API Documentation: Dual Fuel System Data Communication

Data Format: Hardware to App

The data is exchanged in JSON format. The hardware sends different sensor's & RPM data's and hardware status to the app and the app sends configuration data and complete engine tuning parameters to the hardware.

ECU to App (Tx Data):

The Hardware sends the following data to the app in json:

JSON Structure:

```
{
  "supplyVoltage": <float>,   // Supply voltage in Volts
  "fuelTemperature": <float>, // Fuel temperature in Celsius
  "engineTemperature": <float>, // Engine temperature in Celsius
  "manifoldPressure": <float>, // Manifold pressure in kPa
  "gasPressure": <float>,     // Gas pressure in kPa
  "o2SensorValue": <float>,   // Oxygen sensor value in Volts
  "lpgFuelLevel": <float>,    // LPG fuel level percentage (0-100%)
  "rpm": <int>,               // Engine RPM
  "injectionTime": <float>,   // Fuel injection time in milliseconds
  "interpolatedFuelInjectionTime": <float>, // Interpolated fuel injection time in milliseconds
  "mode": <string>,           // Current mode ("M0", "M1", or "M2")
  "fuelLevelState": <string> // LPG fuel level state ("L0", or "L1", "L2", "L3" and "L4")
}
```

Data fields:

supplyVoltage (float): Suypply voltage in volts (V).
fuelTemperature (float): Fuel temperature in degrees Celsius (°C).
engineTemperature (float): Engine temperature in degrees Celsius (°C).
manifoldPressure (float): Manifold pressure in bar (range 0.00 to 1.00).
gasPressure (float): Gas pressure in bar (range 0.00 to 1.00).
o2SensorValue (float): Oxygen sensor value in volts (V).
lpgFuelLevel (float): LPG fuel level in percentage (% or EMPTY, QUARTER, HALF, THREE QUARTER and FULL options to choose).
rpm (unsigned int): Engine speed in revolutions per minute (RPM).
injectionTime (float): Fuel injection time in milliseconds (ms).
workingMode (int): M0-petrol, M1-nuetral, M2-GAS.

App to Hardware (Rx data):

The app sends the following configuration data and tuning parameters to the ECU:
json

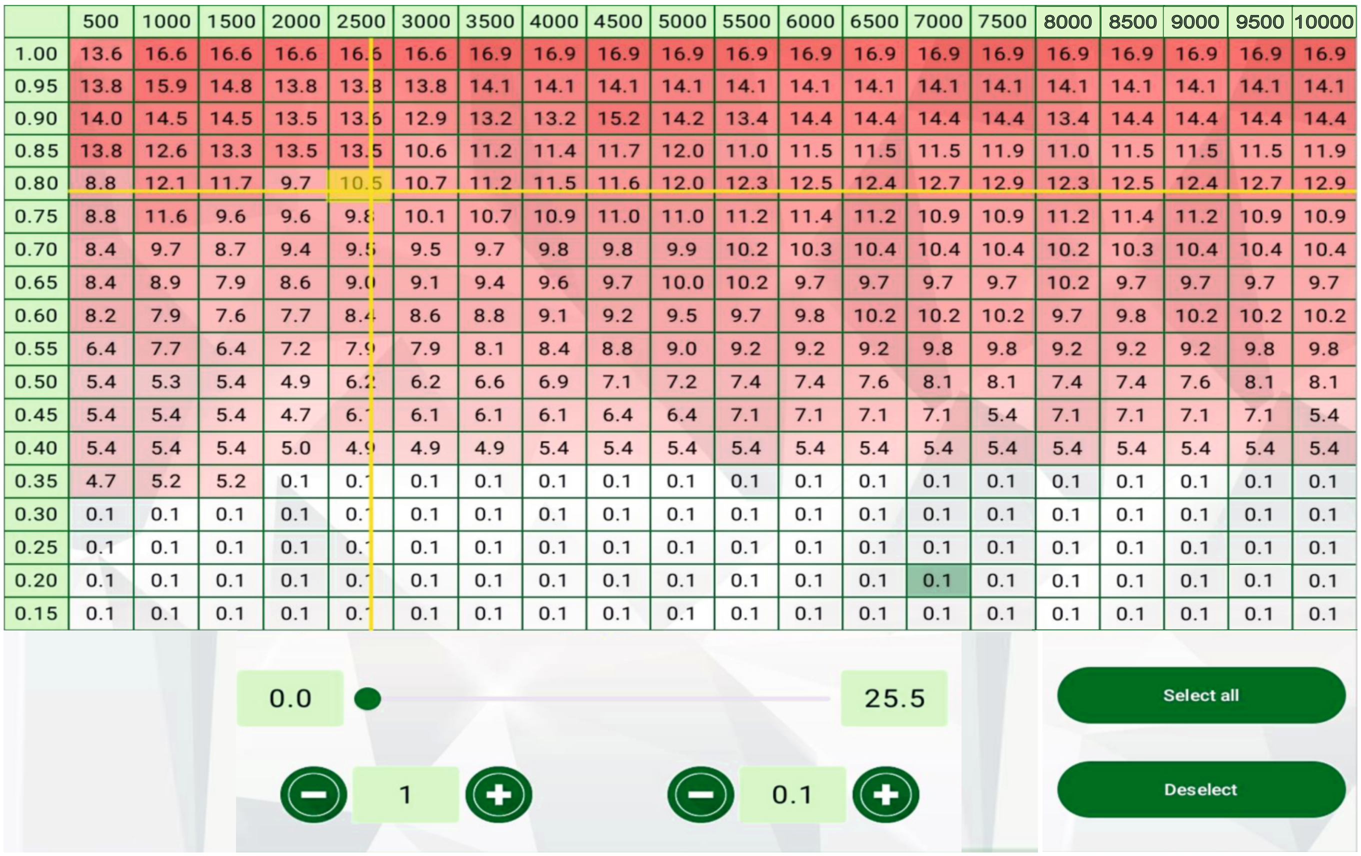
```
{
  "fuelMapTable": [
    [2.0, 2.1, 2.2, ...],
    [2.5, 2.6, 2.7, ...],
    ...
  ]
}
```

