How to get USB-WIFI6-5G and USB-WIFI-5G-ANT working under Olimage

Expected USB state transition

a69c:5721	>	a69c:8d80	>	a69c:8d81
Aic MSC		**AIC Wlan**		**AIC 8800D80**
storage mode		loader mode		working Wi-Fi

What the screenshots in this PDF mean

Dark terminal panels show commands to type and examples of output to expect. The exact bus/device numbers and Wi-Fi interface name can differ between boards.

Note: the Olimex product page spells the antenna variant as ‘USB-WIFI6-5G-ANT‘.

Product page:

https://www.olimex.com/Products/USB-Modules/WiFi/USB-WIFI6-5G/

Driver source used:

https://github.com/shenmintao/aic8800d80/tree/legacy-mcu1

OLIMEX kernel source used:

https://github.com/OLIMEX/linux-olimex/tree/release-20260630-v5.10.180

OLIMAGE source:

https://github.com/OLIMEX/olimage

Short summary

In simple terms, the work is:

1. Confirm the adapter is the AIC8800D80-based Olimex ‘USB-WIFI6-5G‘ or ‘USB-WIFI6-5G-ANT‘.
2. Install the normal build tools and OLIMEX kernel header packages.
3. Prepare the matching full OLIMEX kernel source tree, because the tested Olimage header package alone was not enough for DKMS external module builds.
4. Build and install the AIC8800D80 ‘legacy-mcu1‘ driver branch.
5. Reboot, replug the adapter, confirm the Wi-Fi interface appears, then connect to a Wi-Fi network with ‘nmtui‘ or ‘nmcli‘.

Scope

These steps were tested on Olimage Debian Bookworm with kernel:

TERM **Terminal command**

```
5.10.180-olimex
```

They were tested with AIC8800D80-based Olimex ‘USB-WIFI6-5G‘ / ‘USB-WIFI6-5G-ANT‘ modules that enumerate like this:

TERM **Expected terminal output**

```
Initial storage mode:      a69c:5721  aicsemi Aic MSC
Post-eject loader mode:    a69c:8d80  aicsemi AIC Wlan
Operational Linux mode:    a69c:8d81  AICSemi AIC 8800D80
Operational Windows mode:  a69c:8d83  AIC8800D80 USB WiFi
```

The tested units reported:

TERM **Expected terminal output**

```
chip_id=7, chip_mcu_id = 1
```

That MCU revision needs the ‘legacy-mcu1‘ driver/firmware branch. The normal/main firmware branch can fail with:

TERM **Expected terminal output**

```
cmd timed-out
cmd queue crashed
bin upload fail: 170400
```

Requirements

Use a root shell on the board. Internet access is needed.

Keep Ethernet connected if available, or use serial console access.

Check free disk space:

TERM **Terminal command**

```
df -h / /usr/src /tmp
```

About 4 GB free was enough for the shallow OLIMEX kernel source clone and ‘modules_prepare‘. A full kernel package build would need more.

1. Identify the USB device

Plug the adapter and check:

TERM **Terminal command**

```
lsusb
lsblk -o NAME,MODEL,SIZE,RO,TYPE
```

If it is still in storage mode, expect:

TERM **Expected terminal output**

```
lsusb: a69c:5721 aicsemi Aic MSC
lsblk: flash 3.6M RO
```

Manual eject command, if needed:

TERM **Terminal command**

```
eject /dev/sda
```

Use the device name shown by ‘lsblk‘; it may be ‘/dev/sda‘, ‘/dev/sdb‘, etc. After eject, ‘lsusb‘ should show:

TERM **Expected terminal output**

```
a69c:8d80 aicsemi AIC Wlan
```

After the driver and firmware load correctly, it should later become:

TERM **Expected terminal output**

```
a69c:8d81 AICSemi AIC 8800D80
```

2. Install build tools and OLIMEX headers

TERM **Terminal command**

```
apt update
apt install -y git dkms build-essential bc bison flex libssl-dev libelf-dev
apt install -y eject usb-modeswitch iw wireless-tools wpasupplicant
apt install -y linux-headers-olimex linux-headers-$(uname -r)
```

Check that the kernel build link exists:

TERM **Terminal command**

```
test -e /lib/modules/$(uname -r)/build/Makefile && echo headers-ok || echo missing-headers
```

On the tested image, the OLIMEX header package existed but was incomplete for DKMS external module builds, so the next section prepares the full OLIMEX kernel source tree.

3. Prepare the matching OLIMEX kernel source

Clone the kernel source that matches the installed package version:

TERM **Terminal command**

```
cd /usr/src
git clone --depth 1 --branch release-20260630-v5.10.180 https://github.com/OLIMEX/linux-olim
  ex.git linux-olimex-5.10.180
```

Copy the running kernel config:

TERM **Terminal command**

```
cp /boot/config-$(uname -r) /usr/src/linux-olimex-5.10.180/.config
```

Set the source tree to report exactly the same kernel release as the running kernel:

TERM **Terminal command**

```
cd /usr/src/linux-olimex-5.10.180
./scripts/config --set-str LOCALVERSION -olimex
./scripts/config --disable LOCALVERSION_AUTO
printf '#!/bin/sh\necho -olimex\n' > scripts/setlocalversion
chmod +x scripts/setlocalversion
```

Prepare it for external modules:

TERM **Terminal command**

```
make ARCH=arm olddefconfig modules_prepare
```

Verify the prepared release string:

TERM **Terminal command**

```
cat /usr/src/linux-olimex-5.10.180/include/config/kernel.release
```

Expected:

TERM **Expected terminal output**

```
5.10.180-olimex
```

Point the kernel module build link at this prepared source tree:

