Dear all, just another example on how to use the ELM327 together with an app, to interact with your car.

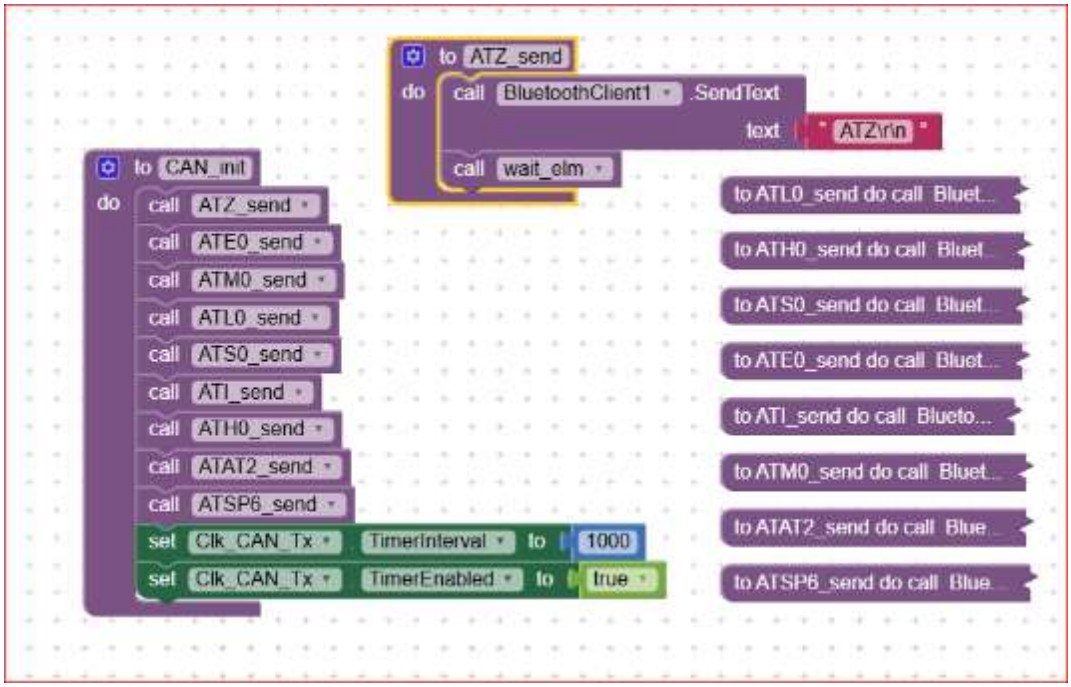
In the following example we'll see how to read and cancel the DTCs, aka Data Trouble Codes, the error codes issued when the car detects a malfunction.

There are plenty of ready made apps that can do the same job, but in this example you will manage everything on your own.

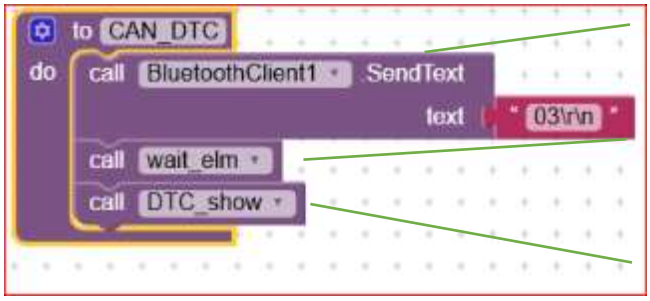
Basically every car manufactured after the year 1996 should (must ?) feature the diagnostic protocol KWP2000 with the UDS (Unified Diagnostic Services). The ELM327 is a microcontroller that has buried in its core a lot of commands and interface protocols. It is typically integrated into a device that features an OBD connector, so to be inserted directly into the OBD socket of the car, but I dont want to bother you about that product, so I'll focus only on the app.

Therefore, the app exchanges commands and data back and forth to the ELM by means of the classic BT client blocks.

The procedure to enter the "diagnostic mode" of a car is made by sending a specific sequence of commands to its central computer (the Body Computer) via the OBD port, which istypically located below the steering wheel, or hidden somewhere else in the dashboard. In my app this sequence is actuated by the procedure "Caninit" that calls, in sequence, the requested commands .



