1. OpenWrt version

The version name is Barrier Breaker, and the version number is 14.07.

2. OpenWRT compilation environment

The virtual machine is compiled on Baidu Cloud (using VMware virtual machine):

Link: https://pan.baidu.com/s/1Ijqsl9xn-6zRkM_LKq22RA Extraction code: gpxg This is the VMware disk image. Virtual machine username: hlk Password: 12345678

Link: https://pan.baidu.com/s/1RqU9sRh2iZZ-Y3WNPFypdg Extraction code: 55sy This is the VirtualBox disk image. Virtual machine username: hlk Password: 12345678

After decompressing the virtual machine image, you need to import it using VirtualBox. The VirtualBox software...

You can download it from https://www.virtualbox.org/.

The virtual machine comes with the OpenWRT 14.07 SDK source code, located at /home/hlk/mtkopenwrt, and already has a default configuration. The settings can be compiled directly.

SDK source code package link: https://pan.baidu.com/s/1X4Om05zAaNAOurfiGuUPxA Extraction code: unhb

3. OpenWRT Configuration and Compilation

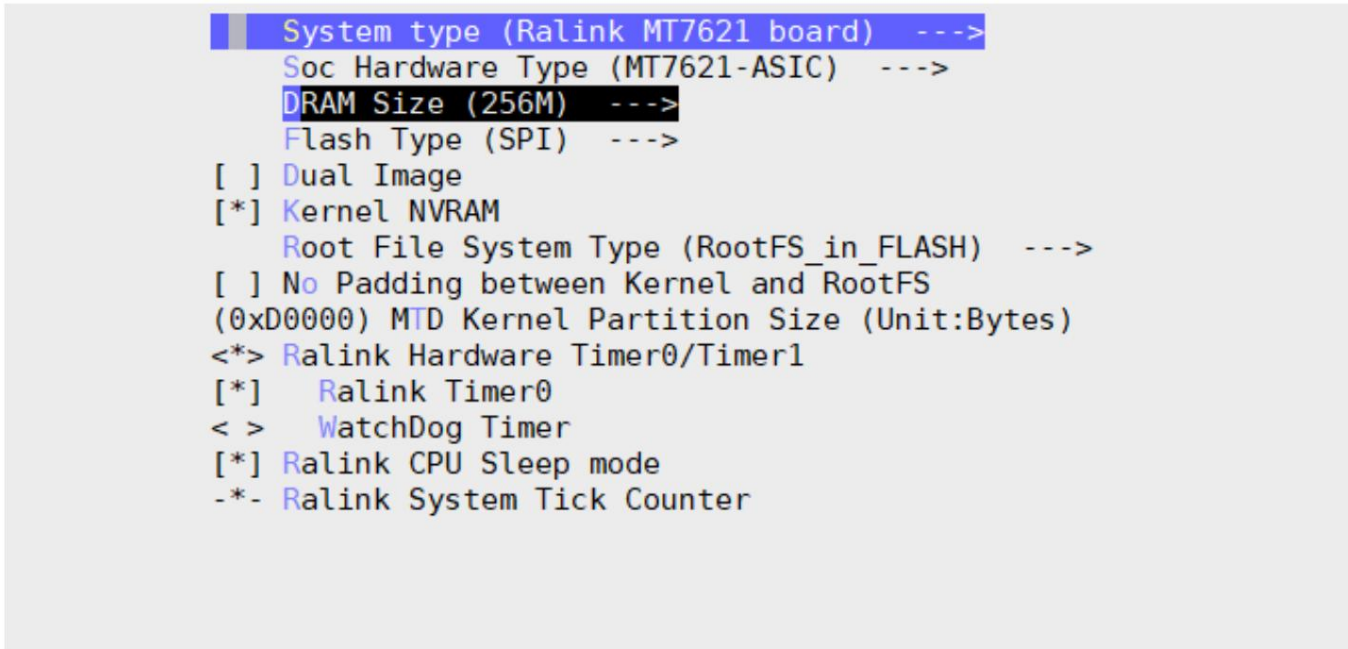
The SDK already includes a default configuration that meets the basic routing functions. Customers can also customize the configuration according to their own needs.

