

Nissan Leaf 2018

March 12, 2021

ECU	ID Query	ID Response
Vehicle Control Module (VCM)	0x797	0x79A
Body Control Module (BCM)	0x743	0x763
AntiBlockierSystem (ABS)	0x740	0x760
Li-ion Battery Controller (LBC)	0x79B	0x7BB
Traction Motor Inverter (INV/MC)	0x784	0x78C
Meter	0x745	0x765
HVAC	0x744	0x764

Table 1: Nissan Leaf 2018 ECU IDs

Vehicle Control Module (VCM)

Name: Power_Software

Type: Boolean

Description: Power_SW is True if the Leaf's power switch is ON otherwise it is False. The fifth reply byte value is 0x80 if the car is powered on.

Query: 0x797 03 22 13 04 00 00 00 00

Answer: 0x79A 05 62 13 04 **80** FE 00 00

Formula: `Power_SW = (data[4] & 128 == 128);`

Name: Gear

Type: Integer

Description: Gear position (1=Park, 2=Reverse, 3=Neutral, 4=Drive, 7=Eco). The fifth reply byte value is the gear value. In the example gear is 1.

Query: 0x797 03 22 11 56 00 00 00 00

Answer: 0x79A 04 62 11 56 **01** 00 00 00

Formula: `Gear = data[4]`

Name: Bat_12V_Voltage (V)

Type: Integer

Description: The voltage of the 12 volt battery as read by the Leaf's Vehicle Control Module ECU. The Bat_12V_Voltage is the fifth reply byte value times 0.08. In the example Bat_12V_Voltage is 13.04V

Query: 0x797 03 22 11 03 00 00 00 00

Answer: 0x79A 04 62 11 03 **A3** 00 00 00

Formula: `Bat_12V_Voltage = (data[4] * 0.08)`

Name: Bat_12V_Current (A)

Type: Integer

Description: 12 volt battery current. Minus is drain and positive is charging. In the example Bat_12V_Current is -0.5546875A

Query: 0x797 03 22 11 83 00 00 00 00

Answer: 0x79A 05 62 11 83 **FF 72** 00 00

Formula:

```
Bat_12V_Current_temp = ( data[4] << 8 ) | data[5];  
if (Bat_12V_Current_temp & 32768 == 32768):  
    Bat_12V_Current_temp = Bat_12V_Current_temp | -65536; //Negative value  
Bat_12V_Current = Bat_12V_Current_temp / 256
```

Name: Quick_Charge

Type: Integer

Description: Number of quick charges (QC). Each time a quick charge occurs this number increments by 1, even not fully. In the example Quick_Charge is 1.

Query: 0x797 03 22 12 03 00 00 00 00

Answer: 0x79A 05 62 12 03 **00 01** 00 00

Formula: `Quick_Charge = (data[4] << 8) | data[5]`

Name: L1_L2_Charge

Type: Integer

Description: Number of L1/L2 connections and charges, even not fully. In the example L1_L2_Charge is 115.

Query: 0x797 03 22 12 05 00 00 00 00

Answer: 0x79A 05 62 12 05 **00 73** 00 00

Formula: `L1_L2_Count = (data[4] << 8) | data[5]`

Name: Ambient_C_Temp (°C)

Type: Integer

Description: Ambient temperature comes from the Leaf's outside temperature sensor and is firstly got in °F. It's then converted to °C. In the example the Ambient_C_Temp is 24 °C.

Query: 0x797 03 22 11 5D 00 00 00 00

Answer: 0x79A 04 62 11 5D **81** 00 00 00

Formula:

```
Ambient_F_Temp = (data[4] * 0.9) - 40.9;  
Ambient_C_Temp = (Ambient_F_Temp - 32) * 5/9
```

Name: EstPwr_AC_50W (W)

Type: Integer

Description: Estimated Air Conditioning system power (COOL)

Query: 0x797 03 22 12 61 00 00 00 00

Answer: 0x79A 04 62 12 61 **02** 00 00 00

Formula: `EstPwr_AC_50W = data[4]*50`

Name: EstPwr_Htr_250W (W)

Type: Integer

Description: Estimated Cabin PTC heater power (HEAT)

Query: 0x797 03 22 12 62 00 00 00 00

Answer: 0x79A 04 62 12 62 **01** 00 00 00

Formula: `EstPwr_Htr_250W = data[4]*250`

Name: Aux_Pwr_100W (W)

Type: Integer

Description: Power used by the auxiliary equipment (Lights, Radio, Navigation system, rear defroster...)

