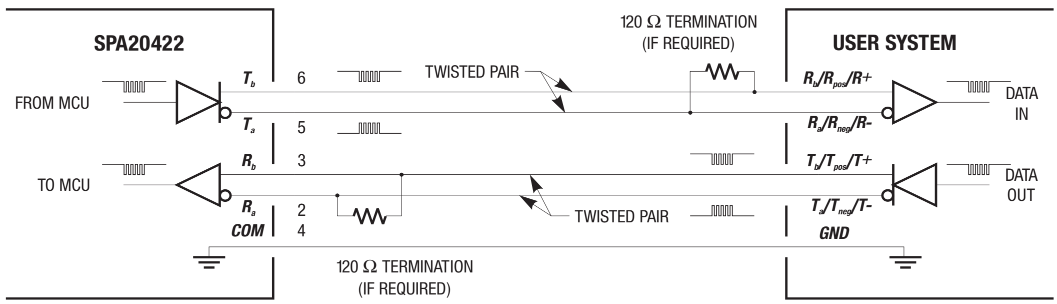



**Figure 3. RS-422 Wiring to User Port.**

**2.3 Connections to RS-232 Systems.** A serial voltage level converter (such as the Microbotics SLC10232) may be used to interface the SPA20422 to an RS-232 port like those used on a PC or laptop computer. A typical connection to a PC-style serial port using an SLC10232 Serial Voltage Level Converter is shown in Figure 4.

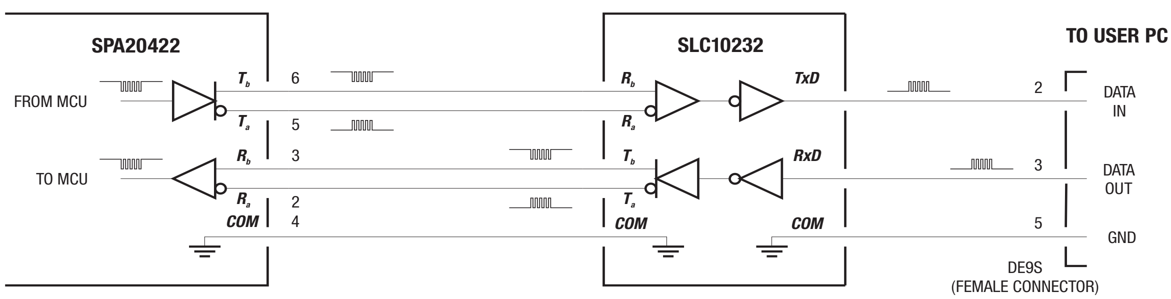



**Figure 4. RS-422 to RS-232 (PC) Wiring Using an SLC10232 Signal Level Converter.**

**2.4 Connections to TTL-Level Systems.** Additionally, the SPA20422 can be used directly with a TTL-compatible serial port (such as a user microprocessor) by using $T_b$ line for data transmission to the user system (the ***Ta*** line is unused), and $R_b$ for data receipt from the user system. The on-board $V_{ref}$ signal is used to bias the $R_a$ signal to allow the SPA20422 to effectively receive the

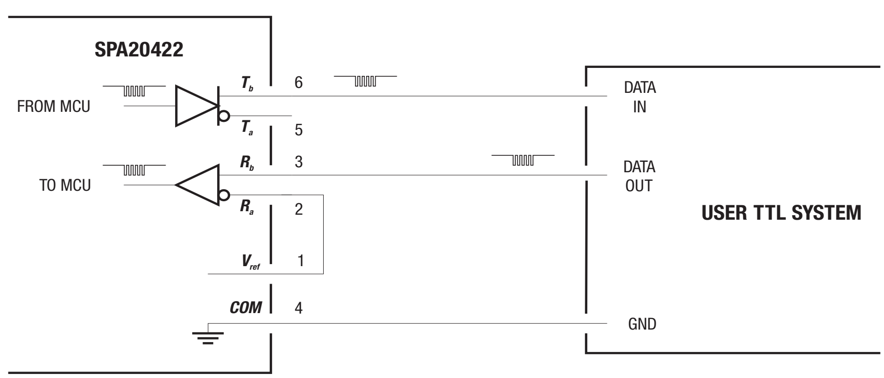



**Figure 5. RS-422Wiring to TTL Port Using $V_{ref}$ to Bias RS-422 Receiver.**

TTL input as a differential signal (see Figure 5). In each of the these wiring examples, the ***COM*** line ***must*** be connected to the Ground of the user port.

**2.5 External Temperature Sensor.** An AD590-style external temperature sensor can be wired to the SPA20422. This type of sensor has a current output of approximately 1 microamp/°K. The sensor is a polarized two-wire device – the negative side of the sensor is connected to the ***EXT_TEMP*** signal, while the positive side of the sensor is connected to the ***5V*** signal (see Figure 6).

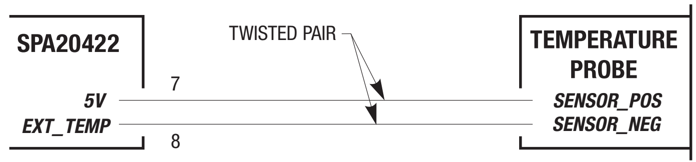



**Figure 6. Wiring to External Temperature Probe.**

**3 Data Provided by the SPA20422.** Physical measurements are made for absolute pressure, differential pressure, internal temperature, and external temperature (if an external temperature probe is present). The on-board microprocessor then calculates altitude, dry air density, and airspeed based on these measurements and the currently set effective sea-level pressure. The user can choose whether the data values are output in SI (metric) or US units. The SPA20422 provides the following air data information to the user (SI units are used to represent the values):

| Parameter | Description | SI Units | US Units |
|---|---|---|---|
| $P_a$ | Absolute barometer pressure | KPa x 100 | in-Hg x 100 |
| $P_d$ | Differential barometer pressure | KPa x 1000 | in-Hg x 1000 |
| $P_o$ | Effective sea-level air pressure | KPa x 100 | in-Hg x 100 |
| $T_{int}$ | On-board temperature | °C x 10 | °F x 10 |
| $T_{ext}$ | External temperature | °C x 10 (if sensor present) | °F x 10 (if sensor present) |
| ***H*** | Altitude | meters x 10 | feet x 10 |
| ***rho*** | Dry air density | kg/cu-meter x 1000 | lb/cu-ft x 1000 |
| ***V*** | Airspeed | KPH x 10 | knots x 10 |

The absolute pressure ($P_a$) is the calibrated pressure reading from the MPX6115 absolute barometer (the sensor with the single pressure port), with an effective range of 15 to 115 KPa (4.43 to 33.96 in-Hg). The differential pressure ($P_d$) is the calibrated pressure reading from the MPX5010 differential barometer (the sensor with two pressure ports), with an effective range of 0 to 10 KPa (0 to 2.95 in-Hg). Due to its low pressure range and environmental effects, this differential sensor can have a zero offset and actually present a negative reading to the microprocessor. The user can reset this pressure reading (i.e., adjust the sensor offset) when the vehicle is at a known stop. The on-board temperature ($T_{int}$) is obtained from an LM50 sensor mounted near the pressure sensors, with an effective range of -40 to +85 °C (-40 to +185 °F). External temperature ($T_{ext}$) is provided by a probe using an AD590 sensor, with an effective range of -55 to +85 °C (-67 to +185 °F). The microprocessor uses these measurements, along with the effective sea-level pressure ($P_o$, explicitly set by the user or derived when adjusting altitude), to calculate altitude (***H***), dry air density (***rho***), and airspeed (***V***).

