# Protocol for the JPL Board

## Using the board

### Theory of operation

The unit consists of 3 boards: a sensor board, which contains mountings for sensor cartridges, an analog board that does the signal recovery, and a digital board for communicating with a computer (or some other device) via RS-232, using the protocol explained at the end of this document.

The analog and digital boards come from JPL, and are an old design they developed – our newer designs will be much simpler and smaller. As a result, there are certain legacy strangenesses.

The sensor board contains:

32 *sensor elements* - 4 metal oxide sensor elements and 28 chemiresistive sensor elements (note: we call the collection of 32 a *sensor* and each individual one a *sensor element*)

a pump

heaters for the metal oxide sensors

a thermistor, maybe two

Each sensor element (metal oxides and chemiresistive) acts as a variable resistor. Under normal operations, the resistance drifts slowly. When an analyte gas is applied, the resistance changes, in such a way that delta R over R is (approximately) constant for a given analyte at a given concentration.

### Channels & Groups

The board consists of 4 *channels* A, B, C, D. Each channel contains replicated circuitry. If you look at the analog board (the bottom board), you can see the A, B, C, D labels. Each channel consists of 8 sensors. The way the circuitry is designed, you can typically take measurements from all four channels simultaneously for a given *group* of sensors. The groups are numbered 0-7, so you might, for example, take simultaneous measurements of A3, B3, C3, D3.

### Mapping

We typically report the sensor elements in this order:

```
C7
C5
C3
C1
C6
C4
C2
C0
D1
D3
D5
D7
D0
```

```
D2
D4
D6
B7
B5
B3
B1
B6
B4
B2
B0
A1
A3
A5
A7
A0
A2
A4
A6
```

That corresponds to a reasonable order when viewing the sensor board from the top. (Refer to sensor numbering PowerPoint document for more details.)
Note: this is part of the legacy strangeness alluded to above.

V0, V1, V3

The boards measure the resistances to approximately 24 bit of accuracy. To do that, they:

1) Apply a voltage V0 across a given sensor element in a resistor divider. This results in a voltage proportional to the resistance.
2) Subtract an offset voltage V1 from the resulting voltage.
3) Amplify the difference, so that it ranges from 0 to 5 V, and digitize that voltage difference. The result is called V3, for historical reasons.

V0 and V1 are specified as 12-bit numbers, and V3 is a 12-bit value. The complex 3-step circuitry is designed to use these values to get >> 12 bits of accuracy.

The V0 and V1 values need to be set correctly so that the V3 measurement is in the dynamic range of the circuitry.

You will probably never need to set V0 or V1 by hand (although that is a useful thing to do when trying to figure out if a board is faulty). Instead, use the *find* or *baby find* command to calibrate V0 and V1 to optimal values.

Find

The *find* command (see command reference) finds optimal V0 and V1 values for all groups and all channels. Note that it stores those values on the board; it doesn't return them. (To return the values, use the *ram dump* command.) The *find* command takes something like 4 seconds.

In other words, it calibrates the board based on the present values of all resistances.

The *baby find* command (see command reference) finds V0 and V1 values for a subset of the sensor elements, specifically for exactly one given group, and from one to 4 channels. Note that, like *find*, it stores the values on the board, it doesn't return them. This command takes something like ½ second per group. The number of channels per group doesn't affect this time significantly.

The *ram dump* command returns V0 and V1 values from the board.

The *measure* command measures and returns V3 values from the board. It takes ½ a second (is this true?).

Recalibration

When the V3 value for a given sensor element gets near the edge of its dynamic range, you should recalibrate using either *find* or *baby find*. Our software uses baby-find, although this requires a little more record keeping to figure out which groups and which channels you need to run *baby find* on.

Specifically, if V3 < 0x200 (512 decimal) or V3 > 0xE00 (3584 decimal), you should recalibrate that sensor element.

Be a little careful if you do this and the sensor remapping mentioned above; be sure that you're recalibrating the right sensor, not the remapped one.

Resistances

To get resistance measurements, you need to do the following:

1) Issue a *ram dump* command to get V0 and V1 values.
2) Issue a *measure* command to get V3 values.
3) Convert all textual hex values to actual integers.
4) Use the following formula:

```
v0 = <Hex V0> * .0005
v1 = <Hex V1> * .001
v3 = <Hex V3> * .001
G = 261
R0 = 10000
r = (((v3 + v1)/G + v1) - v0) / (v0/R0)
```

Note: if r is negative, it means one of two things:

1) There's a typo in your formula (e.g., you forgot to multiply by 0.0005 or 0.001.
2) There's a problem with the board. All the existing boards 1-10 have been checked and are error-free, but new boards might have a problem. (Specifically, we saw a problem where boards were not producing the correct V1 voltages, i.e., the DACs were not working correctly.)

