BasicAirData – Air Data Computer

Communication Protocol - Common Message Set

Attention, this is a work in progress version

Draft Version 0.5 – 6th Feb 2018
Draft Version 0.4 – 6th Jan 2018
Draft Version 0.3 – 11th May 2017
Draft Version 0.2 – 26th Apr 2017
Draft Version 0.1 – 4th Jan 2017

Legend for the table colors:
- Green = implemented
- Red = not implemented
- Blue = work in progress

ID	Field	Field Name	Description
0	HBQ	HEARTBEAT_REQ	Check if the system is present and responding
1	HBA	HEARTBEAT_ASSERT	Answer about the system presence
2	TMS	TIME_SET	Set date and time of the ADC
3	TMQ	TIME_REQ	Ask for date and time
4	TMA	TIME_ASSERT	The Date and time
5	STS	STATUS_SET	Set the status
6	STQ	STATUS_REQ	Ask for ADC configuration status
7	STA	STATUS_ASSERT	The Status
8	DTS	DATA_SET	Set the data
9	DTQ	DATA_REQ	Ask for the ADC data
10	DTA	DATA_ASSERT	The data
11	LCS	LOG_CURRENTFILE_SET	Set the current log file name
12	LCQ	LOG_CURRENTFILE_REQ	Ask for the current log file name
13	LCA	LOG_CURRENTFILE_ASSERT	The current log file
14	DFS	DATA_FREQ_SET	Set the ADC to send Data at specified frequencies
15	DFQ	DATA_FREQ_REQ	Ask for the ADC Data Frequencies
16	DFA	DATA_FREQ_ASSERT	The Data Frequencies
17	FMQ	FILE_MANAGER_REQ	Manage SD Files
18	FMA	FILE_MANAGER_ASSERT	File manager answers
19	CCS	CALIBRATION_CON_SET	Setup Sensor Calibration
20	CCA	CALIBRATION_CON_ASSERT	General calibration ack and parameter from ADC

Every message string will begin with a “$” character and will end with a “\n” (newline).

0 – HBQ – HEARTBEAT_REQ

The heartbeat message ask if a system is present and responding.
The remote system will respond with a HEARTBEAT_ASSERT (#1).

0	Field String	$HBQ
1	description	The description of the requesting device
2	firmware_version	The version of the communication protocol

Example:
`$HBQ,StatusVisualizer,1`

1 – HBA – HEARTBEAT_ASSERT

The heartbeat message answer if a system is present and responding.
This message is typically the answer of a HEARTBEAT_REQ (#0).

0	Field String	$HBA
1	description	The description of the responding device
2	firmware_version	The version of the communication protocol

Example:
`$HBA,ASGARD,0.5`

2 – TMS – TIME_SET

The request to set the remote datetime.
The remote system will respond with a TIME_ASSERT (#4).

0	Field String	$TMS
1	Time as seconds since Jan 1, 1970	Long Integer, seconds

Example:
`$TMS,1493151334`

3 – TMQ – TIME_REQ

Ask for the remote datetime.
The remote system will respond with a TIME_ASSERT (#4).

0	Field String	$TMQ

Example:
`$TMQ`

4 – TMA – TIME_ASSERT

The message contains the current datetime of the device.

0	Field String	$TMA
1	Time as seconds since Jan 1, 1970	Long Integer, seconds

Example:
`$TMA,1493151334`

5 – STS – STATUS_SET

The request to set the status of the remote device.
It is possible to set some parameters and leave untouched some others, using the value “-1”.
The remote system will respond with a STATUS_ASSERT (#7).

0	Field String	$STS
1	SD card present	1 Present. 0 Not present. E=error code
2	Deltap sensor	1 Present. 0 Not present. E=error code
3	Absolute pressure sensor	1 Present. 0 Not present. E=error code
4	External Temperaure sensor	1 Present. 0 Not present. E=error code
5	Deltap sensor temperature	1 Present. 0 Not present. E=error code
6	Absolute pressure sensor temperature	1 Present. 0 Not present. E=error code
7	Real time clock battery	1 Present. 0 Not present.
8	Error/Warning	That should be a code handled directly by firmware. Exception handling is not implemented. “SDLOW” This is a warning
9	BT Present	1 Present. 0 Not present. E=error code (BT module hooked to Serial1)

Example:
`$STS,1,1,-1,1,1,1,1,N,1`

6 – STQ – STATUS_REQ

Ask for the remote status.
The remote system will respond with a STATUS_ASSERT (#7).