While the SPA20422 can provide altitude solutions from approximately -1100 to +13750 meters (-3600 to +45000 feet), the effective range is -500 to +7500 meters (-1640 to +24600 feet). Altitude calculations (***H*** in meters) use $P_a$ (in KPa) and $P_o$ (in KPa) to solve the standard altitude equation:

$$H = \frac{10^{\frac{\log \frac{P_a}{P_o}}{5.25588}} - 1}{-2.25574 \times 10^{-5}} \qquad \text{(meters)}$$

If $T_{ext}$ is available due to the presence of an external temperature probe, this value is automatically used in the following calculations. If it not available, $T_{int}$ is used. In the equations below, the temperature $T_k$ (in °K) is derived from $T_{ext}$ or $T_{int}$ :

$$T_k = T_{int} + 273.15 \quad \text{or} \quad T_k = T_{ext} + 273.15 \qquad (°K)$$

While humidity affects the actual velocity calculations, the sensors used by the SPA20422 are not accurate enough to make the effects of humidity significant in the final results. The dry air density (***rho***) is calculated for use in velocity calculations:

$$rho = \frac{P_a}{.2869 \times T_k} \qquad \text{(kg/cu-meter)}$$

The airspeed (***V***) is determined from $P_d$ (in KPa) and ***rho*** (in Kg/cu-meter) by the standard velocity equation:

$$V = \sqrt{\frac{2000 \times P_d}{rho}} \qquad \text{(meters/sec)}$$

**4 Serial Data Communications.** The SPA20422 uses an RS-422 asynchronous serial data port for reporting system values and receiving user inputs. This port is formatted at a fixed 38400 Baud, eight data bits, no parity bit, and one stop bit. When the SPA20422 is initially powered, it sends a title page similar to the following:

```
Microbotics, Inc.
Air Data System
Copyright 2008

Model Number:       SPA20422
Serial Number:      00001234
Software Revision:  V1.0.0
System Build:       Standard
```

After about six seconds, this title block is followed by the system data at the rate and in the protocol set by the user as saved in the non-volatile EEPROM. Both an ASCII and a binary protocol are supported by the SPA20422. The ASCII protocol is more applicable when using a data terminal or a terminal emulation program on a PC, or when gathering data for applications where a string of readable digits can be entered into a spreadsheet-type of program. The binary protocol is better suited for embedded applications (e.g., autopilots, vehicle instrumentation) where the values need to be transferred between sub-systems in binary format with a minimum of translation. The protocol used for output data transmissions can be set by the user, while either binary or ASCII protocols can be employed for user inputs regardless of which output protocol is currently being used.

**4.1 ASCII Output Protocol.** When the ASCII Output Protocol mode is set (the original shipped default), the SPA20422 transmits the current data values in a *SPACE*-delimited sequence. The ASCII output is a free-field (i.e., leading zeros are not sent) series of numbers, each separated by a *SPACE*, with the line terminated by a *Carriage-Return* and *Line-Feed* (CR LF). This output is easily viewed on a terminal device, and the data can be directly input to an Excel file, etc. A typical output stream (with SI units set) would be similar to the following:

```
10164 10133 -260 244 -32768 1188 15 180 0 120
10165 10133 -263 244 -32768 1188 15 176 0 160
10165 10133 -263 245 -32768 1188 15 176 0 200
10164 10133 -259 245 -32768 1188 14 172 0 240
10165 10133 -263 245 -32768 1188 14 172 0 280
10165 10133 -263 246 -32768 1188 14 172 0 320
```

These output values are, in sequence:

| Data Item | Description | SI Units | US Units | Example Above |
|---|---|---|---|---|
| $P_a$ | Absolute Pressure | KPa x 100 | in-Hg x 100 | 10165 = 101.64 KPa |
| $P_o$ | Effective Sea-Level Pressure | KPa x 100 | in-Hg x 100 | 10133 = 101.33 KPa |
| ***H*** | Altitude | meters x 10 | feet x 100 | -260 = -26.0 meters |
| $T_{int}$ | Internal Temperature | °C x 10 | °F x 10 | 244 = 24.4° C |
| $T_{ext}$ | External Temperature | °C x 10 | °C x 10 | -32768 = (Sensor not present) |
| ***rho*** | Dry Air Density | kg/cu-meter x 1000 | lb/cu-ft x 100 | 1188 = 1.188 kg/cu-meter |
| $P_d$ | Differential Pressure | KPa x 1000 | in_hg x 1000 | 14 = .014 KPa |
| ***V*** | Airspeed | KPH x10 | Knots x 10 | 172 = 17.2 KPH |
| ***Status*** | System Status | 16-bit bit field – Value dependent upon system status | | |
| ***UTime*** | Free-Running Timer | Unsigned 16-bit value – Number of 50 ms periods since reset | | |

The $T_{int}$ and $T_{ext}$ values can be negative; however, a value of -32768 indicates a missing or faulty temperature sensor. The $P_d$ value can be negative due to sensor offset (this offset can be reset by the user), with the airspeed being clamped at zero in this event. ***Status*** is a bit field providing certain operational indicators (discussed in Paragraph 4.6). ***UTime*** is a free-running recirculating 16-bit counter (modulo-65536) indicating the number of 50 millisecond periods since power was applied, and is provided as an interval counter and missed-packet indicator. Note that the $P_a$, $P_o$, ***rho***, and ***V*** values can never be negative.

**4.2 ASCII Input Protocol.** User input can use the ASCII input protocol at any time, regardless of which protocol is set for data transmission. Character echoing is not provided for ASCII input data streams, nor are any confirmations transmitted for any command entered via the ASCII protocol. Data values can be entered free-field (i.e., leading zeros do not need to be entered). An *ESC* can be entered, followed by a *Carriage-Return*/*Line-Feed* ($^{C}_{R}{}^{L}_{F}$) sequence, to abort a command in progress. An ASCII input command has the following form:

**~** [*Command_Letter*] [*Value, if required or optional*] $^{C}_{R}{}^{L}_{F}$

The tilde ('**~**') opens the command packet and indicates to the SPA20422 that an ASCII command is being entered. The *Command_Letter*, which immediately follows the tilde, is a single alphabetic character, and is case sensitive. A *Value*, when required or optional, is a numeric entry up to 8 digits. Negative values can be entered if needed, and, if a value is required but not included, it has an assumed value of zero. Values are ignored when entered for commands not requiring a value. The *Carriage-Return*/*Line-Feed* ($^{C}_{R}{}^{L}_{F}$) sequence completes the command. Valid user commands are:

| | |
|---|---|
| ***~a*** $^{C}_{R}{}^{L}_{F}$ | Set ASCII output protocol |
| ***~b*** $^{C}_{R}{}^{L}_{F}$ | Set binary output protocol |
| ***~e*** $^{C}_{R}{}^{L}_{F}$ | Write to EEPROM |
| ***~h*** [*value*] $^{C}_{R}{}^{L}_{F}$ | Update altitude, required *value* in meters x 1 or feet x 1 |
| ***~m*** [*value*] $^{C}_{R}{}^{L}_{F}$ | Poll data values, optional *value* sets output message interval in units of 50 ms or off if zero |
| ***~r*** [*value*] $^{C}_{R}{}^{L}_{F}$ | Update effective sea-level pressure ($P_o$), required *value* in KPa x 100 or in-Hg x 100 |
| ***~s*** $^{C}_{R}{}^{L}_{F}$ | Set SI (metric) units |
| ***~u*** $^{C}_{R}{}^{L}_{F}$ | Set US units |
| ***~v*** $^{C}_{R}{}^{L}_{F}$ | Reset differential pressure ($P_d$) |

Invalid commands, or commands with invalid values, are ignored. No confirmation of success or failure of any operation is sent by the SPA20422 when an ASCII input command is used.

**4.2.1 Set Output Data Transmission Protocol.** Two ASCII input commands are used to set the output data transmission protocol: the ***~a*** $^{C}_{R}{}^{L}_{F}$ command sets the ASCII output protocol (the original shipped default), while the ***~b*** $^{C}_{R}{}^{L}_{F}$ command sets the binary output protocol. Any *value* entered is ignored. The new output protocol becomes active at the next output message, and remains in effect until explicitly changed or power is cycled (turned off, then on) for the board. Whenever a command for data transmission protocol is received, the SPA20422 sets the ***EE_Needs_Update*** bit in ***Status***. Writing to the User EEPROM is optional, but if the EEPROM is not written before power-down after the data transmission protocol has been changed, the SPA20422 will revert to the protocol stored in the EEPROM at the next power-up.