Query: 0x797 03 22 11 52 00 00 00 00

Answer: 0x79A 04 62 11 52 **02** 00 00 00

Formula: `Aux_Pwr_100W = data[4]*100`

Name: AC_Pwr_250W (W)

Type: Integer

Description: A/C system Power (250W): Power used by the Air Conditioning System power. This includes the power used by the cabin PTC Heater.

Query: 0x797 03 22 11 51 00 00 00 00

Answer: 0x79A 04 62 11 51 **01** 00 00 00

Formula: `AC_Pwr_250W = data[4]*250`

Name: PlugState

Type: Integer

Description: Plug state of J1772 (L1/L2) charge port (0=Not plugged, 1=Partial Plugged, 2=Plugged)

Query: 0x797 03 22 12 34 00 00 00 00

Answer: 0x79A 04 62 12 34 **02** 00 00 00

Formula: `PlugState = data[4]`

Name: Charge_Mode

Type: Integer

Description: Charging mode. (0=Not charging, 1=Level 1 charging [100-120 volts], 2=Level 2 charging [200-240 volts]), 3=Level 3 Quick Charging)

Query: 0x797 03 22 11 4E 00 00 00 00

Answer: 0x79A 04 62 11 4E **02** 00 00 00

Formula: `Charge_mode = data[4]`

Name: RPM (r/min)

Type: Integer

Description: Motor revolutions per minute

Query: 0x797 03 22 12 55 00 00 00 00

Answer: 0x79A 05 62 12 55 **FF AB** 00 00

Formula:

```
RPM = ( data[4] << 8 ) | data[5];
```

```
if (RPM & 32768 == 32768):
```

```
    RPM = RPM | -65536;
```

Name: VIN

Type: String

Description: The vehicle identification number indicates which vehicle the data came from. The string of HEX must be converted to UTF-8.

Query: 0x797 02 21 81 00 00 00 00 00

0x797 02 30 00 00 00 00 00 00

Answer:

0x79A 10 15 61 81 **53 4A 4E 46**

0x79A 21 **41 41 5A 45 31 55 30**

0x79A 22 **30 35 33 39 32 37 00**

0x79A 23 00 00 00 00 00 00 00

Formula: `(534A4E4641415A45315530303533393237).decode('utf-8')`

Name: OBC_Out_Pwr (W)

Type: Integer

Description: Charging power coming into the Leaf from the on-board charger, when the car is charging.

Query: 0x797 03 22 12 36 00 00 00 00

Answer: 0x79A 05 62 12 36 **00 15** 00 00

Formula: `OBC_Out_Pwr = ((data[4] << 8) | data[5])*100;`

Name: Motor_Pwr (W)

Type: Integer

Description: Driving motor power.

Query: 0x797 03 22 11 46 00 00 00 00

Answer: 0x79A 04 62 11 46 **00 27** 00 00

Formula: `Motor_Pwr = ((data[4] << 8) | data[5])*40;`

Name: Speed (km/h)

Type: Integer

Description: Car speed

Query: 0x797 03 22 12 1A 00 00 00 00

Answer: 0x79A 05 62 12 1A **00 01** 00 00

Formula: `Speed = ((data[4] << 8) | data[5])/10;`

Name: AC

Type: Boolean

Description: AC status (On/Off)

Query: 0x797 03 22 11 06 00 00 00 00

Answer: 0x79A 05 62 11 06 **01** 7f 00 00

Formula: `AC = data[4]`

Name: RearHeater

Type: Boolean

Description: Rear heater status (On/Off)

Query: 0x797 03 22 11 0F 00 00 00 00

Answer: 0x79A 04 62 11 0F **A2** 00 00 00

Formula: `RearHeater = (HEX(data[4]) == '0xA2')`

Name: ECO

Type: Boolean

Description: ECO mode status (On/Off)

Query: 0x797 03 22 13 18 00 00 00 00

Answer: 0x79A 05 62 13 1B **10** 39 00 00

Formula: `ECO = (HEX(data[4]) == '0x10' or HEX(data[4]) == '0x11')`

Name: e-Pedal

Type: Boolean

Description: e-Pedal mode (On/Off). The e-Pedal[?] allows the driver to start, accelerate, decelerate and stop using only the accelerator pedal.