Custom configuration

Command: make menuconfig

Modify the default memory size

configuration: `make kernel_menuconfig -> Machine selection ---> DRAM Size (256M) --->`



The command `make V=99`

compiles the firmware, which is then saved in the `bin/

ramips/` directory. The generated firmware name is: `openwrt-ramips-mt7621-mt7621-squashfs-sysupgrade.bin`.

4. Use the reg command to control the registers of the 7621.

5. How to restore OpenWrt to factory settings

Enter the following in the command line:

umount /dev/mtdblock6; firstboot

In the first boot menu, type Y to confirm restoring default settings.

OpenWrt will have its existing configuration information cleared, restoring it to its default factory settings.

6. Network port configuration in OpenWRT

7. Add your own application to OpenWRT and compile it into the firmware.

The following example uses "Hello World" to illustrate how to add an application in OpenWRT.

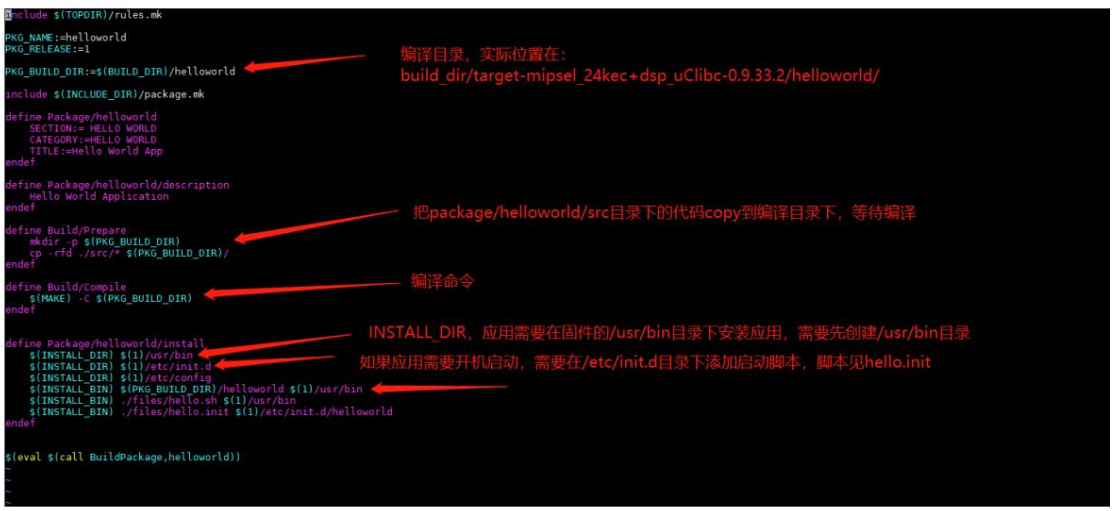
Create a directory named helloworld under the package directory.

Create a Makefile file in the helloworld directory: the file is attached.

Makefile

For a detailed description of Makefile rules, please refer to the following documentation:

https://openwrt.org/docs/guide-developer/packages



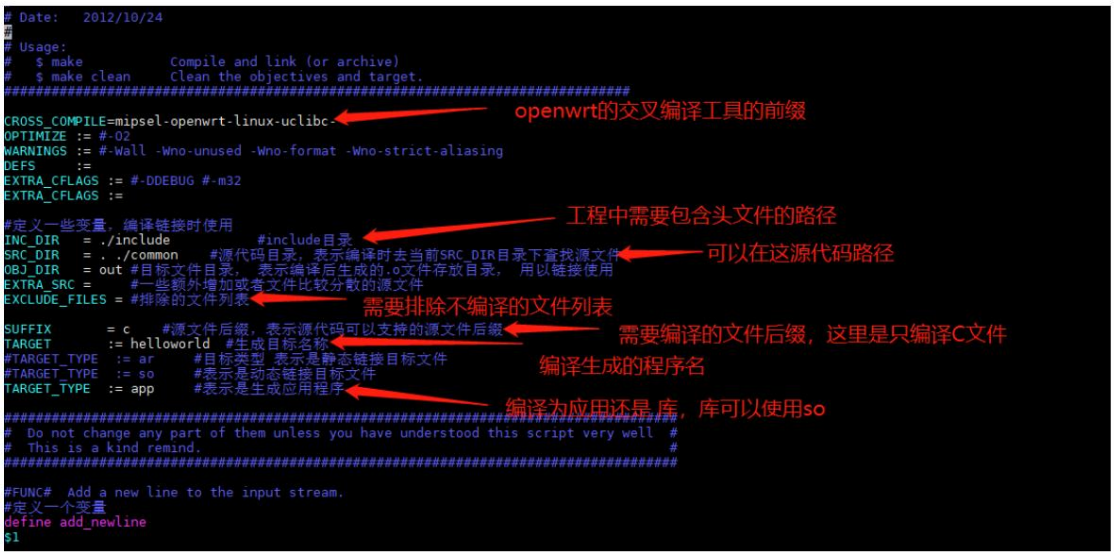
helloworld.tar.gz

See the attached package for sample code:

Extract directly to the OpenWRT package directory.