**4.2.2 Set Output Data Transmission Units.** The output data may be transmitted in either SI (metric – the original shipped default) or US units. Two ASCII commands are used to set the units used in the output data transmissions: the ***~s*** $^{C}{}_{R}{}^{L}{}_{F}$ command sets SI units, while the ***~u*** $^{C}{}_{R}{}^{L}{}_{F}$ command sets US units. Any *value* entered is ignored. The new units setting will become active at the next output message, and remains in effect until explicitly changed or power is cycled (turned off, then on) for the board. Whenever a command for data transmission protocol is received, the SPA20422 sets the ***EE_Needs_Update*** bit in ***Status***. Writing to the User EEPROM is optional, but if the EEPROM is not written before power-down after the units setting has been changed, the SPA20422 will revert to the units setting stored in the EEPROM at the next power-up.

**4.2.3 Poll Data Values and Set Output Message Interval.** The ***~m*** [*value*] $^{C}{}_{R}{}^{L}{}_{F}$ command is used to poll the current values, and, optionally, set the output data message rate. Whenever this command is received, the SPA20422 immediately sends a data output message (using whichever output data transmission protocol has be set). If no *value* is included in the command, the message update rate is not affected, and message updates continue at the currently set interval. If the optional *value* is included, the *value* changes the output data message rate in units of 50 milliseconds (from 1 to 100 – *value* entries over 100 are forced to 100 – netting intervals from 50 milliseconds to 5 seconds), or turns the output transmissions off if the *value* is zero. This new message update rate becomes effective immediately after the polled data message is sent. Whenever the data transmission rate is set (an actual *value* received), the SPA20422 sets the ***EE_Needs_Update*** bit in ***Status***. Writing to the User EEPROM is optional, but if the EEPROM is not written before power-down after the data transmission rate has been set, the SPA20422 will revert to the rate stored in the EEPROM at the next power-up.

**4.2.4 Update Altitude or Effective Sea-Level Pressure.** Weather induced barometric changes, as well as long term drift in the absolute barometer output, affect the altitude calculations of the SPA20422. These effects can be compensated by updating the current altitude (e.g., as derived from a known location, a GPS reading, or an external sensor) by issuing the ***~h*** [*value*] $^{C}{}_{R}{}^{L}{}_{F}$ command. The required *value* parameter is the new altitude in [meters x 1] if SI units are set or in [feet x 1] if US units are set, and, if it is missing, the *value* is assumed to be zero. This command attempts to reset the current altitude by changing the effective sea-level pressure ($P_o$) setting used for the altitude calculation. If the command is successful, the ***EE_Needs_Update*** bit in ***Status*** is set. If the *value* entered is out of range, or if it attempts to create an invalid $P_o$ setting, the command is ignored. No confirmation of success or failure of the operation is sent by the SPA20422 when this command is used.

An alternate method of correcting altitude is by using the ***~r*** [*value*] $^{C}{}_{R}{}^{L}{}_{F}$ command to explicitly change the $P_o$ value used in the calculations. The required *value* is the new $P_o$ in [KPa x 100] if SI units are set or in [in-Hg x 100] if US units are set (e.g., 10133 is 101.33 KPa for SI units, or 2992 is 29.92 in-Hg for US units). The input range for the new $P_o$ is restricted to 90-110 KPa (26.57-32.48 in-Hg), and, if the value is out of range or missing, the command is ignored. If the command is successful, the ***EE_Needs_Update*** bit in ***Status*** is set. Note that, as this command explicitly sets $P_o$, it will supercede any previous changes made with the ***~h*** [*value*] $^{C}{}_{R}{}^{L}{}_{F}$ command. If both commands are needed, the ***~r*** [*value*] $^{C}{}_{R}{}^{L}{}_{F}$ command should be used before the ***~h*** [*value*] $^{C}{}_{R}{}^{L}{}_{F}$ command. No confirmation of success or failure of the operation is sent by the SPA20422 when this command is used.

With either of these commands, writing to the User EEPROM is optional, but if the EEPROM is not written before power-down after the $P_o$ value has been set, the SPA20422 will revert to the $P_o$ value stored in the EEPROM at the next power-up.

**4.2.5 Reset Differential Pressure.** With the differential pressure readings quite low, long term sensor drift can create an offset in the actual value read by the SPA20422 (either positive or negative), and will affect the airspeed calculation. While the board will report a "negative" pressure reading, also setting the ***$P_d$_Neg*** bit in ***Status***, this negative value for $P_d$ is clipped to zero for the airspeed calculation (i.e., the airspeed will be set to zero). When the vehicle as at a known stop, the $P_d$ reading can be forced to zero by using the ***~v*** $^{C}{}_{R}{}^{L}{}_{F}$ command. Any *value* entered is ignored, the $P_d$ is reset to zero (resetting the airspeed ***V*** to zero), and the ***EE_Needs_Update*** bit in ***Status*** is set. If the command cannot reset $P_d$ to zero (e.g., the vehicle is actually moving when the command is issued), $P_d$ is not changed, and the ***EE_Needs_Update*** bit in ***Status*** is not set. No confirmation of success or failure of the operation is sent by the SPA20422 when this command is used..

Writing to the User EEPROM is optional, but if the EEPROM is not written before power-down after the $P_d$ value has been reset, the SPA20422 will revert to the $P_d$ offset value stored in the EEPROM at the next power-up.

**4.2.6 Write to EEPROM.** A non-volatile EEPROM is used to maintain certain system information while the SPA20422 is not powered. These data include user changeable parameters: output data transmission protocol, output data transmission units, output message rate, current effective sea-level pressure ($P_o$), and the offset value for the differential sensor. Whenever any of these parameters have been changed or updated during operation, the ***EE_Needs_Update*** bit in ***Status*** is set to indicate the values in the EEPROM might not reflect the current status. The ***~e*** ${}^{C}_{R}{}^{L}_{F}$ command is used to write the updated values to the EEPROM. Any *value* included in the command is ignored.

When the SPA20422 starts the write operation, it checks whether the parameters stored in the EEPROM match the current status. If they do, no EEPROM write is actually performed. If any parameter has been changed, the new values are saved to the EEPROM. If an error occurs during the writing process, the ***EE_Write_Error*** bit in ***Status*** is set until another write is attempted or power is removed from the board. Upon a successful write, the ***EE_Needs_Update*** bit is reset. The EEPROM has a limited erase/write life of over two million cycles (more than adequate for the service life of the SPA20422). Two bits in ***Status*** are used to keep track of the approximate life left on the EEPROM (***EE_Status_1*** and ***EE_Status_0***) which indicate the remaining EEPROM life:

| | | |
|---|---|---|
| ***EE_Status_1*: *EE_Status_0*** | **0:0** | Over half of EEPROM life remaining |
| ***EE_Status_1*: *EE_Status_0*** | **0:1** | Less than half of EEPROM life remaining |
| ***EE_Status_1*: *EE_Status_0*** | **1:0** | Less than 10% of EEPROM life remaining |
| ***EE_Status_1*: *EE_Status_0*** | **1:1** | EEPROM exhausted – no more writing allowed |

Even when the EEPROM is exhausted, the board will still operate properly (i.e., output mode, units, and message rate can be changed, $P_o$ and altitude can be updated, and the differential pressure can be reset), but the changes cannot be updated in EEPROM to effect power-up defaults. No confirmation of success or failure of the write operation is sent by the SPA20422 when this command is used.

**4.3 Microbotics Standard Binary Protocol.** Both the binary output protocol and the binary input protocol use the standard Microbotics Binary Protocol. This is a serial data stream of 8-bit bytes with the following basic format:

[*Sync0*] [*Sync1*] [*Packet_ID*] [*Payload_count*] [*Payload bytes, if any*] [*CS0*] [*CS1*]

The *Sync0* and *Sync1* bytes are used to indicate the beginning of a specific message, and have the binary values of **0x81** and **0xA1**, respectively. The *Packet_ID* is an unsigned 8-bit integer identifying the command type. The *Payload_count* is an unsigned 8-bit integer indicating the number of ***bytes*** in the payload to follow. This count can be **0x00** (no payload) through **0xFF** (255 theoretic maximum ***bytes*** in the payload).