0	Field String	$STQ

Example:
`$STQ`

7 – STA – STATUS_ASSERT

The message contains the current status of the device.

0	Field String	$STA
1	SD card present	1 Present. 0 Not present. E=error code. N not set.
2	Deltap sensor	1 Present. 0 Not present. E=error code. N not set.
3	Absolute pressure sensor	1 Present. 0 Not present. E=error code. N not set.
4	External Temperaure sensor	1 Present. 0 Not present. E=error code. N not set.
5	Deltap sensor temperature	1 Present. 0 Not present. E=error code. N not set.
6	Absolute pressure sensor temperature	1 Present. 0 Not present. E=error code
7	Real time clock battery	1 Present. 0 Not present.
8	Error/Warning	That should be a code handled directly by firmware. Exception handling is not implemented. “SDLOW” This is a warning
9	BT Present	1 Present. 0 Not present. E=error code (BT module hooked to Serial1)

Example:

```
$STA,1,1,1,1,1,1,1,,0
All the sensors, SD card, real time clock battery are present and
operational, no errors are reported.

$STA,1,1,0,1,1,1,0,SDLOW,1

All the sensors except external temperature, and SD card are present. SD
card space shortage is reported.
```

8 – DTS – DATA_SET

The request to set (force) the data of the remote device.
It is possible to set some parameters and leave untouched some others, using the value “-1”.
The remote system will respond with a DATA_ASSERT (#10). [Note. How to decide what #10 message should include? The reply #10 will contain no data at all “$DTA”. It’ a simple acknowledge]

i	Field String	$DTA
1	Timestamp see msg #4	Time as seconds since Jan 1, 1970
2	Deltap [Sensor units, counts]	integer. Sensor dependent.
3	Absolute Pressure [Sensor units, counts]	integer. Sensor dependent
4	Ext Temperature [Sensor units, counts]	integer. Sensor dependent
5	Temp deltap [Sensor units, counts]	integer. Sensor dependent
6	Temp absolute [Sensor units, counts]	integer. Sensor dependent
7	Deltap [Pa]	float
8	Absolute Pressure [Pa]	float
9	Ext Temperature [K]	float
10	Temp deltap [K]	float
11	Temp absolute [K]	float
12	IAS [m/s]	float
13	TAS [m/s]	float
14	Altitude [m]	float
15	OAT [K]	float
16	Relative time micro millis [s*10^-6]	integer
17	Uncertainty IAS [m/s]	float
18	Uncertainty TAS [m/s]	float
19	Uncertainty Altitude [m]	float
20	Uncertainty OAT [K]	float
21	Air Density [kg/m^3]	float
22	Air Viscosity [Pa*s]	float
23	Re	float
24	c factor	float

Example:

```
Set the value of ext temperature
$DTS,,,,,,,,350,,,,,,,,,,,,,,,
```

9 – DTQ – DATA_REQ

Ask for the remote data. Selects the required data with the fields value. If a 0 is sent at the position i then the ADC will send a #10 reply with the field i not valued. If a 1 is sent at the position i then the ADC will send a #10 reply with the field i valued.
Configuration data will be saved to the SD card (CONFIG.CFG file),

Use the "`$DTQ`" command, without parameters, and the remote system will respond with a DATA_ASSERT (#10) instead.

i	Field String	$DTQ
1-24	Select	0 if data field not required. 1 if data field required

Example:

```
Request all the data fields of #10
$DTQ,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
Request all the data fields except Deltap and Ext Temperature
$DTQ,1,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
Request time, deltap, p, ext temperature
$DTQ,1,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
Simply request a string of data
$DTQ
```