After the diagnostic mode is entered the app asks for reading the Data Trouble Codes (DTCs) by using the UDS command 03 ("service" in the UDS jargon).



The app sends the request to enter the "read DTC" service

Then waits for the answer from the car's ECU via ELM327

Then shows what has been received: no DTCs or a maximum of three DTCs is shown.

The display looks like:

The complete CAN message received is shown here

A maximum of 3 DTCs is shown here

RPM of your engine. Just to see that it is working (*).

Speed of your car, see the NOTE below (*)

Button to Erase all the DTCs (Service 04))

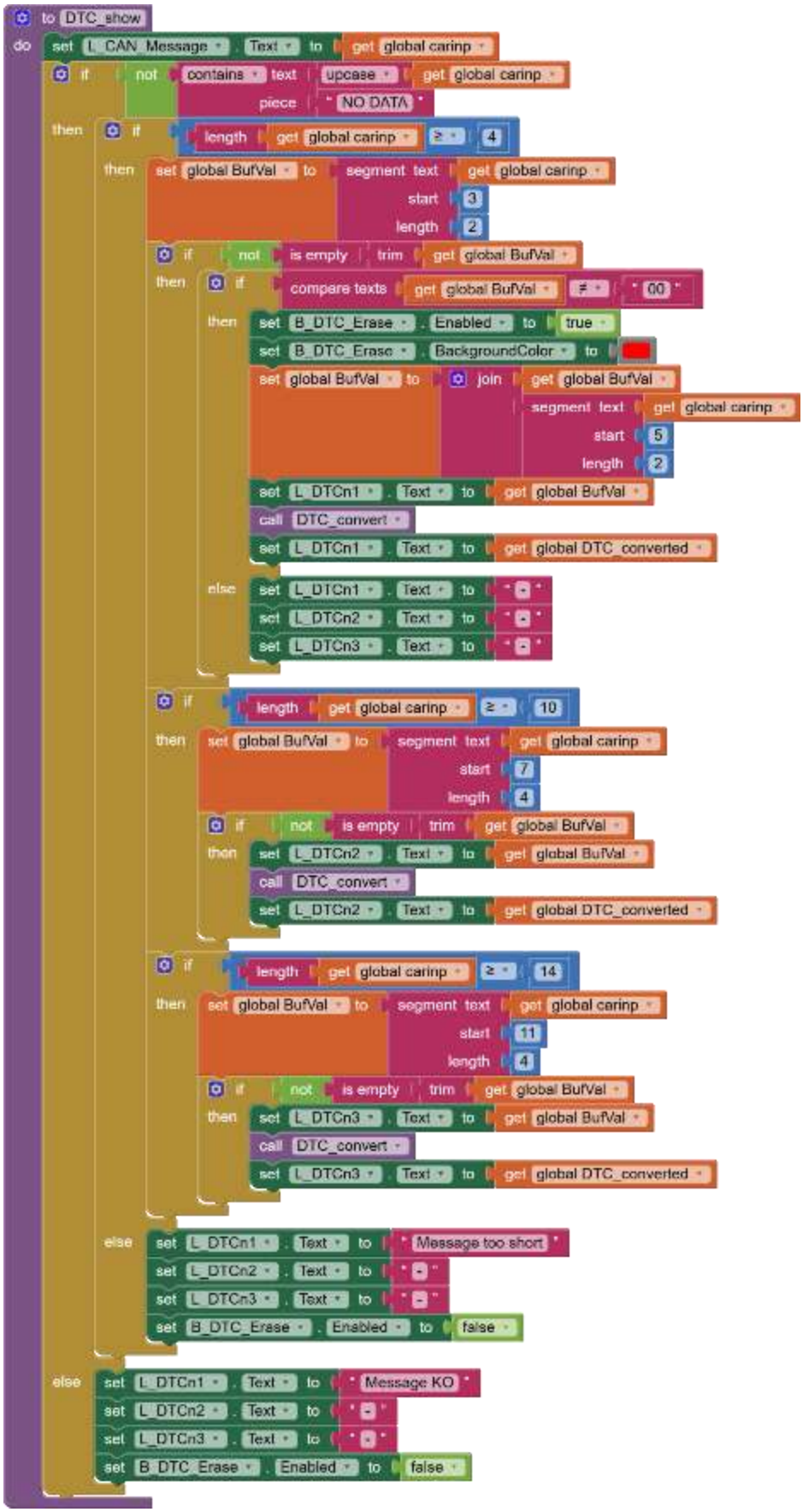
Button to Read the DTCs (Service 03)