The `src` directory under the `Helloworld` directory is where the source code is stored. In the example, the `Makefile` in the `src` directory is the actual Makefile used to compile the code. Customers can also write their own Makefiles according to their needs. A few points to note: a. Cross-compilers whose OpenWRT compilation tool names begin with `mipsel-openwrt-linux-uclibc-` cannot be directly linked using `mipsel-openwrt-linux-uclibc-ld`.

A. If you use the Makefile provided in the example, it will work for simple projects. It can iterate through all the .c files in the src directory and compile them. The Makefile description is as follows:



After the file is created, use `make menuconfig` to configure OpenWRT:

After selecting, navigate to the project directory and run `make`. After compilation, upgrade the firmware.

Compile-time library dependency

issues: If you are using the pthread multithreading

library, you can add the following to the outer Makefile:

```
include $(TOPDIR)/rules.mk

PKG_NAME:=helloworld
PKG_RELEASE:=1

PKG_BUILD_DIR:=$(BUILD_DIR)/helloworld

include $(INCLUDE_DIR)/package.mk

define Package/helloworld
    SECTION:= HELLO WORLD
    CATEGORY:=HELLO WORLD
    TITLE:=Hello World App
    DEPENDS := +libpthread
endef

define Package/helloworld/description
    Hello World Application
endef

define Build/Prepare
    mkdir -p $(PKG_BUILD_DIR)
    cp -rfd ./src/* $(PKG_BUILD_DIR)/
endef

define Build/Compile
    $(MAKE) -C $(PKG_BUILD_DIR)
endef


define Package/helloworld/install
    $(INSTALL_DIR) $(1)/usr/bin
    $(INSTALL_DIR) $(1)/etc/init.d
    $(INSTALL_DIR) $(1)/etc/config
    $(INSTALL_BIN) $(PKG_BUILD_DIR)/helloworld $(1)/usr/bin
    $(INSTALL_BIN) ./files/hello.sh $(1)/usr/bin
    $(INSTALL_BIN) ./files/hello.init $(1)/etc/init.d/helloworld
endef


$(eval $(call BuildPackage,helloworld))
~
~
```

Another method is to trick the OpenWRT compilation process by adding the following to the

Makefile:

```
define Package/helloworld/extra_provides
    echo "libpthread.so.0"
```

This way, compilation won't result in errors due to missing libraries. However, the application still won't run; the corresponding libraries need to be included.

You need to copy it to the system library directory.

Power-on startup script:

OpenWrt executes the scripts under /etc/init.d upon power-on and executes them sequentially according to the size of the START variable in the script.

hello.init

For instructions on writing the startup script, please refer to:

```
luke@ub:~/mtkopenwrt/package/helloworld/files$ cat hello.init
#!/bin/sh /etc/rc.common

START=99

start() {
        echo "start helloworld" >> /tmp/helloworld
}

stop() {
        echo "stop helloworld" >> /tmp/helloworld
}
luke@ub:~/mtkopenwrt/package/helloworld/files$
```

启动数字，数字越大越晚启动

启动时执行的shell函数名字

执行/etc/init.d/helloworld stop 调用此函数脚本 停止应用执行

8. Controlling GPIO

The GPIO mode register GPIO_MODE for the M6721 pins can be

found on page 32 of the datasheet. The pins can be controlled to be used as GPIOs by controlling the corresponding bits in the register.

1E000060 GPIO_MODE GPIO purpose selection

Bit	31	30	29	28	27	26	25	24	23	22	21	20
Name												ESWINT_MODE
Type												RW
Reset												0
Bit	15	14	13	12	11	10	9	8	7	6	5	4
Name	RGMII2_MODE	RGMII1_MODE	MDIO_MODE		PERST_MODE		WDT_MODE		JTAG_MODE	UART2_MODE		UA
Type	RW	RW	RW		RW		RW		RW	RW		
Reset	1	1	0	1	0	1	0	0	0	0	1	0

Using the RGMII2 pins as an example, page 66

of the datasheet shows that the RGMII2_MODE register can actually control 12 pins.

