

# Zhihe series LTE dongles (generic-zhihe)

## Contributors

- notfound405

## Users owning this device

- Adomerle (Notes: UF896)
- CodeDaraW (Notes: UFI003\_MB\_V02 & UFI103S\_V05)
- Craftyguy (Notes: UZ801 v3.0)
- Jlaxyga (Notes: FY\_UZ801\_V3.2 4Gb)
- Notfound405 (Notes: UZ801 v3.0 & UFI001C\_MB\_V01 & UFI003\_MB\_V02)
- Taijigong (Notes: UF896 V 1.1)
- TravMurav (Notes: uf896)

## Naming

The name zhihe comes from an early online shop selling these devices. The shop had been gone for a long time but the name is widely used to refer to these devices now. It's spelled in Chinese "正荷", also usually misspelled as "纸盒", both pronounced in pinyin "zhihe".

## How to enter flash mode

```
$ adb reboot edl # Qualcomm EDL, or
$ adb reboot bootloader # fastboot.
```

Note: only applies to stock Android system.

### UFI-001C and many more with xinxun brand

By default, we get only shell permission with adb shell. However, you don't have to install SuperSU or Magisk for temporary root access. The following instruction is much easier for temporary root access:

```
$ adb shell
$ # Now you are in adb
$ setprop service.adb.root 1; busybox killall addb
$ # wait for a few seconds
$ adb shell
$ # Now you get the root permission!
```

### UZ801 V3.0

adb interface is not enabled by default. To enable adb access and root permission, plug this device into your PC and use browser to access this URL: [http://192.168.100.1/usb\\_debug.html](http://192.168.100.1/usb_debug.html), and then reboot your device(replug it). You'll find an adb device and get root permission with adb shell. See the section below if your device bricks.

## Debug UART pin and test pins

### UZ801 V3.0



Note: voltage: 1.8V

### UFI103S, UFI001C(B), UFI003 etc.

They are clearly labeled on the board, the only button is the EDL button.

Note: voltage: 3-3V

## Preparation

It is recommended to backup the eMMC before flashing. Refer to EDL (<https://github.com/bkerler/EDL>). The original Android image contains too old firmware to boot mainline kernel, use firmwares from Dragon Board 410c as replacement. Also, you may need qhystub as well as lk1st (lk2nd is not recommended since vendor about firmwares vary and have different strange quirks which causes a lot of trouble).

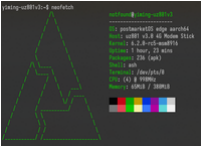
```
$ # Please do NOT use a later release from Linaro, the tz firmware becomes too large to flash to tz partition. Just use the link below.
$ wget https://releases.linaro.org/96boards/dragonboard410c/linaro/rescue/17.09/dragonboard410c_bootloader_emmc_android-88.zip
$ unzip dragonboard410c_bootloader_emmc_android-88.zip
$ # put your device into EDL mode or fastboot mode, flash sb1(only required for uz801) and tz with the corresponding files extracted.
$ # That is, the minimum firmwares needed to be replaced are: for uz801, both sb1 and tz; for other models, only tz.
$ # It is also required to flash hyp(see below). qhystub is strongly recommended as a replacement instead of the hyp firmware from db410c because it supports booting linux in EL2 so that KVM can be enabled. With hyp downloaded from Linaro, Linux can only boot in EL1 so KVM is not accessible. Although qhystub is unofficial and not qualified by Qualcomm or Linaro so it's up to you using qhystub or the official hyp firmware extracted from db410c.
$ # Either one of them is okay and enough to boot mainline Linux kernel. However, mixing the use of tz from Linaro and stock hyp is not supported and causes device brick. Don't reboot the device unless both hyp and tz are replaced!
$ # Note: Flashing other images such as rpm/sbl1 is not recommended anymore. It is causing problems for some other models, better to keep them untouched.
```

For qhystub and lk1st:

```
$ git clone https://github.com/msm8916-mainline/qhystub.git
$ cd qhystub; git clone https://github.com/msm8916-mainline/qtestsign.git
$ make CROSS_COMPILE=aarch64-linux-gnu-
$ # get the generated qhystub-test-signed.mbn and store it somewhere
$
$ cd ..; git clone https://github.com/msm8916-mainline/lk2nd.git
$ # The command below assumes Ubuntu as the host OS and UZ801 as the device.
$ # You may also need to install some additional packages with a package manager.
$ # Refer to the README.md of the repository for detailed instructions.
$ cd lk2nd; make LK1ST_DTB=msm8916-512mb-mtp LK1ST_COMPATIBLE=zhihe,uz801-v3 TOOLCHAIN_PREFIX=arm-none-eabi- lk1st=msm8916
$ ../qhystub/qtestsign/qtestsign.py aboot build-lk1st-msm8916/emmc_appsboot.mbn
$ # get build-lk1st-msm8916/emmc_appsboot-test-signed.mbn and store it somewhere
$
$ # Now flash them to hyp and aboot partitions with EDL or fastboot.
```

Sadly, with the modifications above, it's impossible to boot original Android system anymore if you do not restore the original firmwares back. Say goodbye to the stock Android system, and do what you think necessary before proceeding, like modifying IMEI, switching SIM card slot and so on, since it's difficult to perform the tasks in postmarketOS.