(*) **NOTE**: These values (RPM and Speed) aren't necessary for DTCs erasing but they come from a previous app of mine (you can find it in the forum) and I left them if you want to go a bit deeper in the CAR diagnostic protocol.

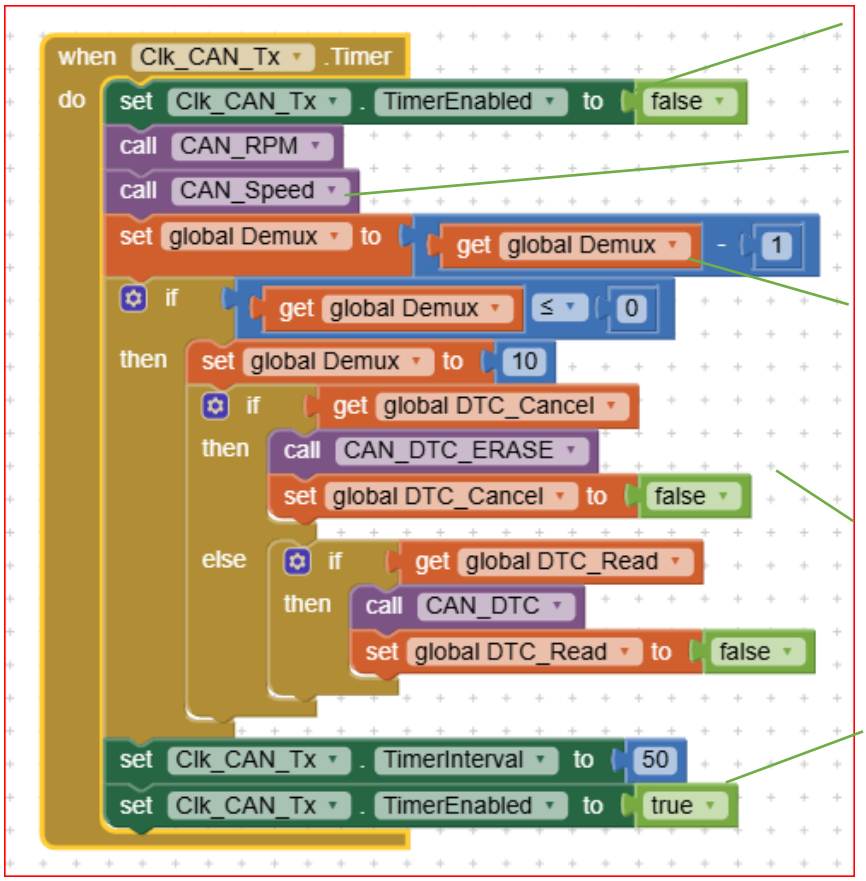
The most interesting procedure is the DTC showing and decoding from the received UDS string:

1. At fisrt it verifies if the message contains the string "NO DATA", which means that the ELM 327 has unsuccessfully accomplished the data exchange. In this case a warning message is shown in the DTC labels.
2. If the incoming message is good, then it verifies whether the message is shorter than 4 characters. If yes another warning message is shown, if longer of equal to 4, it starts to decode it.
3. If the fisrt two chars are "00" this means that no DTC are present, therefore the ERASE DTC button is left not enabled and the procedure does anything else.
4. If the first two characters are different from "00" this means that some DTCs are present (one or more), the nit starts to decode them. The DTC in the uds rules, is always composed by two bytes whose most significant bits (B7 and B6 of the high byte) represent the source of error (Engine, Body, Chassis or Unknown), while the remaining are the coded number of the error. On the web one can find the complete list of DTC's because they are standardized by the UDS protocol.
5. Ifa t least one DTC is received, then the button DTC cancel is enabled. When one presses that button a request to cancel the DTCs (service 04) is called. Please note that each request is demanded to a

clock and it is not done directly by the “button hit” event, in order to have only one procedure that sends and receives from the ELM, thus avoiding superimposition of requests.