If you're writing new software, we have a cartridge or two that have resistors of known values instead of sensors – these can be very useful making sure the protocol is working correctly and the calculations are done correctly.

Pump
The pump on the board is turned on or off using the *pump* command.

Heaters
The metal oxide sensors require heating to work properly. We have tied what used to be the valve on the JPL board to the heaters. So to turn the heaters on or off, you use the *valve* command.

Our system has no valve, only heaters.

Once on, you should let them stay on for 1 minute before running *find* (or *baby find*) on the metal oxide sensors and taking their measurements. This gives them sufficient time to warm up the metal oxides and stabilize them.

Thermistors
The JPL board was originally designed to have 4 independent thermistors. Since we added the sensor board, we typically use one or two thermistors.

The thermistors are read as part of the *pic dump* command. The one on the sensor board is #3 (indexed 0, 1, 2, 3), and the one (if it exists) on the analog board is #2. The value is between 0 and 255, and we probably have a conversion to degrees somewhere. We've seen thermistors die on units that have stayed in the field, though. We make little use of them.

The thermistor on the sensor board is permanently mounted under the metal oxide sensors. The one on the analog board is connected at the front of the unit.

Startup
On startup, the board sends the following on the RS-232 port:

```
\r\n
F\r\n
OK\r\n
\r\n
T\r\n
00-00-00 00:00:00\r\n
\r\n
```

Note that this occurs some time after power on (maybe a minute? Maybe less.). The easiest thing to do is to not start sending/receiving commands until that has occurred. There's some sort of odd freeze up if you send commands too soon after power on – we've never been able to replicate and diagnose it.

Note that this means the board has run a *find* command.

Typical cycle
The typical cycle is this:

1) Send a *pic dump* command as a quick 'is the board alive?'

2) Turn the pump on using the *pump* command.
3) Turn the heaters on using the *valve* command.
4) Let the heaters heat up for 1 minute.
5) Send a *find* command.
6) Loop
    a. Recalibrate sensors if necessary (see e) using *baby find* command.
    b. Send a *ram dump* command to get V0s and V1s.
    c. Send a *measure* command to get V3s.
    d. Calculate resistances from V0s, V1, and V3s.
    e. Figure out which V3s are out of range, mark those as 'recalibrate next time around.'
7) Turn off the heaters using the *valve* command.
8) Turn off the pump using the *pump* command.

Note that we've seen an error where the system reboots at step 2 (which may be incorrectly diagnosed as a problem in step 3, because the command in step 3 fails due to the reboot). This is due to insufficient current to drive the pump, which browns out the system.

**Protocol Basics**

19,200 baud
8 bits
No parity
1 stop bit

In general, you send a command of the form:

```
x arg1 arg2
```

i.e., lowercase command, arguments separated by spaces (or any other delimiter, it doesn't actually care). **Note: 'x' is not a valid command, rather it's just a fill in for the actual command.**

The board responds with

```
xX arg1 arg2<LF><CR>
```

i.e., echo of command, capitalized command, echo of space and arguments, line feed, carriage return.

Sometimes, the board also responds with

```
OK<LF><CR>
<LF><CR>
```

A subsequent section will show all of the commands.

From HyperTerminal or another such program, you can type the commands and get reasonable responses. When you go to automate the commands, there is a timing issue. Basically, you need to be in lockstep with the command. So, for example, suppose you're sending the command x 01 ab. It would go like this:

| *You send* | *System responds* |
|---|---|
| x | xX |
| <space> | <space> |
| 0 | 0 |
| 1 | 1 |
| <space> | <space> |
| a | a |
| b | b<LF><CR><br>OK<LF><CR><br><LF><CR> |

In particular, you don't want to get ahead of the system. The PIC has a 2 character buffer, then it starts losing things. Your best bet is to send characters one at a time and wait for the appropriate response before going on.