Query: 0x797 03 22 13 1A 00 00 00 00

Answer: 0x79A 05 62 13 1A **04** 14 00 00

Formula: `e-Pedal = (HEX(data[4]) == '0x4')`

Body Control Module (BCM)

Name: Odometer (km)

Type: Integer

Description: Total Km run by the car from the production.

Query: 0x743 03 22 0E 01 00 00 00 00

Answer: 0x763 06 62 0E 01 **00 04 DB** 00

Formula: `Odometer = (data[4] << 16 | ((data[5] << 8) | data[6]))`

Name: TP_FR (kPa)

Type: Integer

Description: Front right tire pressure

Query: 0x743 03 22 0E 25 00 00 00 00

Answer: 0x763 04 62 0E 25 **92 FF FF FF**

Formula: `TP_FR = (data[4] * 0.068947576) * 100 / 4`

Name: TP_FL (kPa)

Type: Integer

Description: Front left tire pressure

Query: 0x743 03 22 0E 26 00 00 00 00

Answer: 0x763 04 62 0E 26 **94 FF FF FF**

Formula: `TP_FL = (data[4] * 0.068947576) * 100 / 4`

Name: TP_RR (kPa)

Type: Integer

Description: Rear right tire pressure

Query: 0x743 03 22 0E 27 00 00 00 00

Answer: 0x763 04 62 0E 27 **94 FF FF FF**

Formula: `TP_RR = (data[4] * 0.068947576) * 100 / 4`

Name: TP_RL (kPa)

Type: Integer

Description: Rear left tire pressure

Query: 0x743 03 22 0E 28 00 00 00 00

Answer: 0x763 04 62 0E 28 **94 FF FF FF**

Formula: `TP_RL = (data[4] * 0.068947576) * 100 / 4`

Name: Range (km)

Type: Integer

Description: Remaining km to turtle mode

Query: 0x743 03 22 0E 24 00 00 00 00

Answer: 0x763 05 62 0E 2E **00 DD** 00 00

Formula: `Range = ((data[4] << 8) | data[5])/10;`

Li-Ion Battery Controller (LBC)

The Li-ion battery controller accepts the multi-message query only. In fact, the LBC transmits many groups: the first one contains lots of High Voltage battery data as SOC, currents, and voltage; the second replies with all the battery's cells voltages in millivolt, the third and the fifth one are still unknown, the fourth contains the four battery packs temperatures, and the last one tells which cell has the shunt active. There are also two more groups: group 61, which replies with lots of CAN messages (up to 48); here we found the SOH value, and group 84 that replies with the HV battery production serial.

0.0.1 Group 1

Query:

0x79B 02 21 01 00 00 00 00 00

0x79B 30 00 00 00 00 00 00 00

Answer:

0x7BB 10 35 61 01 FF FF FC 18

0x7BB 21 02 AF FF FF FB 62 FF

0x7BB 22 FF F0 DD 0B 1C 30 D4

0x7BB 23 95 1D 33 06 03 95 00

0x7BB 24 01 70 00 26 9A 00 0C

0x7BB 25 44 B5 00 11 0B B8 80

0x7BB 26 00 01 FF FF FB 62 FF

0x7BB 27 FF FC AA 01 AD FF FF

Name: Hx (%)

Type: Integer

Description: High Voltage Battery's health. In a factory new condition, the battery's HX is 100%. The number is thought to be an indication of battery internal resistance with 100 indicating a new battery with the lowest resistance. As the value decreases the internal resistance is increasing. As the internal resistance increases more energy is wasted as heat inside the battery instead of powering the Leaf.

Formula:

0x7BB 24 01 70 00 **26 9A** 00 0C

$HX = ((data[4] \ll 8) \mid data[5]) / 102.4$

Name: SOC (%)

Type: Integer

Description: State of Charge (SOC) of the HV Battery

Formula:

0x7BB 24 01 70 00 26 9A 00 **0C**

0x7BB 25 **44 B5** 00 11 0B B8 80

$SOC = (data_24[7] \ll 16 \mid ((data_25[1] \ll 8) \mid data_25[2])) / 10000$

Name: Ahr (Ah)

Type: Integer

Description: Capacity of HV Battery: e.g. how much energy the battery could hold when fully charged.

