OBD-Auto

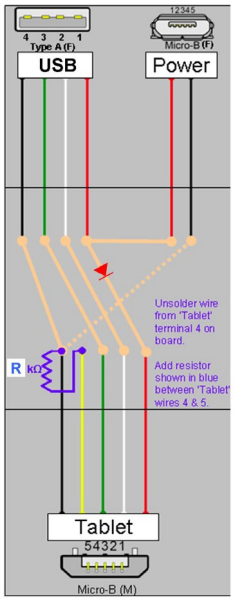
These are flows for the (free) Android Automate app, in a non-rooted device.
They are used by those who:
(i) keep an OBD reader plugged into their car OBD port, and
(ii) keep an Android host device in the car for display of selected OBD readings;
and want:
(a) automatic OBD connection and display when the car is started, with
(b) minimal drain on the car and host device batteries when the car is off.

OBD-Auto (USB) URL for download:- https://scithings.id.au/OBD-Auto(USB).flo

This is for those who keep a fast (USB) OBD reader plugged into their car OBD port.

This flow was tried in a 2019 Rav4 Hybrid, a USB OBD reader like the OBDLink SX, and an OBD app like OBDLink or Fusion, installed in a device like the Lenovo M8 tablet (Android 10). I have used the example of the free OBDLink app as an example.

OBDLink ceased manufacture of the SX reader, but the same considerations probably apply to the (more expensive) EX model that also provides for connection of proprietary Ford OBD networks. USB OBD readers take power from both the car and the device OTG port.



For sustained use, this requires either wireless charging or an OTG Y cable (for micro-USB) or hub (for USB-C) that permits simultaneous charging of the device with data exchange through the OTG port. In the case of micro-USB devices this requires a resistor of the correct value between pins 4 (ID) and 5 (Ground). Unfortunately, most device manufacturers neither follow the USB charging standard nor divulge the values they use. Cables are also a lucky-dip, but you may be able to customize some, or make your own, with a multi-meter and soldering iron. For Lenovo M8, a resistance of 15 kΩ is effective. The cable or hub should also have a diode that prevents power being drawn from the host device (this may have to be added by soldering in the junction box of some cables or hubs). Power is supplied from an ignition-switched source in the car, such as a USB port or a USB charge adaptor. Sadly, the Lenono M8 has a quirk of high battery use in docked mode (whenever this cable is engaged); and this cannot be overcome using Automate or ADB shell.

The OBD reader should have a ‘sleep’ mode that draws very little power from the car OBD port when there is no UART activity in the reader. The OBD app must be installed (with any desired enhanced PIDs), and set up for the car and manual connection to the OBD reader in use (to prevent background ‘polling’ that may keep the reader awake). Most cars keep power on to the OBD port, so ‘sleep’ of the OBD reader when ignition is off is important to minimize drain on the car battery. OBDLink readers can be set up this way using **STSL*** commands from a terminal (see below), and give a visual check by LED. Unfortunately, background ‘polling’ can not be turned off in some OBD apps.

OBDLink remembers settings between starts. It should be running (restricted) in the background, which does not need power.

The Y cable set-up is complicated because the Lenovo M8 enters a perpetual “Plugged In” (dock) mode when such a cable is attached (even when not powered), so Automate cannot use Android-reported power status as a trigger. But Android thinks the USB OBD reader is ‘disconnected’ when the car ignition is switched off, so that can be used as a trigger by Automate. The flow can be simplified for host devices that do not have this Lenovo quirk.

This Flow uses “Interact” Blocks. Touch locations may vary with devices or OBDLink versions. Use Automate gesture record. This flow also requires “Privileged access” (it requires a fresh shell command `adb tcpip 5555` from a PC after each Android 10 reboot). This can be a bit tricky to set up. Follow instructions in Automate Settings and Documentation.

Lenovo M8 incorrectly reports “Plugged In / Charging” during Y cable use with the external power off. According to Automate logs from Flows with Block “Device doze mode active?” doze does not activate unassisted in the M8 with Y cable. Therefore, I tried doze set state Blocks.

In this tablet, Automate Block “Device doze mode set state (Force activate)” works when “Plugged In”. Most devices will first require the Automate Block “Shell command privileged `dumpsys battery unplugged`” if Android thinks it is “Plugged In”. The Shell command removes the M8 false “plugged in / charging” report and enables “battery-saver” while M8 uses the Y cable, but it may cause other unwanted effects: *eg* the device can be removed for charging if required, but if `dumpsys battery unplugged` is active the battery fails to report (it still charges and discharges). Reported battery charge state and usage will be wrong unless you have sent `ADB shell dumpsys battery reset` (from automate or PC). Battery-saver is much more aggressive (effective) than doze in reducing battery use during sleep, though it may not improve much if you have manually applied battery-saving Settings in the device. I tried “Shell command privileged” Blocks.