The payload format is specific to the *Packet_ID*, with the payload being any mix of bytes, 16-bit words, or 32-bit words. Payload values are sent in big-endian format – the first byte of each multi-byte value contains the most significant bits of that value.

The *CS0* and *CS1* form a two-byte Fletcher checksum of all ***bytes*** between *Sync1* and *CS0* (i.e., the checksum includes *Packet_ID*, *Payload_count*, and all payload bytes). The Fletcher checksum is calculated in the following manner:

| | |
|---|---|
| *CS0* and *CS1* are initially both set to zero | *CS0* = *CS1* = 0; |
| The next byte is added to the current *CS0* value to form the new *CS0* value [modulo-256] | *CS0* += [*byte_value*]; |
| The new *CS0* value is added to the current *CS1* value to form the new *CS1* value [modulo-256] | *CS1* += *CS0*; |
| Continue for all bytes | *loop;* |
| *CS0* is sent first, followed by *CS1* | *CS0*: *CS1*; |

**4.4 Binary Output Protocol.** If the binary output protocol is currently set and the output message interval is not zero, the SPA20422 will transmit binary Data Messages at the output message interval. Additionally, if any Update Commands have been sent using the binary input protocol, the Confirm Messages for these commands will be sent at the next 50 millisecond processing period, regardless of the output message interval (commands sent using the ASCII input protocol do not have associated Confirm Messages). Note that the output message protocol can only be changed by using the ASCII input messages ***~a*** ${}^{C}_{R}{}^{L}_{F}$ to set the ASCII output protocol, or ***~b*** ${}^{C}_{R}{}^{L}_{F}$ to set the binary output protocol. The ASCII output protocol is the original shipped default.

**4.4.1 Data Message.** At each output message interval, a Data Message is sent. This message has a *Packet_ID* of **0x01** and a fixed *Payload_count* of **0x16** (22 bytes in the payload), with the following format:

[**0x81**] [**0xA1**] [**0x01**] [**0x16**] [*22-byte payload, 10 data items*] [*CS0*] [*CS1*]

The payload is a sequence of binary data values sent in big-endian format (i.e., the most significant byte of a multi-byte data value is sent first). The data in either SI or US units. The output units can only be set by using the ASCII input messages ***~s*** $^{C}{}_{R}{}^{L}{}_{F}$ to set SI units for the data values, or ***~u*** $^{C}{}_{R}{}^{L}{}_{F}$ to set US units for the data values. The data are output in the following order:

| Data Item | Data Size | Description | Units |
|---|---|---|---|
| ***Status*** | 2 bytes, 16-bit bit field | System Status | Value dependent upon system status |
| ***UTime*** | 2 bytes, 16-bit unsigned integer | Running Timer | Recirculating count of 50 ms periods since reset |
| $P_a$ | 2 bytes, 16-bit unsigned integer | Absolute Pressure | KPa x100 or in-Hg x 100 |
| $P_o$ | 2 bytes, 16-bit unsigned integer | Effective Sea-Level Pressure | KPa x100 or in-Hg x 100 |
| ***H*** | 4 bytes, 32-bit signed integer | Altitude | meters x10 or feet x 10 |
| $T_{int}$ | 2 bytes, 16-bit signed integer | Internal Temperature | °C x10 or °F x10 |
| $T_{ext}$ | 2 bytes, 16-bit signed integer | External Temperature | °C x10 or °F x10, **0x8000** = Sensor not present |
| ***rho*** | 2 bytes, 16-bit unsigned integer | Dry Air Density | kg/cu-meter x1000 or lb/cu-ft x 1000 |
| $P_d$ | 2 bytes, 16-bit signed integer | Differential Pressure | KPa x1000 or in-Hg x 1000 |
| ***V*** | 2 bytes, 16-bit unsigned integer | Airspeed | KPH x10 or knots x 10 |

***Status*** is a bit field providing certain operational indicators (discussed in Paragraph 4.6). ***UTime*** is a free-running recirculating 16-bit counter (modulo-65536) indicating the number of 50 ms periods since power was applied, and is provided as an interval counter and missed-packet indicator. The $T_{int}$ and $T_{ext}$ values can be negative; however, a value of **0x8000** indicates a missing or faulty temperature sensor. The $P_d$ value can be negative due to sensor offset, with the airspeed being clamped at zero in this event. Note that the $P_a$, $P_o$, ***rho***, and ***V*** values can never be negative.

**4.4.2 Confirm Message.** Whenever a binary Update Command is issued, a Confirm Message is sent at its completion. Note that any command sent using the ASCII input protocol does not cause a Confirm Message transmission. If several commands have been executed, their Confirm Messages will queue up to transmit in the subsequent 50 millisecond processing period. Confirm Messages are sent at the first 50 millisecond processing period after completion or queuing, and are sent regardless of the output message update rate setting. If a Confirm Messages are being sent during the same 50 millisecond processing as a Data Message, the Confirm Messages will follow the Data Message. A Confirm Message has a *Packet_ID* of **0x03** and a fixed *Payload_count* of **0x06** (6 bytes in the payload), with the following format:

[**0x81**] [**0xA1**] [**0x03**] [**0x06**] [*6-byte payload, 4 data items*] [*CS0*] [*CS1*]

The payload is a sequence of binary data values sent in big-endian format (i.e., the most significant byte of a data value is sent first), in the following order:

| Data Item | Data Size | Description | Value |
|---|---|---|---|
| ***Status*** | 2 bytes, 16-bit bit field | System Status | Value dependent upon system status |
| ***UTime*** | 2 bytes, 16-bit unsigned integer | Running Timer | Number of 50 ms periods since reset |
| ***Sub_command*** | 1 byte, 8-bit unsigned integer | *Sub_command* confirmed | *Sub_command* number of Update Command |
| ***Update_status*** | 1 byte, 8-bit bit field | Execution status | Value dependent upon execution status |

***Status*** is a bit field providing certain operational indicators (discussed in Paragraph 4.6). ***UTime*** is a free-running recirculating 16-bit counter (modulo-65536) indicating the number of 50 millisecond periods since power was applied, and is provided as an interval counter and missed-packet indicator. ***Sub_command*** is the *Sub_command* of the Update Command that is being confirmed (see

Paragraph 4.5.2 for valid *Sub_commands*). ***Update_status*** is the execution status of the command, and is presented as an 8-bit bit field (see the individual sub-commands below for a description of ***Update_status***).

**4.5 Binary Input Protocol.** User input can use the binary input protocol at any time, regardless of which protocol is set for data transmission. The binary input protocol uses only two messages: one to poll the data values and set the update interval (*Packet_ID* of **0x01**), and the other to update system parameters (*Packet_ID* of **0x03**):

[**0x81**] [**0xA1**] [**0x01**] [*Payload_count =* **0x00** *or* **0x01**] [*Optional interval value* **0x00 - 0x64**] [*CS0*] [*CS1*]
[**0x81**] [**0xA1**] [**0x03**] [*Payload_count*] [*Sub_command*] [*update_value, if any*] [*CS0*] [*CS1*]

Any other binary input message is ignored, with no error indicator sent by the SPA20422.

**4.5.1 Poll Data Values and Set Update Interval.** A message with *Packet_ID* of **0x01** is used to poll the current values, and, optionally, set the output data message rate:

[**0x81**] [**0xA1**] [**0x01**] [*Payload_count =* **0x00** *or* **0x01**] [*Optional interval value* **0x00 - 0x64**] [*CS0*] [*CS1*]

Whenever this command is received, the SPA20422 immediately sends a data output message (using whichever output data transmission protocol has be set). If no *interval value* is included in the command, the message update rate is not affected, and message updates continue at the currently set interval. If the optional *interval value* is included, the *interval value* changes the output data message rate in units of 50 milliseconds – **0x01** to **0x64** (1 to 100 – *interval value* entries over **0x64** are forced to **0x64**) – netting intervals from 50 milliseconds to 5 seconds, or turns the output transmissions off if the *interval value* is **0x00**. This new message update rate becomes effective immediately after the polled data message is sent. Whenever the data transmission rate is set (an actual *value* received), the SPA20422 sets the ***EE_Needs_Update*** bit in ***Status***. Writing to the User EEPROM is optional, but if the EEPROM is not written before power-down after the data transmission rate has been set, the SPA20422 will revert to the rate stored in the EEPROM at the next power-up.