Formula:

0x7BB 25 44 B5 00 **11 0B B8** 80

$AHR = (data[4] \ll 16 \mid ((data[5] \ll 8) \mid data[6])) / 10000$

Name: GIDs

Type: Integer

Description: GIDs is a Nissan's number that indicate the energy in the Leaf battery. Nissan multiplies this value by 80 Wh to get current capacity of the HV Battery. Discovered by Gary Giddings. The Gid measurement is temperature dependent: understates charge in hot weather and overstates it in cold weather.

Formula: $GIDS = ((4.425 * (SOC * 0.01)) * AHR)$

Name: HV_Bat_Current_1 (A)

Type: Integer

Description: High Voltage battery current; it's positive if the car is running/driving while it's negative when it's regenerating (by braking) or charging.

Formula:

0x7BB 10 35 61 01 **FF FF FC 18**

```
HV_Bat_Current_1 = (data[4] << 24) | (data[5] << 16 | ((data[6] << 8) | data[7]))
```

```
if(HV_Bat_Current_1 & 0x80000000 == 0x80000000):
```

```
    HV_Bat_Current_1 = ( HV_Bat_Current_1 | -0x100000000 ) / 1024
```

```
else:
```

```
    HV_Bat_Current_1 = HV_Bat_Current_1 / 1024
```

Name: HV_Bat_Current_2 (A)

Type: Integer

Description: High Voltage battery current; it's positive if the car is running/driving while it's negative when it's regenerating (by braking) or charging. It's not clear why there are two different HV current values.

0x7BB 21 02 AF **FF FF FB 62** FF

```
HV_Bat_Current_2 = (data[3] << 24) | (data[4] << 16 | ((data[5] << 8) | data[6]))
```

```
if(HV_Bat_Current_2 & 0x80000000 == 0x80000000):
```

```
    HV_Bat_Current_2 = ( HV_Bat_Current_2 | -0x100000000 ) / 1024
```

```
else:
```

```
    HV_Bat_Current_2 = HV_Bat_Current_2 / 1024
```

Name: HV_Bat_Voltage (V)

Type: Integer

Description: High Voltage battery voltage

Formula:

0x7BB 23 **95 1D** 33 06 03 95 00

```
HV_Bat_Voltage = ((data[1] << 8) | data[2]) / 100
```

Name: Insulation

Type: Integer

Description: Unknown

Formula:

0x7BB 23 95 1D 33 06 **03 95** 00

```
Insulation = ((data[5] << 8) | data[6])
```

0.0.2 Group 2

Name: Cells voltage (mV)

Type: Integer

Description: These are the 96 cell pair voltages measured in millivolts.

Query:

0x79B 02 21 02 00 00 00 00 00

0x79B 30 00 00 00 00 00 00 00

Answer:

0x7BB 10 C6 61 02 **10 6F 10 6E**

0x7BB 21 **10 6D 10 6F 10 6E 10**

0x7BB 22 **6F 10 70 10 6E 10 6F**

...

Formula:

CV_array[0] = ((data_10[4] << 8) | data_10[5])

CV_array[1] = ((data_10[6] << 8) | data_10[7])

CV_array[2] = ((data_21[1] << 8) | data_21[2])

CV_array[3] = ((data_21[3] << 8) | data_21[4])

CV_array[4] = ((data_21[5] << 8) | data_21[6])

CV_array[5] = ((data_21[7] << 8) | data_22[1])

...

So far, the **group 3**'s data is unknown.

Group 4

Name: Packs temperature (°C)

Type: Integer

Description: HV Battery temperature sensors from the four packs.

Query:

0x79B 02 21 04 00 00 00 00 00

0x79B 30 00 00 00 00 00 00 00

Answer:

0x7BB 10 1F 61 04 **02 0D 13 02**

0x7BB 21 **03 14 FF FF FF 02 0B**

0x7BB 22 13 13 00 FF FF FF FF

Formula:

Temp_raw_1 = (data_10[4] << 8 | (data_10[5]))

Temp_raw_2 = (data_10[7] << 8 | data_21[1])

Temp_raw_3 = (data_21[3] << 8 | data_21[4])

Temp_raw_4 = (data_21[6] << 8 | data_21[7])

Pack_Temp_C_1 = ((Temp_fromRAW_to_F(Temp_raw_1) - 32) * 5) / 9

Pack_Temp_C_2 = ((Temp_fromRAW_to_F(Temp_raw_2) - 32) * 5) / 9

Pack_Temp_C_3 = ((Temp_fromRAW_to_F(Temp_raw_3) - 32) * 5) / 9

Pack_Temp_C_4 = ((Temp_fromRAW_to_F(Temp_raw_4) - 32) * 5) / 9

The conversion from RAW temperature to °F is reported in the algorithm 1.