Here is what I tried (but device battery use stays high in M8 with Y cable):

Flow beginning > When USB device attached (When changed) *{as power sensing is confused in dock mode}* >

YES > Device doze mode set state (Deactivate) *{deactivates doze}* > Shell command privileged (`dumpsys battery reset`) *{in M8 with Y cable enables the device battery gauge and disables the battery-saver}* > Keep device awake (CPU and screen & Awake screen) *{wakes the screen}* > Start app ... OBDLink *{ensures it is in the foreground}* > Delay 2s > Interact touch Click 50 88 *{connects to SX}* > Delay 1s > Interact touch Click 72 68 *{closes the enhanced network connection nag}* > Delay 8s *{while OBD modules load}* > Interact touch Click 50 32 *{goes to dashboard}* > Back to ‘When USB device attached (changed)’.

NO > Delay 1s > Interact touch Click 50 75 *{closes the disconnect nag, SX has disconnected and can sleep}* > Interact Back OBDLink *{goes to the OBDLink home screen}* > Delay 1s > Interact Home *{backgrounds the app}* > Keep device awake (Allow sleep) *{important to allow the host device screen to sleep again}* > Shell command privileged (`dumpsys battery unplugged`) *{enables battery-saver if set in device}* > Device doze mode set state (Force activate) *{maximizes device battery life}* > Back to ‘When USB device attached (changed)’.

Sadly, even using Flight Mode and everything in M8 device Settings that seemed likely to reduce battery drain when asleep, there was about 32% device battery use per day with the Y cable in (Automate `battery unplugged` and force doze Blocks did not help). This drops to <2% per day without the Y cable. Some Delay times may need to be altered for other devices or vehicles. The Flow may get confused if ignition is turned off too quickly for completion. Sometimes the OBDLink app gets confused (*eg* ‘connects’ but shows zero values for all PIDs) and has to be restarted. Many settings differ between Android devices and versions - you may need to experiment. Users with fewer constraints can simplify the flow.

OBD-Auto (BT) URL for download:- https://scithings.id.au/OBD-Auto(BT).flo

This is for those who keep a Bluetooth (BT) OBD reader plugged into their car OBD port. BT readers can be simpler to use, but they achieve lower data rates than USB OBD readers. BLE may be even slower. They can be insecure, and subject to interference from nearby BT activity.

This flow works with a car like the 2019 Rav4 Hybrid, a USB OBD reader like the OBDLink LX or MX+, and an OBD app like OBDLink, installed in a device like the Lenovo M8 tablet (Android 10). The OBDLink devices have two important features:

(1) They pair only after a physical button press (though this was not the case in some firmware versions - a serious security flaw). This is very important for security, and the pair button should only be used in a secure environment to prevent bad actors from gaining control of the car OBD capabilities. You must pair when instructed during set up. Many cheaper BT readers allow pairing continuously, and should not be left continuously in a car OBD port.

(2) They have a 'sleep' mode that draws very little power from the car OBD port when there is no UART activity. Most cars keep power on to the OBD port, so 'sleep' of the OBD reader when ignition is off is important to minimize drain on the car battery. A sleeping BT modem can not receive a UART signal. The OBDLink BT readers can be 'woken' by small pulses in the car voltage due to door opening (with a courtesy light) or starting. They can be set up this way using **STSL*** commands from a terminal (see below), and give a visual check by LED.

The OBD app must be installed (with any desired enhanced PIDs), and set up for the car and manual connection to the OBD reader in use. OBDLink remembers settings between starts. It should be running (restricted) in the background, which does not need power. Automatic connection set in the OBD app allows a slightly simpler Flow, but some apps continue background 'polling' that may keep the reader awake. The Flow below uses the Automate app to connect the OBDLink app (which does not poll in manual connect mode) to the OBD reader.

There is no need for a Y cable, as the host device OTG port can be used for charging alone (from an ignition-switched source in the car, such as a USB port or a USB charge adaptor). Therefore, power sensing is fine as a trigger. The device should enter battery-save and doze states by itself at set times after external power is off, so such Blocks are not used in the Flow.

Here is what works for me (remember to pair once on first use in the same device):

Flow beginning > When power source plugged (When changed) *{ignition on}* >

YES > Enable BT > Start app ... OBDLink *{ensures it is in the foreground}* > Delay 1s > Interact touch Click 50 88 *{connects to MX+ }* > Delay 1s > Interact touch Click 72 68 *{closes the enhanced network connection nag}* > Delay 8s *{while OBD modules load}* > Interact touch Click 50 33 *{goes to dashboard}* > Back to 'When power source plugged (changed)'.

NO > Interact Back OBDLink *{goes to the OBDLink home screen}* > Delay 1s > Interact touch Click 50 88 *{disconnects from MX+ }* > Delay 1 s > Disable BT > Interact Home *{backgrounds the app to ensure minimal power use}* > Back to 'When power source plugged (changed)'.
Any disconnect nag can be ignored, as it resolves on reconnect.

This Flow uses "Interact" Blocks but it does not require "Privileged access". Touch locations may vary with devices or OBDLink versions. Use Automate gesture record. Those who have other needs (such as BT to remain on in their Android device) can modify the Flow accordingly.