TERM **Terminal command**

```
BUILD_LINK=/lib/modules/$(uname -r)/build
if [ -L "$BUILD_LINK" ]; then
    rm "$BUILD_LINK"
else
    echo "$BUILD_LINK is not a symlink; stop and inspect it"
    exit 1
fi
ln -s /usr/src/linux-olimex-5.10.180 "$BUILD_LINK"
readlink -f /lib/modules/$(uname -r)/build
```

Expected:

TERM **Expected terminal output**

```
/usr/src/linux-olimex-5.10.180
```

4. Install the AIC8800D80 legacy MCU driver

Clone the legacy branch:

TERM **Terminal command**

```
cd /root
git clone --branch legacy-mcu1 https://github.com/shenmintao/aic8800d80.git aic8800d80-legac
  y-mcu1
cd /root/aic8800d80-legacy-mcu1
```

Remove any stale DKMS entry:

TERM **Terminal command**

```
dkms remove -m aic8800 -v 1.0.0 --all
```

It is OK if this says the module is not in the DKMS tree.

Run the installer:

TERM **Terminal command**

```
bash ./install.sh
```

Expected good signs:

TERM **Expected terminal output**

```
Module built successfully.
Module installed via DKMS.
DKMS module registered: aic8800/1.0.0, 5.10.180-olimex, armv7l: installed
```

If ‘update-initramfs‘ fails, that warning can usually be ignored for this USB Wi-Fi use case.
The important part is that DKMS builds and installs the modules.

5. Reboot and replug

Unplug the USB Wi-Fi adapter.

TERM **Terminal command**

```
sync
reboot
```

After the board boots, log in again and plug the adapter back in. Wait about 10 seconds.

6. Verify operation

Check USB state:

TERM **Terminal command**

```
lsusb
```

Expected final state:

TERM **Expected terminal output**

```
a69c:8d81 AICSemi AIC 8800D80
```

Check network interfaces:

TERM **Terminal command**

```
ip link
```

Expected: a new Wi-Fi interface named like:

TERM **Expected terminal output**

```
wlx001d43100a9d
```

The exact name depends on the adapter MAC address.

Set a variable for the interface:

TERM **Terminal command**

```
IFACE=$(iw dev | awk '$1=="Interface"{print $2; exit}')
echo "$IFACE"
```

Scan for networks:

TERM **Terminal command**

```
iw dev "$IFACE" scan | grep SSID
```

Expected: nearby 2.4 GHz and 5 GHz SSIDs are listed.

7. Connect to a Wi-Fi network

TERM **Generated nmtui-style screen**

```
+-- NetworkManager TUI ----------------------------------+
| Activate a connection                                  |
|                                                        |
|   Wired                                               |
|   Wi-Fi                                               |
|   > OlimexTest5G                        Signal: ****  |
|     OlimexAP                             Signal: ***  |
|     OlimexGuest-5GHz                     Signal: **   |
|                                                        |
|                   <Activate>     <Back>                |
+--------------------------------------------------------+
```

The easiest interactive method is ‘nmtui‘:

TERM **Terminal command**

```
nmtui
```

In the text UI:

1. Select ‘Activate a connection‘.
2. Select the Wi-Fi network SSID.
3. Enter the Wi-Fi password.
4. Select ‘OK‘ or ‘Activate‘.
5. Exit ‘nmtui‘.

If ‘nmtui‘ is not installed:

TERM **Terminal command**

```
apt install -y network-manager
systemctl enable --now NetworkManager
```

If you prefer ‘nmcli‘, connect like this:

TERM **Terminal command**

```
nmcli device wifi list
nmcli device wifi connect "SSID_NAME" password "PASSWORD" ifname "$IFACE"
```

After connecting with either method, verify Internet access:

TERM **Terminal command**

```
ping -c 4 8.8.8.8
ping -c 4 google.com
```

Troubleshooting

If the adapter stays as storage mode:

TERM **Expected terminal output**

```
a69c:5721 aicsemi Aic MSC
```

then eject it manually:

TERM **Terminal command**

```
lsblk -o NAME,MODEL,SIZE,RO,TYPE
eject /dev/sda
```

Use the correct ‘/dev/sdX‘ from ‘lsblk‘.

The installer also installs a udev rule at:

TERM **Expected terminal output**

```
/usr/lib/udev/rules.d/aic.rules
```

This rule auto-ejects ‘a69c:5721‘, so after installation the device may appear to be "always ejected". That is normal.

If the driver builds but ‘modprobe‘ says:

TERM **Expected terminal output**

```
version magic ’5.10.180+ ...’ should be ’5.10.180-olimex ...’
```

then the kernel source tree release string is wrong. Recheck:

TERM **Terminal command**

```
cat /usr/src/linux-olimex-5.10.180/include/config/kernel.release
```

It must be exactly:

TERM **Expected terminal output**

```
5.10.180-olimex
```

Then remove and reinstall the DKMS module:

TERM **Terminal command**

```
dkms remove -m aic8800 -v 1.0.0 --all
cd /root/aic8800d80-legacy-mcu1
bash ./install.sh
```

If the main/new firmware branch was used by mistake, logs may show:

TERM **Expected terminal output**

```
chip_id=7, chip_mcu_id = 1
cmd timed-out
cmd queue crashed
bin upload fail: 170400
```

Use the ‘legacy-mcu1‘ branch instead.

To inspect relevant logs:

TERM **Terminal command**

```
dmesg | grep -Ei 'aic|8800|8d80|8d81|8d83|firmware|wlan|chip_id|chip_mcu|cmd timed|queue cra
  shed|bin upload'
```

If copy/paste into the serial terminal inserts text like ‘^[[200~‘, type complex commands manually or disable bracketed paste in the terminal before continuing.