PAD_G1_RXD2	g1_rxd[2] (I/O)	gpio (I/O)		
PAD_G1_RXD3	g1_rxd[3] (I/O)	gpio (I/O)		
PAD_G1_RXDV	g1_rxdv (I/O)	gpio (I/O)		
PAD_G1_RXC	g1_rxc (I/O)	gpio (I/O)		
PAD_G2_TXD0	g2_txd[0] (I/O)	gpio (I/O)		
PAD_G2_TXD1	g2_txd[1] (I/O)	gpio (I/O)		
PAD_G2_TXD2	g2_txd[2] (I/O)	gpio (I/O)		
PAD_G2_TXD3	g2_txd[3] (I/O)	gpio (I/O)		
PAD_G2_TXEN	g2_txen (I/O)	gpio (I/O)		
PAD_G2_TXC	g2_txc (I/O)	gpio (I/O)		
PAD_G2_RXD0	g2_rxd[0] (I/O)	gpio (I/O)		
PAD_G2_RXD1	g2_rxd[1] (I/O)	gpio (I/O)		
PAD_G2_RXD2	g2_rxd[2] (I/O)	gpio (I/O)		
PAD_G2_RXD3	g2_rxd[3] (I/O)	gpio (I/O)		
PAD_G2_RXDV	g2_rxdv (I/O)	gpio (I/O)		
PAD_G2_RXC	g2_rxc (I/O)	gpio (I/O)		
PAD_SPI_CS0_N	spi_cs0 (I/O)	gpio (I/O)	nd_cs_n (O)	
PAD_SPI_CS1_N	spi_cs1 (I/O)	gpio (I/O)	nd_we_n (O)	
PAD_SPI_SCLK	spi_clk (I/O)	gpio (I/O)	nd_re_n (O)	

When bit 15 RGMII2_MODE in the GPIO_MODE register is set to 1, the aforementioned 12 pins can be used as GPIO pins.

15 RGMII2_MODE **RGMII2 GPIO mode**

1: GPIO
0: RGMII2

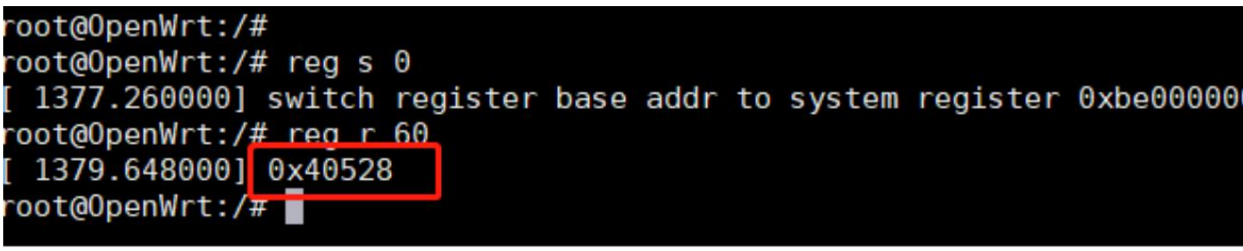
Control methods:

a. Use the reg command for control. i.

Configure the pin to GPIO mode. Log in to the

serial terminal and use `reg s 0; reg r 60` in the command line to read the value of the GPIO_MODE register. The firmware

defaults to RGMII2 configuration in RGMII2 mode, with bit 15 set to 0.



Set bit 15 to 1 using the

command: `reg w 60 0x48258`. After execution,

use `reg r 60` to check if the configuration was successful. ii.

Configure GPIO input/output modes.

After configuring the pin to GPIO mode, you also need to specify whether the GPIO is an input or

output. The MT7621 and MT7688 use the same configuration register; refer to section 5.8, General Purpose IO, in the MT7688 datasheet.

The GPIO control register is shown in the following figure:

GPIO_CTRL_X: This is the GPIO direction control register, where X = 0/1/2. Each register controls 32 GPIOs.