OBDLink Sleep Settings

These are relevant for both (a) car battery drain when a reader is left in the OBD port between drives and (b) Android host device battery drain while a USB reader remains connected to the USB-OTG port. They may not be correct by default. To check, you will need a terminal app. On my Android phone, I used Serial USB Terminal 1.57 by Kai Morich for a USB reader, or a simpler app like OBD Now for a paired BT reader. Serial USB Terminal may need some changes under Settings to work with OBD readers and display as expected: (Baud rate: 115200; Receive Newline: CR; Clear input on send). You can put frequently-used commands in memory buttons (or select from history). It may help to set Menu ⋮ > Data > Log (On) before use, to record the results in a file; or you can save the session later. You can also change the default save location from Android > Data > de.kai... > files, for easier access.

The terminal app must find and connect to the reader FTDI chip (via OTG-USB cable or BT). Then you can send: **ATI** to see the ELM version. **STDI** (or **AT@1**) for the device description.
ATZ if a reboot/reset of the reader is needed (*eg* after sending new parameters).
STSLCS to see current 'powersave' settings.
STSLLT to see last sleep and wake triggers (since the last reset).
STSLUIT *sec* to set UART sleep time (in seconds).
STSLUWP *min*, *max* to set UART wake pulse width in microseconds. Default is 0-0 (any UART signal >20 ns). Increase ***min*** (above 15) and reduce ***max*** (below 30000) for more noise rejection.
STSLVL *sleep*, *wakeup* to change whether the reader sleeps if the car battery drops too low (on or off; default is off, off).
STSLVLS *<|> volts|0xhhh, sec* to configure voltage level sleep trigger (default is <13.00, 3600 meaning below 13V for 3600 seconds).
STSLU *sleep, wakeup* to turn UART sleep/wakeup triggers on/off.
STSLVG *on|off* to turn voltage change wakeup trigger on/off.
STSLVGW *[+|-]volts, ms* to configure voltage change wakeup trigger (default is 0.20, 1000 meaning voltage changing by 0.2V in any direction, with one second between the samples).

If the response is the ELM version, the reader was asleep (it is woken by the first command, but does not echo or respond normally to that first command). One caution is that it will then respond to **STSLLT** with `WAKE: UART`. If ever the response is ?, the command is not recognised (it was probably a typo). If changes are made, send **ATZ** then **STSLCS** to check that they 'stuck'.

I do not understand some of the differences in sleep settings between units as supplied. After reading the FRPM, I made only those changes that seemed necessary for my use scenario.

For SX (USB) I used

```
CTRL MODE:  NATIVE
PWR_CTRL:   LOW PWR = LOW
UART SLEEP: ON,  60 s
UART WAKE:  ON,  0-0 us
EXT INPUT:  HIGH = SLEEP
EXT SLEEP:  OFF, HIGH FOR 3000 ms
EXT WAKE:   ON,  LOW FOR 2000 ms
VL SLEEP:   ON, <13.00V FOR 600 s
VL WAKE:    OFF, >13.20V FOR 1 s
VCHG WAKE:  OFF, 0.20V IN 1000 ms
```

For MX+ (BT) I used

```
CTRL MODE:  NATIVE
PWR_CTRL:   LOW PWR = LOW
UART SLEEP: ON,  60 s
UART WAKE:  ON,  0-0 us
EXT INPUT:  LOW = SLEEP
EXT SLEEP:  OFF, LOW FOR 3000 ms
EXT WAKE:   ON,  HIGH FOR 40 ms
VL SLEEP:   ON, <13.00V FOR 600 s
VL WAKE:    OFF, >13.20V FOR 1 s
VCHG WAKE:  ON, 0.20V IN 1000 ms
```

I shortened `UART SLEEP` to `60 s`. I used `VL SLEEP: ON` to protect the car battery if for some reason the OBD reader continued to draw power from the OBD port during a long period in Park. I assume this will work without USB power, if ever needed (not tested). The car system voltage should not drop below 13v for 10 min while driving (+/- my car's LFP battery).

It may be wise to check 'powersave' settings (using **STSLCS**) after any firmware update.

Lenovo M8 (Android 10) Battery Use While Screen is Asleep*

Apps (Tested in Background)	Y cable attached	% Battery used per day
OBDLink (Restricted) Automate (Unrestricted) Flow with “When changed” loop *{battery-save set in device}*	-	<2
OBDLink (Restricted)	+	32
OBDLink (Restricted) Automate (Unrestricted) No Flow	+	32
OBDLink (Restricted) Automate (Unrestricted) Flow with “When changed” loop	+	32
OBDLink (Restricted) Automate (Unrestricted) Flow with “When changed” loop Shell command privileged (`dumpsys battery unplugged`) *{battery-save set in device}* Device doze mode set state (Force activate)	+	32

* Using Flight Mode, Location Off, and any other obvious battery-saving options in Settings