## Command reference

| Command | Response | Description |
|---|---|---|
| `p <0 or 1>` | `pP <0 or 1>\r\n`<br>`OK\r\n`<br>`\r\n` | 'Pump' command - turns the pump on or off. |
| `v <0 or 1>` | `vV <0 or 1>\r\n`<br>`OK\r\n`<br>`\r\n` | 'Valve' command - turns the inlet valve on or off.<br>**Note: we have the 'valve' connected to the heaters for the metal oxide sensors, so this is the command to send to turn the heaters on.** |
| `f` | `fF\r\n`<br>`OK\r\n`<br>`\r\n` | 'Find' command - causes the board to find optimal V0 and V1 values for all sensors.<br>Note: this command takes several seconds to complete, so be careful with timeout values. |
| `b <0 to 7><0 to F>`<br><br>where<br><0 to 7> is the group number<br><0 to F> is a binary combination of channels<br>0 = binary 0000 -> channels a,b,c,d respectively (all off)<br>F = binary 1111 -> channels a,b,c,d respectively (all on) | `bB <0 to 7><0 to F>\r\n`<br>`OK\r\n`<br>`\r\n` | 'Baby find' command - finds optimal V0 and V1 values for specific sensors.<br><br>Note that the time to do a baby find for all channels is only slightly more than the time to do it for one channel - the time is dominated by the "wait for the V0/V1/V3 values to stabilize" time, which runs in parallel for all channels. |
| `m` | `mM \r\n`<br>`XXX XXX XXX XXX \r\n`<br>`XXX XXX XXX XXX \r\n`<br>`XXX XXX XXX XXX \r\n`<br>`XXX XXX XXX XXX \r\n`<br>`XXX XXX XXX XXX \r\n` | 'Measure' command - returns 12-bit V3 values for all 32 sensors.<br>Note: each line XXX XXX XXX XXX \r\n is one group. Order is group 0, group 1, ... , group 7. Within |

| | | |
|---|---|---|
| | XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>\r\n | a line, order is channel A, B, C, D. |
| r | rR\r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>XXX XXX XXX XXX \r\n<br>\r\n | 'RAM dump' command. Returns:<br>V0 table values and<br>V1 table values.<br><br>Note: return order is:<br>All V0 values<br>All V1 values<br>where "All V0 values" are the first 8 rows, and consist of group 0, group 1, ... group 7 see measure command for more info on encoding. |
| i | iI\r\n<br>XX XX XX XX XX XX XX XX<br>XX XX XX XX YY\r\n<br>OK\r\n<br>\r\n | 'pIc adc' command - returns values of all PIC Analog to Digital Converters.<br><br>first 4 XXs are thermistor values for channels a,b,c,d<br>second 4 XXs are ???<br>last 4 XXs are heater values for channels a,b,c,d<br>YY is a binary number something like 0001 00VP<br>where first 4 are serial number of the board (1 for current version of software)<br>V is the valve state<br>P is the pump state<br><br>Note: we use this command in various places to confirm we can talk to the board, as it is relatively fast and has no (known) side effects. |
| | | |
| **Unused commands** | | |
| h XX XX XX XX | hH XX XX XX XX\r\n<br>OK\r\n<br>\r\n | Set values for 4 onboard heaters.<br><br>Note: not 100% sure what the values mean. I think 0 is 0V and 255 is 9V, but I could be wrong.<br><br>**Note: we don't have these heaters hooked up. You should use the 'v' command to turn on** |

| | | |
|---|---|---|
| | | **the heaters we do have hooked up.** |
| t ???? | ???? | Gets time from the real time clock.<br>**Note: we've never hooked up a real time clock.** |
| s ???? | ???? | Sets the time of the real time clock.<br>**Note: we've never hooked up a real time clock.** |
| | | |
| **Debugging commands** | | |
| g <0 to 7> | gG <0 to 7>\r\n<br>OK\r\n<br>\r\n | 'Group' command - sets the current group number. |
| q | qQ  G<group #>\r\n<br>XXX XXX XXX XXX \r\n<br><br>Note: double space between Q and G. | 'Query' command - queries the current group, returns 12-bit V3 values for all 4 channels.<br><br>You must call setGroup first to tell it which group to use. |
| d <a,b,c,d> XXX XXX | dD <a,b,c,d> XXX XXX\r\n<br>OK\r\n<br>\r\n | 'DAC' command - sets the values of the digital to analog converters for the current group and the given channel (i.e., sets V0 and V1).<br><br>Note: this was really useful in diagnosing the problem with bad DAC converter chips. |
| n | nN  G<group number>\r\n<br>NO: XXX XXX XXX XXX<br>\r\n<br>V0: XXX XXX XXX XXX<br>\r\n<br>V1: XXX XXX XXX XXX<br>\r\n<br>\r\n | 'Nostril dump' command - returns the values of:<br>the nostril table (sensor values of current group)<br>V0 and V1 values for current group.<br><br>Note: you need to set group before calling this function.<br><br>It's unclear to me what NO is (I mean, obviously, it's the nostril table). I think it means "The most recent value of V3 read." In other words, this command should not take a new reading, but rather, report the value of the most recent reading. I could be very wrong, though. |

## Electrical Characteristics

With pump and heaters on, the board draws about 200mA. Without those, it's probably 50mA.

It needs an input voltage of 12 V.  You could probably get away with 11 or maybe even 10.5, but not much lower.  I forget the upper limit.