## Generic Zhihe-series LTE dongles



ssh with

neofetch

Manufacturer	Generic
Name	Zhihe-series LTE dongles
Codename	generic-zhihe
Released	2019-2023
Category	testing
Pre-built images	no
Original software	Android
Original version	4.3
Hardware	
Chipset	Qualcomm Snapdragon 410 (MSM8916)
CPU	4x 1.0 GHz Cortex-A53
GPU	Adreno 305
Display	none
Storage	4GB(common)/8GB(rare) eMMC
Memory	512MB
Architecture	AArch64

Features	
USB Networking	Works
Flashing	Works
WiFi	Works
Mainline	Works
3D Acceleration	
Bluetooth	
GPS	Works
Mobile data	Works
Internal storage	
SMS	
Calls	
USB OTG	Partial

Misc	
Memory Card	
FOSS bootloader	Works

# Installation

See [Qualcomm\\_Snapdragon\\_410/412\\_\(MSM8916\)#Installation\\_using\\_pmbootstrap](#).

## partition usage strategy

There are several strategies to make full use of the limited eMMC storage. They are listed below, feel free to add more.

### Some common recommendations

1. use btrfs for rootfs

```
$ pmbootstrap install --filesystem btrfs
```

2. reduce the size of /boot partition to 128 MB or even smaller (64MB should be enough to hold the contents, but a slightly larger size is recommended)

```
$ pmbootstrap config boot_size 128
```

### Examples

1. Flash the rootfs and /boot to system partition (not recommended)

```
$ pmbootstrap flasher flash_rootfs
```

The volume of system partiton is too low to hold a rootfs. It works, but the free space is very low, making it painful to use.

2. Flash the rootfs and /boot to userdata partition (the easiest)

```
$ pmbootstrap flasher flash_rootfs --partition userdata
```

This one is the easiest, and the volume of userdata partition should be sufficient for common uses. The drawback is that a large amount of storage space is not used.

3. split /boot and rootfs, and flash them to different partitions.

```
$ pmbootstrap install --split
$ pmbootstrap export
$ cd /tmp/postmarketOS-export/
$ img2simg *-root.img root.simg
$ # Put your device into fastboot mode
$ fastboot flash cache *-boot.img
$ fastboot flash system root.simg # or userdata
```

The operation is a bit complex, but you can make use of 2 different partitions, which reduces space wasting. It is recommended to flash /boot partition to cache partition and flash rootfs to system or userdata.

Drawback: not officially supported.

Note: you may mount the other partition (usually system or userdata, if you flash /boot to cache partition) to, e.g. /var/lib/docker with /etc/fstab and localmount, so that every large partitions can be used.

4. repartition

There are two major choices, one is to replace the entire partition table and use your custom one, the other is to do minor modification with gptfdisk, leaving most partitions untouched.

1. replace the entire partition table

You can do it with (s)gdisk in linux, fastboot flash partition in fastboot or edl tool in Qualcomm EDL mode.

No technical support here. It's dangerous and may brick your device, think twice before proceeding and be prepared to recovery your device in Qualcomm EDL mode.

2. gptfdisk

This method is much safer than the previous one and probably will at least let the device boot to fastboot mode if you did something wrong. Generally, the easiest solution would be combining system and userdata together. You can do it with gdisk, sgdisk or any tools you like. Just delete all partitions after system and extend system to maximum.

Doing this in pmOS may be dangerous and sometimes impossible(the rootfs is mounted and active), so you can try do it from [Debug shell / netboot](#) or [\[MSM8916 fork of Jumpdrive \(https://github.com/dreemurrs-embedded/Jumpdrive/pull/84\)\]](#).

Note: persist partition sits between system and userdata, and unfortunately WCNSS nv items file is stored in it, which need to be loaded during runtime for functionality of WCN36xx(WiFi). You have to backup and relocate persist partition before deleting it, or if you like, extract WCNSS\_qcom\_nv\_items.bin and place it at /lib/firmware.

## See also

- [pmaports!3988](https://gitlab.com/postmarketOS/pmaports/-/merge_requests/3988) ([https://gitlab.com/postmarketOS/pmaports/-/merge\\_requests/3988](https://gitlab.com/postmarketOS/pmaports/-/merge_requests/3988)) Initial merge request
- [Device package](https://gitlab.com/postmarketOS/pmaports/-/tree/master/device/testing/device-yiming-uz801) (<https://gitlab.com/postmarketOS/pmaports/-/tree/master/device/testing/device-yiming-uz801>)
- [Kernel package](https://gitlab.com/postmarketOS/pmaports/-/tree/master/device/community/linux-postmarketos-qcom-msm8916) (<https://gitlab.com/postmarketOS/pmaports/-/tree/master/device/community/linux-postmarketos-qcom-msm8916>)

Retrieved from "https://wiki.postmarketos.org/index.php?title=Zhihe\_series\_LTE\_dongles\_(generic-zhihe)&oldid=57921"