GPIO_POL_X: This is the GPIO polarity control register X. (Same as above)

GPIO_DATA_X: This is the GPIO data register. You can read this register to get the current GPIO status. value

GPIO_DSET_X: This is the GPIO setting register, used to control whether the value of the corresponding GPIO is 1. Setting it to 1 enables control.

GPIO_DCLR_X: This is the GPIO clear register, used to control whether the value of the corresponding GPIO is 0. Setting it to 1 enables clearing.

Address	Name	Width	Register Function
10000600	GPIO_CTRL_0	32	**GPIO0 to GPIO31 direction control register** These direction control registers are used to select the data direction of the GPIO pin. The value driven onto the GPIO pins, are controlled by the GPIO_POL_x, and GPIO_DATA_x registers.
10000604	GPIO_CTRL_1	32	**GPIO32 to GPIO63 direction control register** These direction control registers are used to select the data direction of the GPIO pin. The value driven onto the GPIO pins, are controlled by the GPIO_POL_x, and GPIO_DATA_x registers.
10000608	GPIO_CTRL_2	32	**GPIO64 to GPIO95 direction control register** These direction control registers are used to select the data direction of the GPIO pin. The value driven onto the GPIO pins, are controlled by the GPIO_POL_x, and GPIO_DATA_x registers.
10000610	GPIO_POL_0	32	**GPIO0 to GPIO31 polarity control register** These polarity control registers are used to control the polarity of the data is driven on or read from the GPIO pin.
10000614	GPIO_POL_1	32	**GPIO32 to GPIO63 polarity control register** These polarity control registers are used to control the polarity of the data is driven on or read from the GPIO pin.
10000618	GPIO_POL_2	32	**GPIO64 to GPIO95 polarity control register** These polarity control registers are used to control the polarity of the data is driven on or read from the GPIO pin.
10000620	GPIO_DATA_0	32	**GPIO0 to GPIO31 data register** These data registers store current GPIO data value for GPIO input mode, or output driven value for GPIO output mode. Bit position stand for correspondent GPIO pin.
10000624	GPIO_DATA_1	32	**GPIO32 to GPIO63 data register** These data registers store current GPIO data value for GPIO input mode, or output driven value for GPIO output mode. Bit position stand for correspondent GPIO pin.
10000628	GPIO_DATA_2	32	**GPIO64 to GPIO95 data register** These data registers store current GPIO data value for GPIO input mode, or output driven value for GPIO output mode. Bit position stand for correspondent GPIO pin.
10000630	GPIO_DSET_0	32	**GPIO0 to GPIO31 data set register** These data set registers are used to set bits in the GPIO_DATA_x registers.
10000634	GPIO_DSET_1	32	**GPIO32 to GPIO63 data set register** These data set registers are used to set bits in the GPIO_DATA_x registers.
10000638	GPIO_DSET_2	32	**GPIO64 to GPIO95 data set register** These data set registers are used to set bits in the GPIO_DATA_x registers.
10000640	GPIO_DCLR_0	32	**GPIO0 to GPIO31 data clear register** These data set registers are used to clear bits in the GPIO_DATA_x registers.
10000644	GPIO_DCLR_1	32	**GPIO32 to GPIO63 data clear register** These data set registers are used to clear bits in the

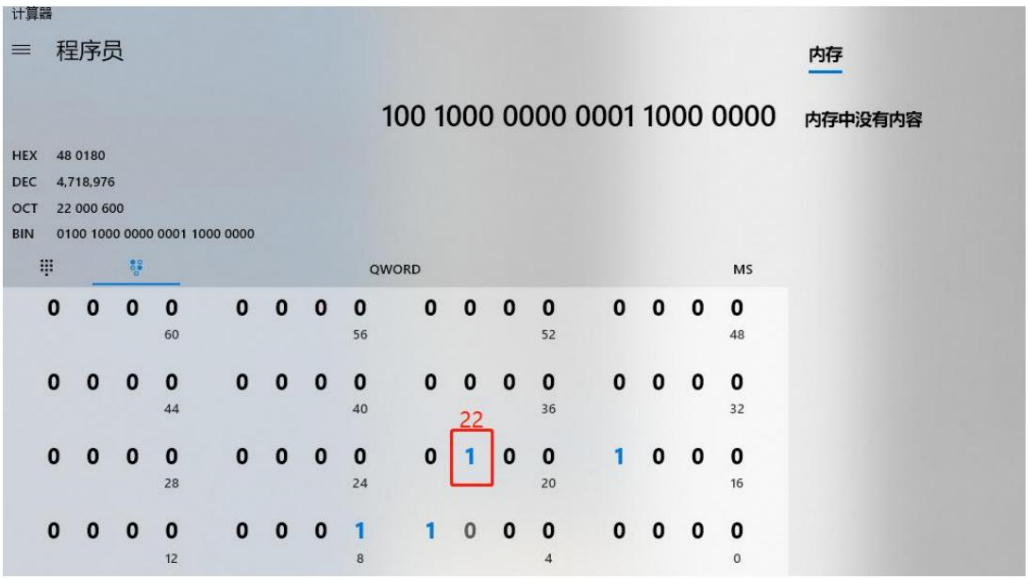
Example:

Control GPIO22 as a GPIO output pin. GPIO22 is in the range of 0~31, using the registers of GPIO_XXX_0.

Register for control

Reading the GPIO_CTRL_0 register: reg r 600

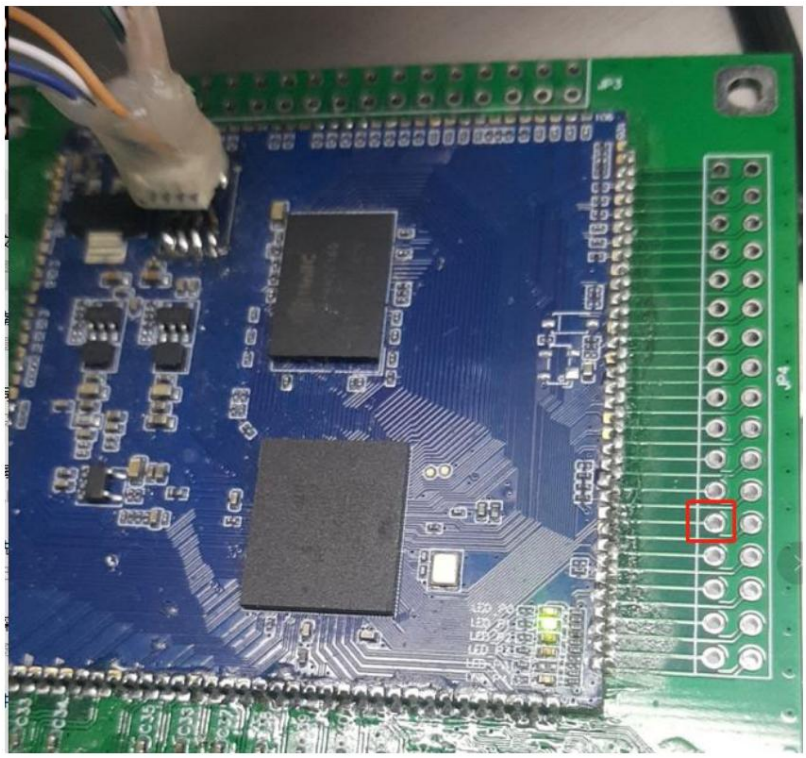
```
root@OpenWrt:/# reg r 600
[ 2949.486000] 0x480180
root@OpenWrt:/#
```



If bit 22 is 1, it means that GPIO22 is set to GPIO output mode. The HLK7621 user manual indicates it is pin 80.

80	GE_TXD0	RGMII2 Tx D

Location on the development board:



Set to high level: reg w 630 400000 Set to low level: reg w 640 400000

Demo video:

16dbdfcab79271f5ff7beabd04e27f92.mp4

Other GPIO control methods are similar to those of RGMII2.

9. SSH login issues

MTKOpenWRT uses Dropbear as its default SSH connection, which does not allow passwordless login. 1. You can modify the `package/base-files/files/etc/shadow` file and replace the `shadow` file containing the root password.