The Tx/Rx clock procedure is here:



The clock is temporary stopped to avoid queing of multiple events

The RPM and the Speed data are requested and shown

The DTC read or erase are requested less often (1/10) than the RPM and Speed, to not overload the bus of messages exchange

The DTC read or erase are performed only if the relevant button has been hit

The Timer is reloaded with its period (50 ms) and restarted

The typical ELM327 device (Amazon):

To discover its BT address you shall plug it into the OBD port, so to give it power supply, then start the SBT app (see Note below) on your phone, or tablet and select the menu options /Terminal/Devices/ At that moment he list of availabe devices is shown and you sall select OBD2 (see below).

Tthis is the BT address that you shall set into your app

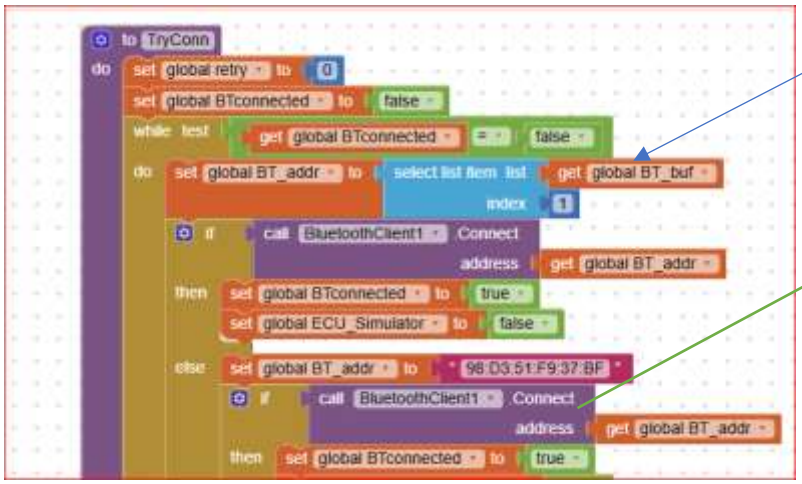
Please take care that the BT shall be the “classic” one, not the BLE. To be sure of that, it is better to choose a device featuring the BT 2.0 (which is older than the more recent 4.0 or 5.0, but for sure it isn’t BLE).

Once discovered, the BT address shall be written tinto a file called BT_Address.txt, that shall be stored into tah Application Specific Directory (ASD) of the app. It shall be structured like the following:

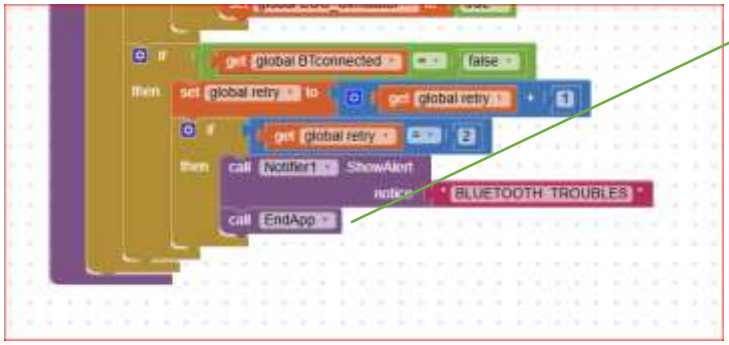
00:1D:A5:0C:51:11!
98:D3:51:F9:37:BF Simulator
Use notes: put the current BT address in the first row ending with a !
in the other rows (after the !) the spare ones or comments. The two lines above and below are not written into the file

---

Once the text read from the file is available, the BT address is fetched and stored into a global variable used by the BT client initialization