Algorithm 1: Temp_fromRAW_to_F(i)

```
    if i == 1021 then
        return 1.0
3:  else if i ≥ 589 then
        return 162.0 - (i * 0.181)
    else if i ≥ 569 then
6:    return 57.2 + ((579 - i) * 0.18)
    else if i ≥ 558 then
        return 60.8 + ((558 - i) * 0.16363636363636364)
9:  else if i ≥ 548 then
        return 62.6 + ((548 - i) * 0.18)
    else if i ≥ 537 then
12:   return 64.4 + ((537 - i) * 0.16363636363636364)
    else if i ≥ 447 then
        return 66.2 + ((527 - i) * 0.18)
15:  else if i ≥ 438 then
        return 82.4 + ((438 - i) * 0.2)
    else if i ≥ 428 then
18:   return 84.2 + ((428 - i) * 0.18)
    else if i ≥ 365 then
        return 86.0 + ((419 - i) * 0.2)
21:  else if i ≥ 357 then
        return 98.6 + ((357 - i) * 0.225)
    else if i ≥ 348 then
24:   return 100.4 + ((348 - i) * 0.2)
    else if i ≥ 316 then
        return 102.2 + ((340 - i) * 0.225)
27:  end if
    return 109.4 + ((309 - i) * 0.2571428571428572);
```

0.0.3 Group 6

Name: Shunts

Type: Integer

Description: Shunts are small resistors that can be switched in to drain a small amount of energy from one or more of the 96 cells that make up the high voltage battery pack. This is the method used by Nissan to balance the pack by draining energy from the high energy cells. This works because charging stops to prevent overcharging the highest energy cell. So by reducing the energy in the highest energy cell, all the other cells are able to be charged to a higher level. The shunt settings (on or off) for the 96 cell pairs are received as 24 groups of four bits for a total of 12B. The shunt order defines how these four bits (numbered 8,4,2,1) are mapped to the four-cell pairs they are associated with.

Query:

0x79B 02 21 06 00 00 00 00 00

0x79B 30 00 00 00 00 00 00 00

Answer:

0x7BB 10 1A 61 06 **14 55 55 51**

0x7BB 21 **50 55 41 2B 56 54 15**

0x7BB 22 **51** FF FF FF FF FF FF

0x7BB 23 FF FF FF FF FF FF FF

Formula:

Algorithm 2: calculateShunts(i)

```
SHUNTS_bits = [-1.0] * 24
SHUNTS_order = "8421"
for bit in range(12): do
4:   SHUNTS_bits[bit*2] = bytes(G6_data)[bit] & 0xF
    SHUNTS_bits[(bit*2) + 1] = bytes(G6_data)[bit] >> 4
end for
for i in range(96): do
8:   i3 = SHUNTS_bits[i >> 2]
    j = int(SHUNTS_order[i % 4])
    if (j & i3) != 0 then
        CV_array[i] = -1*CP_array[i]
12:  end if
end for
```

0.0.4 Group 61

Name: SOH (%)

Type: Integer

Description: State of Health is another indication of the battery's ability to hold and release energy and is reported as a percentage. When the battery is new SOH=100%

Query: 0x79B 02 21 61 00 00 00 00 00

Answer: 0x7BB 11 4B 61 61 26 9A **25 CA**

Formula: `SOH = ((data[6] << 8) | data[7]) / 100`

0.0.5 Group 84

Name: Battery Serial

Type: String

Description: Battery serial number

Query:

0x79B 02 21 84 00 00 00 00 00

0x79B 30 00 00 00 00 00 00 00

Answer:

0x7BB 10 16 61 84 32 33 30 **55**

0x7BB 21 **4B 31 31 39 32 45 30**

0x7BB 22 **30 31 34 38 32 20 A0**

0x7BB 23 00 00 00 00 00 00 00

Formula: `(554b313139324530303134383220a0).decode('utf-8')`

Antilock Braking System (ABS)

Name: SWA (°)

Type: Integer

Description: Steering wheel angle. The angle ranges from -720 to 720 degrees with negative being left turn.