2. After starting the application, use `passwd` to set the password. Once the password is set, you can log in via SSH.

10. VLAN Configuration

11. Network Port Configuration

A.Single WAN port

B. Multiple **WAN** ports

C. No **WAN** port

D. Multi-bridge configuration

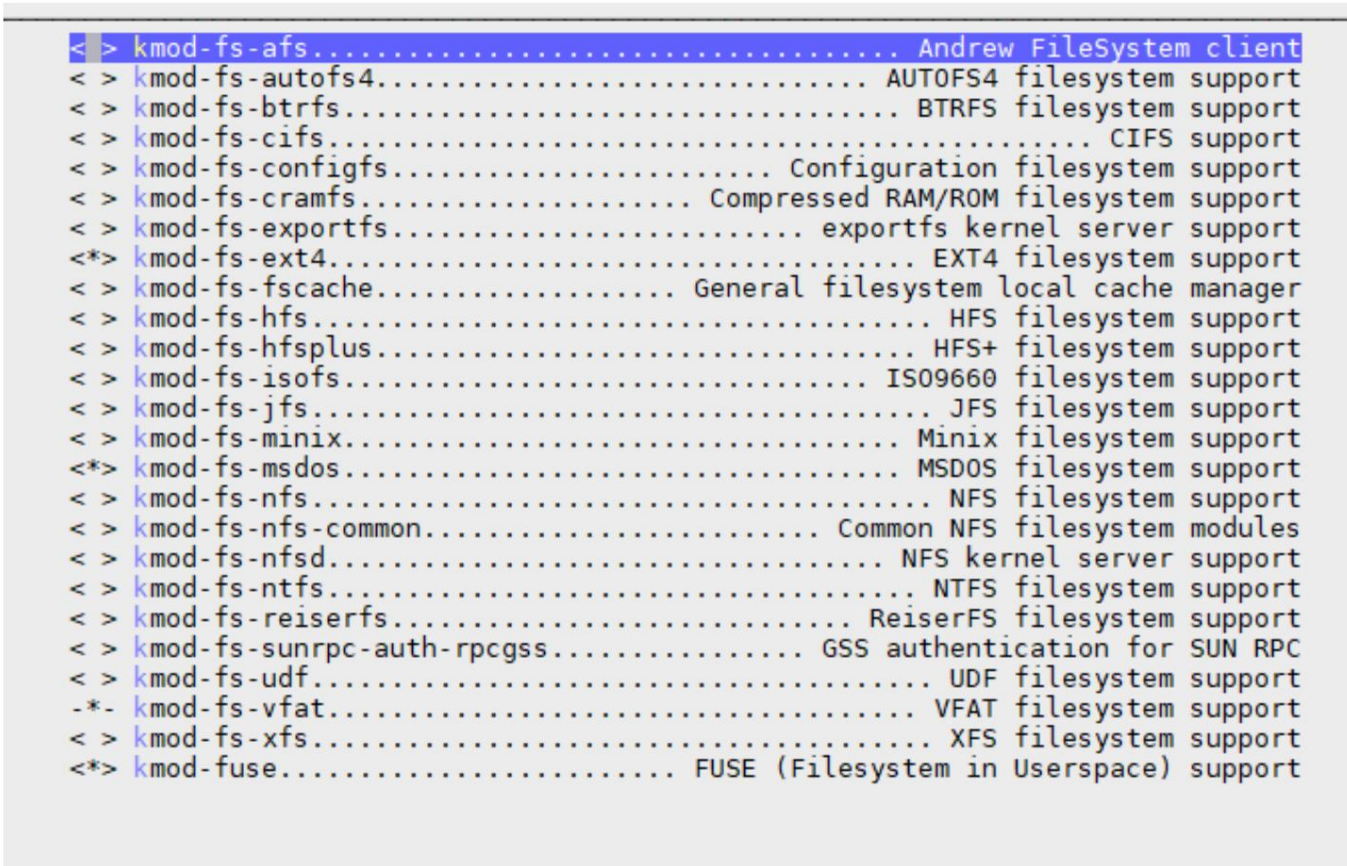
12. Supports SD cards

Requires configuring OpenWRT driver support via `make menuconfig`.

MTK Properties --->
Drivers --->
<*> kmod-mtk-mmc..... MMC/SD card support configuration
file system support:

Kernel modules --->

Filesystems --->



13. Supports Samba network sharing

Configure OpenWRT using `make menuconfig`.