**4.5.2 Update Commands.** The Update Commands are used to reset the differential pressure, update the effective sea-level pressure ($P_o$), update the current altitude, and write to the EEPROM. An Update Command takes the basic form:

[**0x81**] [**0xA1**] [**0x03**] [*Payload_count*] [*Sub_command*] [*update_value, if any*] [*CS0*] [*CS1*]

All Update Commands use a *Packet_ID* of **0x03**. The first byte of the payload is the *Sub_command* defining the particular update, which may be followed by an *update_value* as needed. The valid Update Commands are:

| | | |
|---|---|---|
| ***Reset_P$_d$*** | *Sub_command* = **0x00** | [**0x81**] [**0xA1**] [**0x03**] [**0x01**] [**0x00**] [**0x26**] [**0x14**] |
| ***Update_P$_o$*** | *Sub_command* = **0x01** | [**0x81**] [**0xA1**] [**0x03**] [**0x03**] [**0x01**] [*2-byte value*] [*CS0*] [*CS1*] |
| ***Update_Altitude*** | *Sub_command* = **0x02** | [**0x81**] [**0xA1**] [**0x03**] [**0x05**] [**0x02**] [*4-byte value*] [*CS0*] [*CS1*] |
| ***Write _to_EEPROM*** | *Sub_command* = **0x07** | [**0x81**] [**0xA1**] [**0x03**] [**0x01**] [**0x07**] [**0x2D**] [**0x1B**] |

Note the ***Reset_P$_d$*** and ***Write _to_EEPROM*** commands are fixed-format commands with only one payload byte (the *Sub_command* itself), thus the *CS0* and *CS1* for these commands are known.

Valid Update Commands issue Confirm Messages upon completion or error:

| | |
|---|---|
| ***Reset_P$_d$*** | [**0x81**] [**0xA1**] [**0x03**] [**0x06**] [*UTime*] [*Status*] [**0x00**] [*Update_status*] [*CS0*] [*CS1*] |
| ***Update_P$_o$*** | [**0x81**] [**0xA1**] [**0x03**] [**0x06**] [*UTime*] [*Status*] [**0x01**] [*Update_status*] [*CS0*] [*CS1*] |
| ***Update_Altitude*** | [**0x81**] [**0xA1**] [**0x03**] [**0x06**] [*UTime*] [*Status*] [**0x02**] [*Update_status*] [*CS0*] [*CS1*] |
| ***Write _to_EEPROM*** | [**0x81**] [**0xA1**] [**0x03**] [**0x06**] [*UTime*] [*Status*] [**0x07**] [*Update_status*] [*CS0*] [*CS1*] |

A message with an invalid *Sub_command* will be ignored without any error message sent. Except for the ***Write_to_ EEPROM*** command, multiple Update Commands may be sent in any sequence without waiting for Confirm Messages. However, the commands are dealt with in priority (with the lowest *Sub_command* number having the highest priority) at each 50 millisecond processing period. If the user is not careful of the order of the Update Commands when not waiting for confirmations, it is possible the commands may be processed in a different order than received, and the execution of these commands may not result in the desired updates. Since the Update Commands are processed during each 50 millisecond processing interval, multiple issues of the same command before its Confirm Message is sent will result in only the last command being executed, and only one Confirm Message will be issued.

**4.5.2.1 Reset Differential Pressure.** With the differential pressure readings quite low, long term sensor drift can create an offset in the actual value read by the SPA20422 (either positive or negative), and will affect the airspeed calculation. While the board will report a "negative" pressure reading, also setting the ***$P_d$_Neg*** bit in ***Status***, the negative value is clipped to zero for the airspeed calculation (the airspeed will be forced to zero). When the vehicle as at a known stop, the ***$P_d$*** reading can be forced to zero by using ***Reset_$P_d$*** (*Sub_command* = **0x00**):

[**0x81**] [**0xA1**] [**0x03**] [**0x01**] [**0x00**] [**0x26**] [**0x14**]

If the command is successful, ***$P_d$*** is reset to zero (resetting the airspeed ***V*** to zero), and the ***EE_Needs_Update*** bit in ***Status*** is set. If the command cannot reset ***$P_d$*** to zero (e.g., the vehicle is actually moving when the command is issued), ***$P_d$*** is not changed, and the ***EE_Needs_Update*** bit in ***Status*** is not set. Writing to the User EEPROM is optional, but if the EEPROM is not written before power-down after the ***$P_d$*** value has been reset, the SPA20422 will revert to the ***$P_d$*** offset value stored in the EEPROM at the next power-up.

The *Update_status* byte in the ***Reset_$P_d$*** Confirm Message indicates:

| | |
|---|---|
| **0x00** | Successful operation, ***$P_d$*** reset to zero |
| **0x08** | ***$P_d$*** too high to be reset, command ignored |

**4.5.2.2 Update Altitude.** Weather induced barometric changes, as well as long term drift in the absolute barometer output, affect the altitude calculations of the SPA20422. These effects can be compensated by updating the current altitude (e.g., as derived from a known location, a GPS reading, or an external sensor) by using ***Update_Altitude*** (*Sub_command* = **0x02**):

[**0x81**] [**0xA1**] [**0x03**] [**0x05**] [**0x02**] [*4-byte value*] [*CS0*] [*CS1*]

The required *4-byte value* parameter is the new altitude in [meters x 100] if SI units are set or in [feet x 100] if US units are set, and may be negative. Thus, if SI units are set, a *4-byte value* of **0x0000'7D14** reflects a new altitude request of 320.20 meters, while a *4-byte value* of **0xFFFF'FB0A** reflects a new altitude request of –12.70 meters. If the value is missing, it is assumed to be zero. This command attempts to reset the current altitude by changing the effective sea-level pressure (***$P_o$***) setting used for the altitude calculation. If the command is successful, the ***EE_Needs_Update*** bit in ***Status*** is set. If the *4-byte value* entered is out of range, or if it attempts to create an invalid ***$P_o$*** setting, the command is ignored. Writing to the User EEPROM is optional, but if the EEPROM is not written before power-down after the ***$P_o$*** value has been set, the SPA20422 will revert to the ***$P_o$*** value stored in the EEPROM at the next power-up.

The *Update_status* byte in the ***Update_Altitude*** Confirm Message indicates:

| | |
|---|---|
| **0x00** | Successful operation, ***$P_o$*** adjusted |
| **0x01** | New altitude request trying to force ***$P_o$*** too low, command ignored |
| **0x02** | New altitude request trying to force ***$P_o$*** too high, command ignored |
| **0x04** | New altitude request too low, command ignored |
| **0x08** | New altitude request too high, command ignored |

**4.5.2.3 Update Effective Sea-Level Pressure.** An alternate method of correcting altitude is by using ***Update_P$_o$*** (*Sub_command* = **0x02**) to explicitly change the ***P$_o$*** value used in the calculations:

[**0x81**] [**0xA1**] [**0x03**] [**0x03**] [**0x01**] [*2-byte value*] [*CS0*] [*CS1*]

The required *2-byte value* is the new ***P$_o$*** in [KPa x 100] if SI units are set or in [in-Hg x 100] if US units are set (e.g., if SI units are set, a *2-byte value* of **0x2795** is 101.33 KPa). The input range for the new ***P$_o$*** is restricted to 90-110 KPa (26.57-32.48 in-Hg), and, if the value is out of range or missing, the command is ignored. If the command is successful, the ***EE_Needs_Update*** bit in ***Status*** is set.

Note that, as this command explicitly sets ***P$_o$***, it will supercede any previous changes made with the ***Update_Altitude*** command. If both commands are needed, the ***Update_P$_o$*** command should be used before the ***Update_Altitude*** command.

Writing to the User EEPROM is optional, but if the EEPROM is not written before power-down after the ***P$_o$*** value has been set, the SPA20422 will revert to the ***P$_o$*** value stored in the EEPROM at the next power-up.