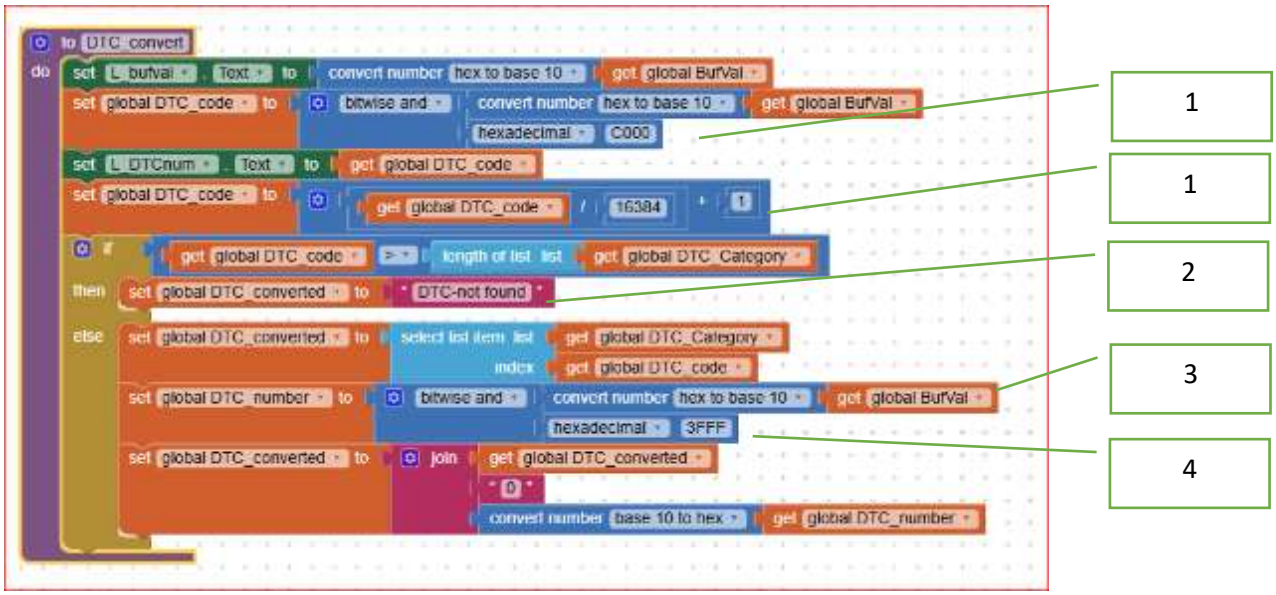
If, after two attempts, the primary address doesn't work, it tries with a simulator (in my lab an HC05 that simulates the ELM). If you don't have this on your bench, just remove these blocks.



If neither the primary neither the simulator IP address is working, it exits out of the app.

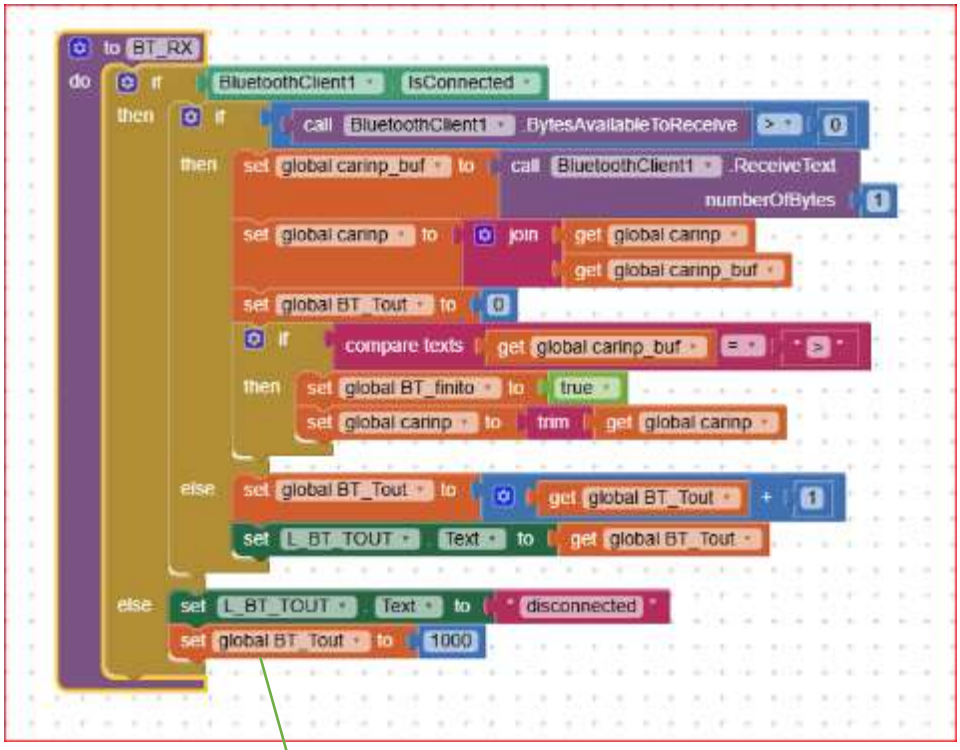
The DTC convert performs the conversion from the UDS format to a readable one.