Query: 0x740 03 22 12 08 00 00 00 00

Answer: 0x760 05 62 12 08 **AA BB** 00 00

Formula:

```
SWA = ( data[4] << 8 ) | data[5];  
if (SWA & 32768 == 32768):  
    SWA = SWA | -65536;  
SWA = SWA / 10
```

Name: Brake

Type: Integer

Description: Brake pedal pressure measured from 0 (not pressed) to 369 (max). Unit is unknown.

Query: 0x740 03 22 12 09 00 00 00 00

Answer: 0x760 05 62 12 09 **AA BB** 00 00

Formula: `Brake = (data[4] << 8) | data[5];`

Name: Acc_Pedal

Type: Integer

Description: Acceleration pedal pressure measured from 0 (not pressed) to 200 (max). Unit is unknown.

Query: 0x740 03 22 11 15 00 00 00 00

Answer: 0x760 04 62 11 15 **AA** 00 00 00

Formula: `Acc_Pedal = data[4]`

Traction Motor Inverter (TMI)

Name: Torque (Nm)

Type: Integer

Description: Motor Torque. Torque is a twisting force that speaks to the engine's rotational force and measures how much of that twisting force is available when an engine exerts itself. Minimum unit step is 0.25 Nm.

Query: 0x784 03 22 12 25 00 00 00 00

Answer: 0x78C 05 62 12 25 **00 00** FF FF

Formula:

```
Torque = ( data[4] << 8 ) | data[5];  
if (Torque & 32768 == 32768):  
    Torque = Torque | -65536;  
Torque = Torque / 64.0
```

Name: Motor_Temp (°C)

Type: Integer

Description: This field is the drive motor temperature

Query: 0x784 03 22 11 21 00 00 00 00

Answer: 0x78C 04 62 11 21 **45** FF FF FF

Formula: `Motor_Temp = (data[4] - 40)`

Meter

This single request to METER ECU tells us how the two levers are set. Hence, we can check if the car is turning, the wiper status or the lights status, according to some HEX values.

Query: 0x745 02 21 09 00 00 00 00 00

Answer: 0x765 10 10 61 09 **AA BB CC XX**

```
Wiper = hex(data[4])
Light1 = hex(data[5])
Light2 = hex(data[6])
```

Name: Left_Arrow

Type: Boolean

Description: 0: Off; 1: Blinking - the car is turning left

Formula: (Wiper == '0x22' or Wiper == '0x9') and (Light1 == '0x6')

Name: Right_Arrow

Type: Boolean

Description: 0: Off; 1: Blinking - the car is turning right

Formula: (Wiper == '0x22' or Wiper == '0x9') and (Light1 == '0xa')

Name: Rain_Level

Type: Integer

Description: Rain sensitivity level for front wiper (automatic mode) (1: low, 4: high)

Formula:

```
if(Wiper == '0xf' or Wiper == '0x1f' or Wiper == '0x47' or Wiper == '0x87'); Rain_level = 1
if(Wiper == '0xd' or Wiper == '0x1d' or Wiper == '0x45' or Wiper == '0x85'); Rain_level = 2
if(Wiper == '0xb' or Wiper == '0x13' or Wiper == '0x43' or Wiper == '0x83'); Rain_level = 3
if(Wiper == '0xa' or Wiper == '0x19' or Wiper == '0x41' or Wiper == '0x81'); Rain_level = 4
```

Name: Front_wiper_level

Type: Integer

Description: -1: Off; 0: Automatic; 1: Continuous slow speed; 2: Continuous high speed

Formula:

```
if(Wiper == '0xf' or Wiper == '0xd'); Front_wiper_level = -1
if(Wiper == '0xb' or Wiper == '0xa'); Front_wiper_level = -1
if(Wiper == '0x1f' or Wiper == '0x1d'); Front_wiper_level = 0
if(Wiper == '0x13' or Wiper == '0x19'); Front_wiper_level = 0
if(Wiper == '0x47' or Wiper == '0x45'); Front_wiper_level = 1
if(Wiper == '0x43' or Wiper == '0x41'); Front_wiper_level = 1
if(Wiper == '0x87' or Wiper == '0x85'); Front_wiper_level = 2
if(Wiper == '0x83' or Wiper == '0x81'); Front_wiper_level = 2
```

Name: Front_washer

Type: Boolean

Description: 0: Not active; 1: Active

Formula: (Wiper == '0x22' or Wiper == '0x9') and (Light1 == '0x1')