The *Update_status* byte in the ***Update_P$_o$*** Confirm Message indicates:

| | |
|---|---|
| **0x00** | Successful operation, ***P$_o$*** adjusted |
| **0x01** | New ***P$_o$*** request too low, command ignored |
| **0x02** | New ***P$_o$*** request too high, command ignored |

**4.5.2.4 Write to EEPROM.** A non-volatile EEPROM is used to maintain certain system information while the SPA20422 is not powered. These data include user changeable parameters: the output data transmission protocol, the output message rate, the current effective sea-level pressure (***P$_o$***), and the offset value for the differential sensor. Whenever any of these parameters have been changed or updated during operation, the ***EE_Needs_Update*** bit in ***Status*** is set to indicate the values in the EEPROM might not reflect the current status. The ***Write_to_EEPROM*** command is used to write to the EEPROM (*Sub_command* = **0x07**):

[**0x81**] [**0xA1**] [**0x03**] [**0x01**] [**0x07**] [**0x2D**] [**0x1B**]

When the SPA20422 starts the write operation, it checks to see if the ***EE_Needs_Update*** bit in ***Status*** is set, and, if so, whether the values stored in the EEPROM match the current status. If the ***EE_Needs_Update*** bit is not set, or the current status matches the stored status, no EEPROM write is actually performed. If any parameter has been changed, the new values are saved to the EEPROM. If an error occurs during the writing process, the ***EE_Write_Error*** bit in ***Status*** is set until another write is attempted or power is removed from the board. Upon a successful write, the ***EE_Needs_Update*** fbit is reset. The EEPROM has a limited erase/write life of over two million cycles (more than adequate for the service life of the SPA20422). Two bits in ***Status*** are used to keep track of the approximate life left on the EEPROM (***EE_Status_1*** and ***EE_Status_0***) which indicate the remaining EEPROM life:

| | | |
|---|---|---|
| ***EE_Status_1*: *EE_Status_0*** | **0:0** | Over half of EEPROM life remaining |
| ***EE_Status_1*: *EE_Status_0*** | **0:1** | Less than half of EEPROM life remaining |
| ***EE_Status_1*: *EE_Status_0*** | **1:0** | Less than 10% of EEPROM life remaining |
| ***EE_Status_1*: *EE_Status_0*** | **1:1** | EEPROM exhausted – no more writing allowed |

Even when the EEPROM is exhausted, the board will still operate properly (i.e., output mode, units, and message rate can be changed, ***P$_o$*** and altitude can be updated, and the differential pressure can be reset), but the changes cannot be updated in EEPROM to effect power-up defaults.

The ***Write_to_EEPROM*** command will not execute if any Confirm Message is pending or queued. In this case, the write operation is aborted, but the ***EE_Needs_Update*** bit in ***Status*** remains set, and the appropriate Confirm Message for this write error is sent. Multiple ***Write_to_EEPROM*** commands received within the same 50-millisecond processing period will result in only one write operation and only one Confirm Message.

The *Update_status* byte in the ***Write_to_EEPROM*** Confirm Message indicates:

| | |
|---|---|
| **0x00** | Successful operation, EEPROM values have been updated |
| **0x01** | ***EE_Needs_Update*** in ***Status*** not set, command ignored |
| **0x02** | Current status matches stored status, command ignored |
| **0x03** | Confirms still pending from other ***Update Status*** commands, command ignored, ***EE_Needs_Update*** in ***Status*** set |
| **0x04** | Write verify of EEPROM failed, contents of EEPROM may be invalid, both ***EE_Write_Error*** and ***EE_Needs_Update*** in ***Status*** set |
| **0x05** | EEPROM exhausted, command ignored, ***EE_Needs_Update*** bit in ***Status*** remains set |

**4.6 *Status* Bit Fields.** The current status of the SPA20422 system operation is maintained in ***Status***. This 16-bit bit-field parameter is sent with each ASCII output data message, and each binary message. Many of the status bits affect other bits in ***Status***. The meanings of these bits are (Bit 15 is the most significant bit):

| Bit | Indicator | Value |
|---|---|---|
| Bit 15 | ***Units*** | **0** = SI units, **1** = US units |
| Bit 14 | ***5V_error*** | **1** = 5V power supply out of range |
| Bit 13 | ***2.5V_error*** | **1** = 2.5V reference out of range |
| Bit 12 | ***Temp_error*** | **1** = Cannot determine temperature |
| Bit 11 | ***altitude_error*** | **1** = Cannot calculate altitude |
| Bit 10 | ***rho_error*** | **1** = Cannot calculate air density |
| Bit 9 | ***speed_error*** | **1** = Cannot calculate airspeed |
| Bit 8 | ***$P_a$_error*** | **1** = Cannot determine $P_a$ |
| Bit 7 | ***$P_d$_error*** | **1** = Cannot determine $P_d$ |
| Bit 6 | ***$P_d$_Neg*** | **1** = $P_d$ has negative value |
| Bit 5 | (*Not used*) | **0** |
| Bit 4 | (*Not used*) | **0** |
| Bit 3 | ***EE_Write_Error*** | **1** = EEPROM write operation failed, EEPROM exhausted, or EEPPOM invalid at power-up |
| Bit 2 | ***EE_Needs_Update*** | **1** = A parameter was changed, EEPROM may need to be updated |
| Bit 1 | ***EE_Status_1*** | (See Paragraphs 4.2.6 and 4.5.2.4) |
| Bit 0 | ***EE_Status_0*** | (See Paragraphs 4.2.6 and 4.5.2.4) |

If the 5V power supply used to power the on-board sensors of the SPA20422 is out of range, the ***5V_error*** bit is set in ***Status***. In this case, neither $P_a$ nor $P_d$ can be properly measured, nor can the altitude and speed calculations be performed. Therefore, the ***$P_d$_error***, ***$P_a$_error***, ***altitude_error***, ***rho_error***, and ***speed_error*** bits will also be set in ***Status***.

If the absolute pressure sensor is faulty, or the ***5V_error*** bit is set, the ***$P_a$_error*** bit is set in ***Status***. Altitude and speed cannot be properly calculated, therefore the ***altitude_error***, ***rho_error***, and ***speed_error*** bits will also be set in ***Status***.

If the differential pressure sensor is faulty, or teh ***5V_error*** bit is set, the ***$P_d$_error*** bit is set in ***Status***. Speed cannot be calculated, setting the ***speed_error*** bit in ***Status***. If the differential pressure is measured as negative (due to an offset in the sensor when the vehicle is stopped), the ***$P_d$_Neg*** bit is set in ***Status***, while the airspeed (***V***) is forced to zero.

If the 2.5V reference of the SPA20422 is out of range, the ***2.5V_error*** bit is set in ***Status***. In this case, neither $T_{int}$ nor $T_{ext}$ can be properly measured, and speed calculations cannot be performed. Therefore the ***temp_error***, ***rho_error***, and ***speed_error*** bits will also be set in ***Status***.

If the external temperature sensor is present, its value will be used in speed calculations regardless of the status of the internal temperature sensor. If the external temperature sensor is not present, the value of internal temperature sensor will be used. If either sensor is faulty, or the 2.5V reference is out of range, the faulty sensor reading will be reported as **0x8000**. The ***temp_error*** bit in ***Status*** is set only when neither temperature sensor is available (e.g., having a faulty internal temperature sensor with the external temperature sensor operating properly is not flagged with the ***temp_error*** bit). If the ***temp_error*** bit is set, neither air density nor airspeed can be calculated, and the ***rho_error*** and ***speed_error*** bits will also be set in ***Status***.