1. The two bytes containing the DTC are masked off so to extract the two most significant bits of the high byte. The two bits, which can range from decimal 0 to 3, are used as pointer to a list from which the source of the error is extracted (see beolw the list). Since the lists in AI2 start from 1, and not 0, the extracted pointer is incremented by 1).
2. If the pointer to the list exceeds the list's length (i.e. by a misdecoding), a warning message is raised
3. If the Category of the DTC belongs to those into the list, the remaining 14 bits of the two bytes, reperesenting the DTC, are finally converted and joined, as a string, to be shown into a label.
4. This procedure plays with decimal and exadecimal numbers representations, so to make "readable" the DTC.

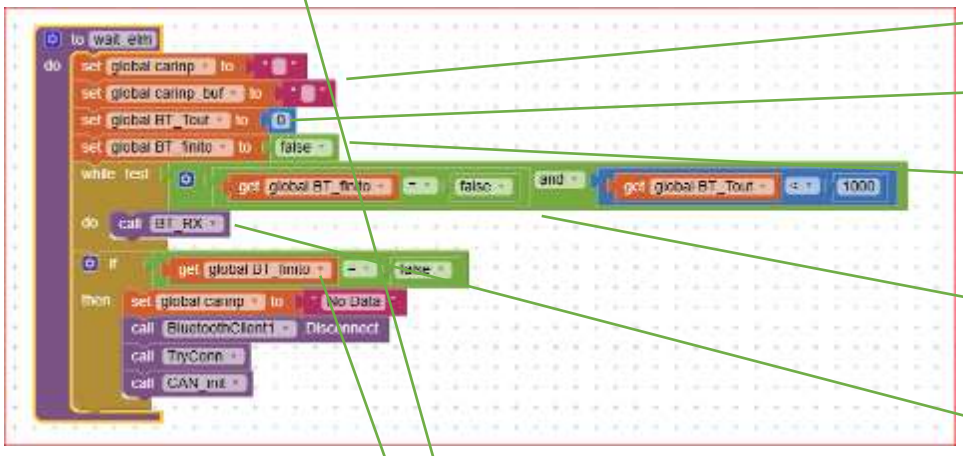


The list containing the DTC Categories

The BT receiving procedure receives one byte at a time from the ELM, until the string terminator character ">" is sent by the ELM. This character is set by default into the ELM firmware and cannot be changed, therefore the app must wait for that character so to detect a message complete. MOreover to avoid to remain stuck in waiting the >" character, a timeout is also provided: when it reaches the value 1000, the waiting is interrupted, an error mesasge is issued and a restart BT communication is perfomed (Tryconn procedure).



The following procedure waits for a complete answer from ELM (or exits for timeout)



Clears the input buffers

Resets the timeout

Presets the flag of message incomplete

Waits for a complete message or timeout

See the description above

Checks if message is complete

No message, retries the connection and resends the Start Diagnostic sequence to the car's ECU

NOTE: all the procedures exchange data by means of global varables, no local variables are used.

CREDITS:

- Anke for her MyFonts extension (for 7 segments font) based on Ken's one
- Taifun for his extensions (2) to keep the screen alive and to show the status of the battery