Name: Rear_washer

Type: Boolean

Description: 0: Not active; 1: Active

Formula: `(Wiper == '0x22' or Wiper == '0x9') and (Light1 == '0x31')`

Name: Rear_wiper_level

Type: Integer

Description: -1: Off; 1: Intermittent; 2: Continuous low speed

Formula:

```
if(Light1 == '0x41'); Rear_wiper_level = 1;  
if(Light1 == '0x91'); Rear_wiper_level = 2;
```

Name: Position_lights

Type: Integer

Description: -1: Off; 0: Automatic mode; 1: On

Formula:

```
if(Light1 == '0x02' and (Light2 == '0x20' or Light2 == '0x28')); Position_lights = 0  
if(Light1 == '0x02' and (Light2 == '0x00' or Light2 == '0x08')); Position_lights = 1  
if(Light1 == '0x03' and Light2 == '0x08'); Position_lights = 0  
if(Light1 == '0x03' and Light2 == '0x00'); Position_lights = 1
```

Name: Headlights

Type: Integer

Description: -1: Off; 0: Automatic mode; 1: On

Formula:

```
if(Light1 == '0x00' and (Light2 == '0x10' or Light2 == '0x18')); Headlights = 0  
if(Light1 == '0x00' and (Light2 == '0x30' or Light2 == '0x38')); Headlights = 0  
if(Light1 == '0x01' and (Light2 == '0x10' or Light2 == '0x18')); Headlights = 0  
if(Light1 == '0x02' and (Light2 == '0xc0' or Light2 == '0xc8')); Headlights = 1  
if(Light1 == '0x02' and (Light2 == '0xe0' or Light2 == '0xe8')); Headlights = 1  
if(Light1 == '0x03' and (Light2 == '0xc0' or Light2 == '0xc8')); Headlights = 0
```

Name: Anti_fog_lights

Type: Boolean

Description: -1: Off; 1: On

Formula:

```
if(Light1 == '0x00' and (Light2 == '0x18' or Light2 == '0x38')); Anti_fog_lights = 1  
if(Light1 == '0x01' and Light2 == '0x18'); Anti_fog_lights = 1  
if(Light1 == '0x02' and (Light2 == '0x08' or Light2 == '0xc8')); Anti_fog_lights = 1  
if(Light1 == '0x02' and (Light2 == '0x28' or Light2 == '0xe8')); Anti_fog_lights = 1  
if(Light1 == '0x03' and (Light2 == '0x08' or Light2 == '0xc8')); Anti_fog_lights = 1
```

Name: HighBeam

Type: Integer

Description: -1: Off; 0: Automatic mode; 1: On; 2: Manual

Formula:

```
if(Light1 == '0x03' and (Light2 == '0x00' or Light2 == '0x08')); HighBeam = 0  
if(Light1 == '0x01' and (Light2 == '0x10' or Light2 == '0x18')); HighBeam = 1  
if(Light1 == '0x03' and (Light2 == '0xc0' or Light2 == '0xc8')); HighBeam = 1  
if(Light1 == '0x00' and (Light2 == '0x30' or Light2 == '0x38')); HighBeam = 2  
if(Light1 == '0x02' and (Light2 == '0x20' or Light2 == '0xe0')); HighBeam = 2  
if(Light1 == '0x02' and (Light2 == '0x28' or Light2 == '0xe8')); HighBeam = 2
```

Heating, Ventilation and Air Conditioning (HVAC)

Name: HeaterTemp

Type: Integer

Description: Heater temperature from 16.5°C to 30°C set by the driver. Apparently this value increases/decreases accordingly to the heater's UP/DOWN button, but the values interval sometimes shifts. Further analysis are required to check how to convert the HEX values to the range 16.5-30.

Query: 0x744 02 21 10 00 00 00 00 00

Answer:

0x764 10 36 61 10 30 3b 33 0d

0x764 21 2f 3b 00 32 0e 00 80

0x764 22 84 **8D** 00 00 00 00 00

Formula: `HeaterTemp = data[2];`

Name: FanSpeed

Type: Integer

Description: Fan speed from 0 (off) to 8 (max)

Query: 0x744 02 21 10 00 00 00 00 00

Answer:

0x764 10 36 61 10 30 3b 33 0d

0x764 21 2f 3b 00 32 0e 00 80

0x764 22 **84** 8D 00 00 00 00 00

Formula: `FanSpeed = data[1] - 131 ;`