The ***EE_Needs_Update*** bit in ***Status*** is set whenever $P_o$ is adjusted (either explicitly or via an altitude update), $P_d$ is reset, the output data transmission protocol is set, the units used for output data transmission is set, or the output data update rate is updated. A successful EEPROM write will automatically clear this flag, and, at the next power-up, the SPA20422 will default to these stored values. If an EEPROM write is not performed while the ***EE_Needs_Update*** bit is set, the SPA20422 will continue to use the new values until power-down. Then, at the next power-up, the SPA20422 will revert to the values last stored in the EEPROM. Any attempt to write to the EEPROM without the ***EE_Needs_Update*** bit set will be ignored. The SPA20422 is originally shipped with the following system defaults set in the EEPROM:

Binary output data transmission protocol is set
SI units are set for data
Output data transmission rate is set for 10 intervals (.5 second)
$P_o$ is set to the "standard" value of 101.33 KPa

The EEPROM has a limited erase/write life of over two million cycles (more than adequate for the service life of the SPA20422). The ***EE_Status_1*** and ***EE_Status_0*** bits in ***Status*** are used to keep track of the approximate life left on the EEPROM, and indicate the remaining EEPROM life:

| | | |
|---|---|---|
| ***EE_Status_1*: *EE_Status_0*** | **0:0** | Over half of EEPROM life remaining |
| ***EE_Status_1*: *EE_Status_0*** | **0:1** | Less than half of EEPROM life remaining |
| ***EE_Status_1*: *EE_Status_0*** | **1:0** | Less than 10% of EEPROM life remaining |
| ***EE_Status_1*: *EE_Status_0*** | **1:1** | EEPROM exhausted – no more writing allowed |

Even when the EEPROM is exhausted, the board will still operate properly (i.e., output mode, units, and message rate can be changed, $P_o$ and altitude can be updated, and the differential pressure can be reset), but these changes cannot be updated in EEPROM to effect power-up defaults.

If the ***EE_Needs_Update*** bit is set in ***Status*** and an EEPROM write operation fails, the ***EE_Write_Error*** bit will be set in ***Status***, and, except in the case of an exhausted EEPROM (both ***EE_Status_1*** and ***EE_Status_0*** bits set), a new write operation must be attempted before removing power from the board to prevent the EEPROM data from being invalid at the next power-up. If the EEPROM data are invalid at power-up (e.g., an EEPROM write failed and power was removed before a successful write has been completed), both ***EE_Write_Error*** and ***EE_Needs_Update*** bits are set in ***Status***. With invalid data in the EEPROM at power-up, the SPA20422 reverts to the original set of factory defaults.

**GENERAL SPECIFICATIONS, SPA20422**

(August 2008)

*All specifications subject to change without notice*

| | Max | Typ | Min | |
|---|---|---|---|---|
| **Measurements** | | | | |
| Absolute Pressure | 115 | | 15 | KPa |
| | 33.96 | | 4.43 | in-Hg |
| Differential Pressure | 10 | | 0 | KPa |
| | 2.95 | | 0 | in-Hg |
| Internal Temperature | +85 | | -40 | °C |
| | +185 | | -40 | °F |
| External Temperature (1 microamp/° K from sensor) | +85 | | -55 | °C |
| | +185 | | -67 | °F |
| **Calculations** | | | | |
| Altitude, calculated (not guaranteed) | 13750 | | -1100 | meters |
| | 45000 | | -3600 | feet |
| Altitude, effective | 7500 | | -500 | meters |
| | 24600 | | -1640 | feet |
| Dry Air Density (typical range) | 1.80 | | .17 | kg/cu-meter |
| | .11 | | .011 | lb/cu-ft |
| Airspeed (sea level, typical range) | 465 | | 0 | KPH |
| | 250 | | 0 | knots |
| Airspeed (5000 meters, typical range) | 612 | | 0 | KPH |
| | 330 | | 0 | knots |
| **Accuracies** | | | | |
| Absolute Pressure (0 to +85° C) | ±1.5 | | | KPa (±1.5%) |
| Absolute Pressure (-40 to 0° C) | ±4.5 | | | KPa (±4.5%) |
| Differential Pressure (0 to +85° C) | ±.5 | | | KPa (±5%) |
| Differential Pressure (-40 to 0° C) | ±1.5 | | | KPa (±15%) |
| Internal Temperature | ±2 | | | °C |
| Altitude (sea level, uncompensated) | ±125 | | | meters |
| Altitude (sea level, with updates) | ±2 | | | meters |
| Airspeed (sea level) | ±3 | | | % |

| | Max | Typ | Min | |
|---|---|---|---|---|
| **Power** | | | | |
| Input Voltage | 32 | | 6 | V |
| Dissipation | 200 | 150 | | mW |
| **RS-422 Signals, 38400 Baud (8, n ,1)** | | | | |
| Differential Driver Output (No Termination) | | 3.3 | | V |
| Differential Driver Output (120Ω Termination) | | | 2.0 | V |
| Receiver Input Voltage | 12.0 | | -7.0 | V |
| Receiver Differential Threshold | 200 | | -200 | mV |
| Receiver Input Hysteresis | | 50 | | mV |
| Input Resistance | | | 12 | KΩ |
| **TTL-Level Conversions, 38400 Baud (8, n, 1)** | | | | |
| ***VR*** Out (Connected to ***RA***) | | 1.5 | | V |
| $V_{IH}$ (***RB***) | 5.5 | | 2.0 | V |
| $V_{IL}$ (***RB***) | .8 | | 0 | V |
| $V_{OH}$ (***TB***) | 3.3 | | 2.0 | V |
| $V_{OL}$ (***TB***) | .4 | | 0 | V |
| $I_{OH}$ (***TB***) | 8 | | | ma |
| $I_{OL}$ (***TB***) | -8 | | | ma |

**Physical**

| | |
|---|---|
| Size | 1.05” [26.7 mm] W x 1.15” [29.2 mm] L x .75” [19.1 mm] T |
| Weight | 9 g [.32 oz] |

**Environment**

| | |
|---|---|
| Operating Temperature | -40 to +85 °C |
| Storage Temperature | -55 to +125 °C |
| Vibration | 6 $g_{rms}$ |
| Shock | 100 g, 8 ms, ½ sine |

# SPA20422 INPUT COMANDS AND OUTPUT DATA

## ASCII Protocol Input Commands

| | |
|---|---|
| ***~a*** $^{C}_{R}{}^{L}_{F}$ | Set ASCII output protocol |
| ***~b*** $^{C}_{R}{}^{L}_{F}$ | Set binary output protocol |
| ***~e*** $^{C}_{R}{}^{L}_{F}$ | Write to EEPROM |
| ***~h*** [*value*] $^{C}_{R}{}^{L}_{F}$ | Update altitude, required *value* in meters x 1 or feet x 1 |
| ***~m*** [*value*] $^{C}_{R}{}^{L}_{F}$ | Poll data values, optional *value* sets output message interval in units of 50 ms or off if zero |
| ***~r*** [*value*] $^{C}_{R}{}^{L}_{F}$ | Update effective sea-level pressure ($P_o$), required *value* in KPa x 100 or in-Hg x 100 |
| ***~s*** $^{C}_{R}{}^{L}_{F}$ | Set SI (metric) units |
| ***~u*** $^{C}_{R}{}^{L}_{F}$ | Set US units |
| ***~v*** $^{C}_{R}{}^{L}_{F}$ | Reset differential pressure ($P_d$) |

## Binary Protocol Input Commands

| | | |
|---|---|---|
| ***Poll_Data__Set_Interval*** | [**0x81**] [**0xA1**] [**0x01**] [**0x00** *or* **0x01**] [*Optional value*] [*CS0*] [*CS1*] | *Value* interval in 50 ms, 0 off |
| ***Reset_*** $P_d$ | [**0x81**] [**0xA1**] [**0x03**] [**0x01**] [**0x00**] [**0x26**] [**0x14**] | |
| ***Update_*** $P_o$ | [**0x81**] [**0xA1**] [**0x03**] [**0x03**] [**0x01**] [*2-byte value*] [*CS0*] [*CS1*] | *Value* KPa x 100, in-Hg x 100 |
| ***Update_Altitude*** | [**0x81**] [**0xA1**] [**0x03**] [**0x05**] [**0x02**] [*4-byte value*] [*CS0*] [*CS1*] | *Value* meters x 100, feet x 100 |
| ***Write _to_EEPROM*** | [**0x81**] [**0xA1**] [**0x03**] [**0x01**] [**0x07**] [**0x2D**] [**0x1B**] | |