10 – DTA – DATA_ASSERT

Data lengths shown in the table below are variable and conservative. The ADC will respond to a #9 message with a DTA – DATA_ASSERT that contains only the data fields that have been marked with 1 within the #9 message.

i	Field String	$DTA
1	Timestamp see msg #4	Time as seconds since Jan 1, 1970
2	Deltap [Sensor units, counts]	6 Chars, integer. Sensor dependent.
3	Absolute Pressure [Sensor units, counts]	6 Chars, integer. Sensor dependent
4	Ext Temperature [Sensor units, counts]	5 Chars, integer. Sensor dependent
5	Temp deltap [Sensor units, counts]	5 Chars, integer. Sensor dependent
6	Temp absolute [Sensor units, counts]	5 Chars, integer. Sensor dependent
7	Deltap [Pa]	8 Chars, 2 decimal digit
8	Absolute Pressure [Pa]	8 Chars, 1 decimal digit
9	Ext Temperature [K]	5 Chars, 1 decimal digit
10	Temp deltap [K]	5 Chars, 1 decimal digit
11	Temp absolute [K]	5 Chars, 1 decimal digit
12	IAS [m/s]	6 Chars, 2 decimal digit
13	TAS [m/s]	6 Chars, 2 decimal digit
14	Altitude [m]	7 Chars, 2 decimal digit
15	OAT [K]	5 Chars, 1 decimal digit
16	Relative time micro millis [s*10^-6]	5 Chars, integer
17	Uncertainty IAS [m/s]	3 Chars, 1 decimal digit
18	Uncertainty TAS [m/s]	3 Chars, 1 decimal digit

19	Uncertainty Altitude [m]	3 Chars, 1 decimal digit
20	Uncertainty OAT [K]	3 Chars, 1 decimal digit
21	Air Density [kg/m^3]	8 Chars, 6 decimal digit
22	Air Viscosity [Pa*s]	10 Chars, 8 decimal digit
23	Re	8 Chars, 1 decimal digit
24	c factor	6 Chars, 4 decimal digit

Example:

```
$DTA,1493151334,6608,******,******,*****,*****,*****,472.60,100926.1,15.0,18
.3,18.6,27.77,27.77,63.1,15.0,1244,0.4,0.7,1.1,0.3,1.225000,18.396057,15081.
1,0.9977
```

11 – LCS – LOG_CURRENTFILE_SET

This message request to set the current active log file name. Reply is #13 msg.
If the file doesn't exist, a new file is created.

0	Field String	$LCS
1	Name	The requested log file name. Use the "?" character to create a new one using a progressive number

Example: (set the file to DATALOG.CSV)
`$LCS,DATALOG.CSV`
Example: (set the file to a new file. a progressive name, starting from LOG_0000.CSV, will be assigned)
`$LCS,LOG_?.CSV`

12 – LCQ – LOG_CURRENTFILE_REQ

Ask for the log file name. Reply is the active file name message #13

0	Field String	$LCQ

Example:
`$LCQ`

13 – LCA – LOG_CURRENTFILE_ASSERT

The message contains the current active log file.

0	Field String	$LCA
1	frequency	The current active log file name

Example:
`$LCA,DATALOG.CSV`

14 – DFS – DATA_FREQ_SET

The request to set the frequency rate of the remote data messages.
It is possible to set some parameters and leave untouched some others, using the value “-1”.
The remote system will respond with a DATA_FREQ_ASSERT (#16).

0	Field String	$DFS
1	COM frequency	The requested frequency rate (messages per second) to Serial interface
2	BT frequency	The requested frequency rate (messages per second) to Bluetooth
3	SD frequency	The requested frequency rate (messages per second) to SD Card

Example: (set the serial frequency to 10 Hz, Bluetooth frequency to 0.5 Hz, and SD frequency to 30 Hz)
`$DFS,10,0.5,30`
Example: (set the Bluetooth frequency to 2 Hz. SD and Serial frequency won’t be changed)
`$DFS,-1,2,-1`

15 – DFQ – DATA_FREQ_REQ

Query the frequency rate of the remote data messages.
The remote system will respond with a DATA_FREQ_ASSERT (#16).