Data fields:

fuelMapTable (2D array of floats): The tuned fuel map table values representing the fuel injection time in milliseconds (ms) for different load and RPM combinations.
The dimensions of the fuelMapTable are defined by MAP_TABLE_LOAD_SIZE (Y-axis) (refer screen shot which already shared) and MAP_TABLE_RPM_SIZE (X-axis) (refer screen shot which already shared).

```
const uint8_t MAP_TABLE_LOAD_SIZE = 18;  // (1.0 - 0.15) / 0.05 + 1 = 18 ( Y-axis)
const uint8_t MAP_TABLE_RPM_SIZE = 19;   // (10000 - 500) / 500 + 1 = 19 (X-axis)
```

Note: That the X-axis can be changed to top or bottom options required and refer below screen.



	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000	7500	8000	8500	9000	9500	10000
1.00	13.6	16.6	16.6	16.6	16.6	16.6	16.9	16.9	16.9	16.9	16.9	16.9	16.9	16.9	16.9	16.9	16.9	16.9	16.9	16.9
0.95	13.8	15.9	14.8	13.8	13.8	13.8	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1	14.1
0.90	14.0	14.5	14.5	13.5	13.6	12.9	13.2	13.2	15.2	14.2	13.4	14.4	14.4	14.4	14.4	13.4	14.4	14.4	14.4	14.4
0.85	13.8	12.6	13.3	13.5	13.5	10.6	11.2	11.4	11.7	12.0	11.0	11.5	11.5	11.5	11.9	11.0	11.5	11.5	11.5	11.9
0.80	8.8	12.1	11.7	9.7	10.5	10.7	11.2	11.5	11.6	12.0	12.3	12.5	12.4	12.7	12.9	12.3	12.5	12.4	12.7	12.9
0.75	8.8	11.6	9.6	9.6	9.8	10.1	10.7	10.9	11.0	11.0	11.2	11.4	11.2	10.9	10.9	11.2	11.4	11.2	10.9	10.9
0.70	8.4	9.7	8.7	9.4	9.5	9.5	9.7	9.8	9.8	9.9	10.2	10.3	10.4	10.4	10.4	10.2	10.3	10.4	10.4	10.4
0.65	8.4	8.9	7.9	8.6	9.0	9.1	9.4	9.6	9.7	10.0	10.2	9.7	9.7	9.7	9.7	10.2	9.7	9.7	9.7	9.7
0.60	8.2	7.9	7.6	7.7	8.4	8.6	8.8	9.1	9.2	9.5	9.7	9.8	10.2	10.2	10.2	9.7	9.8	10.2	10.2	10.2
0.55	6.4	7.7	6.4	7.2	7.9	7.9	8.1	8.4	8.8	9.0	9.2	9.2	9.2	9.8	9.8	9.2	9.2	9.2	9.8	9.8
0.50	5.4	5.3	5.4	4.9	6.2	6.2	6.6	6.9	7.1	7.2	7.4	7.4	7.6	8.1	8.1	7.4	7.4	7.6	8.1	8.1
0.45	5.4	5.4	5.4	4.7	6.1	6.1	6.1	6.1	6.4	6.4	7.1	7.1	7.1	7.1	5.4	7.1	7.1	7.1	7.1	5.4
0.40	5.4	5.4	5.4	5.0	4.9	4.9	4.9	5.4	5.4	5.4	5.4	5.4	5.4	5.4	5.4	5.4	5.4	5.4	5.4	5.4
0.35	4.7	5.2	5.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.30	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.25	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.20	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.15	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

The fuel map table is a 2D array of floats representing the fuel injection time in milliseconds (ms) f
or different load and RPM combinations.

The dimensions of the fuel map table are defined by MAP_TABLE_LOAD_SIZE and MAP_TABLE_RPM_SIZE constants in the code. The app sends the tuned fuel MAP table data to the hardware as a **JSON** array of arrays.

The hardware receives the tuned fuel map table, updates the global fuelMapTable array, and stores it in the EEPROM.

EEPROM Storage:

The tuned fuel map table received from the app is stored in the EEPROM using the function of writeTunedFuelMapTableToEEPROM().

The stored fuel map table is read from the EEPROM during Hardware initialization using the function of readTunedFuelMapTableFromEEPROM().

The EEPROM storage ensures that the tuned fuel map table persists across hardware power cycles.
This API documentation provides a detailed overview of the data structure and format for the communication between the hardware control and the desktop/mobile app based on the provided code.

The hardware sends sensor data and status information to the app in JSON format, including supply voltage, fuel temperature, engine temperature, manifold pressure, gas pressure, oxygen sensor value, LPG fuel level, engine speed (RPM), fuel injection time, and working mode.

The app sends the tuned fuel map table to the hardware as a 2D array of floats representing the fuel injection time for different load and RPM combinations. The hardware receives the tuned fuel MAP table, updates the global fuelMapTable array, and stores it in the EEPROM for persistence.

The communication between the hardware and the app is performed over a serial connection (UART) or Wifi / BLE using JSON format.

This API documentation is based on the hardware code and assumes that the third-party App developer will implement the corresponding functionality in the desktop/mobile app to send and receive / transmit the data in the specified format.

If the APP developer requires any additional information or clarification, please let me know, and I'll be happy to assist further.