## ASCII Output Data (in sequence)

| | | | | |
|---|---|---|---|---|
| $P_a$ | Absolute Pressure | KPa x 100 | in-Hg x 100 | |
| $P_o$ | Effective Sea-Level Pressure | KPa x 100 | in-Hg x 100 | |
| ***H*** | Altitude | meters x 10 | feet x 10 | |
| $T_{int}$ | Internal Temperature | °C x 10 | °F x 10 | |
| $T_{ext}$ | External Temperature | °C x 10 | °C x 10 | -32768 = (Sensor not present) |
| ***rho*** | Dry Air Density | kg/cu-meter x 1000 | lb/cu-ft x 100 | |
| $P_d$ | Differential Pressure | KPa x 1000 | in_hg x 1000 | |
| ***V*** | Airspeed | KPH x10 | Knots x 10 | |
| ***Status*** | System Status | 16-bit bit field – Value dependent upon system status | | |
| ***UTime*** | Free-Running Timer | Unsigned 16-bit value – Number of 50 ms periods since reset | | |

## Binary Output Data (in sequence)

| | | | |
|---|---|---|---|
| ***Status*** | 2 bytes, 16-bit bit field | System Status | Value dependent upon system status |
| ***UTime*** | 2 bytes, 16-bit unsigned integer | Running Timer | Recirculating count of 50 ms periods since reset |
| $P_a$ | 2 bytes, 16-bit unsigned integer | Absolute Pressure | KPa x100 or in-Hg x 100 |
| $P_o$ | 2 bytes, 16-bit unsigned integer | Effective Sea-Level Pressure | KPa x100 or in-Hg x 100 |
| ***H*** | 4 bytes, 32-bit signed integer | Altitude | meters x10 or feet x 10 |
| $T_{int}$ | 2 bytes, 16-bit signed integer | Internal Temperature | °C x10 or °F x10 |
| $T_{ext}$ | 2 bytes, 16-bit signed integer | External Temperature | °C x10 or °F x10, **0x8000** = Sensor not present |
| ***rho*** | 2 bytes, 16-bit unsigned integer | Dry Air Density | kg/cu-meter x1000 or lb/cu-ft x 1000 |
| $P_d$ | 2 bytes, 16-bit signed integer | Differential Pressure | KPa x1000 or in-Hg x 1000 |
| ***V*** | 2 bytes, 16-bit unsigned integer | Airspeed | KPH x10 or knots x 10 |

## CONVERSION FACTORS

| Multiply | By | To Get |
|---|---|---|
| KPa | .2953 | in-Hg |
| in-Hg | 3.3867 | KPa |
| meters | 3.2808 | feet |
| feet | .3048 | meters |
| kg/cu-meter | .061785 | lb/cu-ft |
| lb/cu-ft | 16.185 | kg/cu-ft |
| KPH | .5409 | knots |
| knots | 1.8486 | KPH |
| knots | .8696 | MPH |
| MPH | 1.15 | knots |

$$°C \times (9/5) + 32 = °F$$
$$(°F - 32) \times (5/9) = °C$$
$$°C + 273.15 = °K$$

Microbotics, Inc.
28 Research Drive, Suite G
Hampton, Virginia 23666
Tel 757-865-7728
Fax 757-865-7565
www.microboticsinc.com

**WARRANTY; DISCLAIMER AND LIMITATION OF REMEDIES:** MICROBOTICS WARRANTS ITS PRODUCTS AGAINST DEFECTS IN MATERIALS AND WORKMANSHIP UNDER NORMAL USAGE FOR A PERIOD OF ONE YEAR AFTER DELIVERY TO THE END USER. MICROBOTICS MAKES NO ADDITIONAL WARRANTIES WHATSOEVER WITH RESPECT TO ITS PRODUCTS, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PURPOSE, OR AGAINST INFRINGEMENTS OF PATENT OR OTHER RIGHTS OF THIRD PARTIES IN THE PRODUCTS.

Customer will provide Microbotics the right to inspect any Products which Customer claims do not conform to the Microbotics Warranty. THE EXCLUSIVE REMEDIES FOR NONCOMPLIANCE WITH ANY MICROBOTICS WARRANTY SHALL BE, IN MICROBOTICS' SOLE AND ABSOLUTE DISCRETION, EITHER REPAIR OF THE NONCONFORMING PRODUCTS, REPLACEMENT OF THE NONCONFORMING PRODUCTS, OR RETURN OF SAME FOR REPAYMENT OF THE PURCHASE PRICE.

**MISUSE:** Microbotics shall have no liability or obligation to Customer with respect to any of the Products which have been subject to abuse, misuse, improper use, negligence, accident, modification, alteration, tampering, any use of the Products outside their normal environment, or any alteration of any literature with respect to the Products.

**LIMITATION OF DAMAGES RECOVERABLE BY CUSTOMER:** WITHOUT LIMITATION OF ANY OTHER PROVISION IN THESE TERMS AND CONDITIONS LIMITING OR EXCLUDING LIABILITY OF MICROBOTICS, THE EXCLUSIVE DAMAGES RECOVERABLE BY CUSTOMER FOR ANY CLAIM OF ANY KIND WHATSOEVER ARISING FROM OR IN ANY WAY CONNECTED TO THE PRODUCTS, REGARDLESS OF THE LEGAL THEORY, SHALL NOT BE GREATER THAN THE ACTUAL PURCHASE PRICE OF THE PRODUCTS PAID BY CUSTOMER WITH RESPECT TO WHICH SUCH CLAIM IS MADE, AND IN NO EVENT SHALL MICROBOTICS BE LIABLE FOR ANY SPECIAL, INDIRECT, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY KIND, INCLUDING WITHOUT LIMITATION ANY DAMAGES WITH RESPECT TO LOSS OF INCOME, COMPENSATION OR PROSPECTIVE PROFITS, ANY EXPENDITURES, INVESTMENTS OR COMMITMENTS OF CUSTOMER, ANY LOSS WITH RESPECT TO THE ESTABLISHMENT, DEVELOPMENT OR MAINTENANCE OF BUSINESS REPUTATION OR GOOD WILL, OR ANY LOSS INCURRED IN OBTAINING SUBSTITUTE PRODUCTS OR SERVICES, OR ARISING FROM THE CLAIMS OF THIRD PARTIES, INCLUDING CUSTOMERS.

**FORCE MAJEURE:** Without limitation of any clause herein limiting or exculpating Microbotics from liability, Microbotics shall not be responsible or liable for any failure to perform, or any delay in supplying if occasioned in whole or in part by act of God or the public enemy, fire, explosion, perils of the sea, flood, drought, war, riots, civil insurrection, sabotage, accident, embargo, governmental priority, requisition or allocation or any action of any governmental authority (or any refusal of such governmental authority to provide necessary authorization), or shortage or failure of supply, materials, fuel, transportation or labor, or strikes or other labor trouble, or any occurrence, act, cause or thing beyond the reasonable control of Microbotics, all of which shall excuse any failure or delay on the part of Microbotics, and Microbotics shall have no obligation or liability whatsoever arising out of or in connection with any such failure or delay.

**POLICY ON RETURNS:** In the event of any shortage, damage, defect or discrepancy in or to a shipment of Products, Customer shall promptly report the same to Microbotics and furnish such written evidence or other documentation as Microbotics may reasonably require. Except as specifically authorized in the Warranty provisions above, Microbotics shall not be liable for any such shortage, damage, defect or discrepancy unless Microbotics has received notice and substantiating evidence thereof from Customer within five (5) days of arrival of the Products. If the substantiating evidence delivered by Customer demonstrates to Microbotics' reasonable satisfaction that Microbotics is responsible for such shortage, damage, defect or discrepancy, Microbotics shall deliver additional or substitute Products to Customer in accordance with the delivery terms otherwise set forth herein; provided, however, that in no event shall Microbotics be liable for any additional costs, expenses or damages incurred by Customer, directly or indirectly, a result of such shortage, damage, defect or discrepancy in or to a shipment.

All returns of Products by Customer, for whatever reason, shall be made, at Customer's sole expense, DDP Microbotics' headquarters office in Hampton, Virginia, USA (INCOTERMS 2000). All sales to Customer are final, and Microbotics will not accept any returns of Products unless and until prior written authorization is received by Customer from Microbotics. Microbotics will not accept returns based upon delays in shipment by Microbotics or receipt by Customer or market conditions (including changes in demand for or popularity of any of the Products).