0	Field String	$DFQ

Example:
`$DFQ`

16 – DFA – DATA_FREQ_ASSERT

The message contains the current frequency rate of the data messages.

0	Field String	$DFA
1	COM frequency	The current frequency rate (messages per second) to Serial
2	BT frequency	The current frequency rate (messages per second) to Bluetooth
3	SD frequency	The current frequency rate (messages per second) to SD Card

Example: (A serial frequency of 10 Hz, Bluetooth frequency of 1 Hz, and SD frequency of 30 Hz)
`$DFA,10,1,30`

17 – FMQ – FILE_MANAGER_REQ

Manage Log Files on SD card.

0	Field String	$FMQ

1	Command	`LST = FILE_LIST`
		`NEW = NEW_FILE`
		`DEL = DELETE_FILE`
		`PRP = FILE_PROPERTIES`
		`DMP = FILE_DUMP`

2	Operation parameter	`Filename, if file list is selected is left void`

Example: (Get the list of file currently on the SD root directory)
`FMQ,LST`
Example: (Creates the file LOG1.CSV)
`FMQ,NEW,LOG1.CSV`
Example: (Deletes the file LOG1.CSV)
`FMQ,DEL,LOG1.CSV`
Example: (Gets LOG1.CSV file properties)
`FMQ,PRP,LOG1.CSV`
Example: (Dumps PAGNOTTA.CSV file content)
`FMQ,DMP,PAGNOTTA.CSV`

18 – FMA – FILE_MANAGER_ASSERT

Data from SD files. Payload depends on #17 FMQ Message

In reply to msg #17 `FILE_LIST = LST`
Number of total parameters equal to 3n+2, n is the number of files present on the SD card.

0	Field String	$FMA
1	Command	LST
2	Number of files	Integer, the number of transmitted files
3	Field2	Filename #1
4	Field3	Size file #1
5	Field4	Timestamp #1; Time as seconds since Jan 1, 1970 (Long Integer)
6	Field5	Filename #2
7	Field6	Size file #2
…	…	…
3n+2	Field(3n+2)	Timestamp #n; Time as seconds since Jan 1, 1970 (Long Integer)

Example: (The root directory contains 3 files)
`$LFA,LST,3,DATALOG.CSV,1560,1517256700,LOG1.CSV,0,1517251200,PAGNOTTA.CSV,68203,1517256745`

In reply to msg #17 `NEW_FILE = NEW`
Acknowledge the creation of a new file. The new file is set as current ADC log file.

0	Field String	$FMA
1	Command	NEW

2	File name	The name of the newly created file Use the “?” character to create a new one using a progressive number

Example:
`$FMA,NEW,PAGNOTTA.CSV` *(PAGNOTTA.CSV created)*
Example:
$FMA,NEW,LOG_?.CSV *(if LOG_0000.CSV exists, LOG_0001.CSV created)*

In reply to msg #17 `DEL_FILE = DEL`
Acknowledge the deletion of specified file.

0	Field String	$FMA
1	Command	DEL
2	File name	The name of the file

Example:
`$FMA,DEL,PAGNOTTA.CSV` *(PAGNOTTA.CSV deleted)*
`$FMA,DEL` *(No files deleted)*

In reply to msg #17 `FILE_PROPERTIES = PRP`
Returns the name and the dimension of the specified file.

0	Field String	$LFA
1	Command	PRP
2	Name	The name of the file
3	Size	Size in bytes of specified file
4	Timestamp	Time as seconds since Jan 1, 1970 (Long Integer)

Example:
`$FMA,PRP,PAGNOTTA.CSV,15600,1517256700` *(returns PAGNOTTA.CSV properties)*
`$FMA,PRP` *(returns nothing)*

In reply to msg #17 `FILE_DUMP = DMP`
Returns the log file contents line by line.
Another message received by the same serial interface will stop the dump truncating the file
Message is always ended with a special $EOF sequence.

In order, it returns:
1) The $FMA string:

0	Field String	$FMA
1	Command	DMP
2	Name	The name of the file

2) The file content:
First line
Second line
…. (Send another message to stop this phase and truncate the file)
Last line

3) The End of File acknowledgment
$EOF

Example: (it dumps PAGNOTTA.CSV file)

```
$FMA,DMP,PAGNOTTA.CSV
$DTA,first_line
...
$DTA,last_line
$EOF
```

19 –CCS–CALIBRATION_CON_SET

Manages calibration parameter of sensors.

General command message structure

0	Field String	$CCS
1	Command	EXE – Order to perform a calibration on the specified sensor and with the mode specified USE – Applies the specified calibration parameters to the sensor SEN – Send out the calibration data for the specified sensor
2	Sensor_ID_number	The ID number of the sensorr
3	Mode	1 – Offset Only 2 - Linear
4	Parameter 1	Command dependent
5	Parameter 2	Command dependent

EXE – Command description

After a EXE command is issued the ADC performs the required calibration routine on the sensors and update the required sensor information. After the procedure is complete the ADC sent out a message #20 CCQ.

0	Field String	$CCS
1	Command	EXE
2	Sensor_ID_number	The ID number of the sensor. Integer, 2 Bytes
3	Mode	1 – Offset Only, calibrate the indicated sensor zero offset 2 – Linear, calibrate the indicated sensor zero offset and gain

EXAMPLE:
$CCS,EXE,1,1

Once finished sent out a message #20 containing the offset parameter used

USE – Command description

The ADC uses the specified calibration information and reply with a #20 message.

0	Field String	$CCS
1	Command	USE
2	Sensor_ID_number	The ID number of the sensor. Integer, 2 Bytes
3	Mode	1 – Offset Only, calibrate the indicated sensor zero offset 2 – Linear, calibrate the indicated sensor zero offset and gain

4	Parameter 1	Offset if Mode==1
5	Parameter 2	Gain if Mode==2

EXAMPLE:
$CCS,USE,2,1,45,,
The ADC sets the sensor 2 offset to 45, then send out a #20 message $CGQ,2,45,,

SEN – Command description
Send out the calibration data for the specified sensor. Returns calibration offset and gain

0	Field String	$CCS
1	Command	SEN
2	Sensor_ID_number	The ID number of the sensorr

Example:
$CCS,SEN,1
Reply
$CCA,1,1234,0.995955

HWD– Command description

The ADC uses the specified sensor hardware information and reply with a #20 message that contains all the data fields.

0	Field String	$CCS
1	Command	HWD
2	Deltap Sensor Min Count value	Minimum number of counts read by the deltap sensor
3	Absolute pressure sensor Min Count value	Minimum number of counts read by the abs pressure sensor
4	Deltap Sensor Max Count value	Maximum number of counts read by the deltap sensor
5	Absolute pressure sensor Max Count value	Maximum number of counts read by the abs pressure sensor
6	Deltap Sensor Min Pressure	Minimum pressure deltap sensor. Pascal
7	Abs Sensor Min Pressure	Minimum pressure abs sensor. Pascal
8	Deltap Sensor Max Pressure	Maximum pressure deltap sensor. Pascal
9	Abs Sensor Max Pressure	Maximum pressure abs sensor. Pascal

20 –CALIBRATION_CON_ASSERT

Reply to message #19, EXE Command

Reply to #19 message . Command EXE mode 1. The reply will be the same of that required to a message #19, Command SEN mode 1 or Message #19, Command USE mode 1

0	Field String	$CCA
1	Sensor_ID_number	The ID number of the sensor. Integer, 2 Bytes
2	Parameter 1	Offset value. Integer, 6 bytes

EXAMPLE:
In reply to #19 $CCS,EXE,1,1
$CCA,1,25
The #20 reply contents the used offset value 25 for the sensor 1

Reply to message #19, USE Command

Reply to #19 message . Command USE mode 2, Command SEN mode 2

0	Field String	$CCA
1	Sensor_ID_number	The ID number of the sensor. Integer, 2 Bytes
2	Parameter 1	Offset value. Integer, 6 bytes
3	Parameter 2	Gain

EXAMPLE:
In reply to #19 $CCS,USE,1,2
$CCA,1,25.0.99995
The #20 reply contents the used offset value 25 and the gain 0,99995 for the sensor 1