Kernel modules --->

Filesystems ---> <*>

kmod-fs-cifs..... CIFS support Configuring LUCI

page:

LuCI --->

Applications --->

<*> luci-app-samba..... Network Shares - Samba SMB/CIFS module

Configure Samba

MTK Properties --->

Applications --->

<*> samba-server..... Samba Server

Modify the configuration file: package/network/services/samba36/files/samba.config

config samba

option 'name' 'OpenWrt'

option 'workgroup' 'WORKGROUP'

```
        option 'description' option                 'OpenWrt'
        'homes'                                     '1'

        option 'interface' 'wan wwan lan' #Allow access to the network interface

config sambashare
                Browseable option 'yes'
                option name 'Share'
                option path '/mnt/sda1'
                option users 'root,nobody' #Allow users
                option read_only 'no'
                option guest_ok 'yes'
                option create_mask '0700'
                option dir_mask '0700'
```

14. Supports UVC network cameras

15. Policy-based routing

16. Static Routing

17. Firewall Settings

18. Serial Port Usage

19. Add SFTP service

To add SFTP configuration, use `make menuconfig`.

```
Network --->SSH        <*> openssh-sftp-server.....
OpenSSH SFTP server
```

20. UCI Syntax

UCI configurations are divided into two types: named and anonymous.

How to add anonymous configuration:

For example, the configuration of the Wi-Fi interface is an anonymous configuration:

```
uci add wireless wifi-iface
uci set wireless.@wifi-iface[-1].device=mt7628 uci
set wireless.@wifi-iface[-1].ifname=ra1 uci set
wireless.@wifi-iface[-1].network=lan1 uci set
wireless.@wifi-iface[-1].mode=ap
uci set wireless.@wifi-iface[-1].ssid=HLK-A905
uci set wireless.@wifi-iface[-1].encryption=psk2 uci
set wireless.@wifi-iface[-1].key=12345678
```

How to add a named configuration:

For example, network interface configurations in the network section are named configurations:

```
uci set network.lan1=interface
uci set network.lan1.ifname=eth0.1
uci set network.lan1.force_link=1 uci
set network.lan1.type=bridge uci set
network.lan1.proto=static uci set
network.lan1.ipaddr=192.168.17.254
uci set network.lan1.netmask=255.255.255.0
```

21. WEB

MTK OpenWRT has a default LUCI web interface. LuCI (MTK) ---> 1. Collections --><*> Check the box for luci; otherwise, leave it unchecked.

22. Add serial port login function

23. Add password login to the debug serial port

For security reasons, it would be desirable to require a username and password when logging in via serial port, similar to SSH. You can access the console.

1. Customize Busybox:

```
make menuconfig

Base system --->
  <*> busybox .....
```

[*] Customize Busybox option

Login/Password Management Utilities ---> [*] login

(NEW)

2. Modify the startup

script: vim /etc/inittab

Change `::askconsole:/bin/ash --

login`

to `::askconsole:/bin/login`

3. The default password for root

is set in package/base-files/files/etc/shadow.

You can configure it through the web UI, view the /etc/shadow file, and then write it into the source code.

Note: During system startup, the system will execute `/etc/inittab`. The last line, `::askconsole:/bin/ash

--login /bin/ash`, with the `--login` parameter, will load `/etc/profile` before entering the command loop.

———————————————

Use the debug serial port as a regular serial port.

Debugging needs to be

disabled: Modify the mtkopenwrt kernel code:

074-noconsole.pat ch

Kernel patch:

24. USB port

The definition of pins 1-4 of USB 2.0 is the same as that of USB 3.0, the difference being the addition of two pairs of TX and RX signal lines and one Gnd line.

The 5 pins are the key pins of USB 3.0

When a USB 3.0 device is plugged into a USB 3.0 female port on the motherboard, the 4-pin (USB 2.0) signal wire at the front of the male port will connect with the 4 pins of the female port.

First, connect the 5-pin (USB 2.0) signal cable, then connect the 5-pin signal cable.

If we haven't connected the last 5-pin (USB 3.0) signal cable during the brief period the system detects the device, the device will be detected by the system.

It has been identified as a USB 2.0 device.

In other words, if the device insertion time is greater than the system's device recognition time, then a USB 3.0 device will be recognized by the system as a USB 2.0 device in advance.

Device insertion time refers to the very short period of time from the moment the device is inserted until the last 5 pins of the signal line are connected.

Question: The device was only plugged in a little slowly at first, but it was eventually fully inserted. Why is the device still recognized as a USB 2.0 device?

This is because, after the system detects the device as a USB 2.0 device,

If device detection does not restart after the male and female USB 2.0 pins are touched, we only need to restart the OS or configure the device management settings.

To disable the USB 3.0 controller, simply disable it and then enable it again.

After performing these operations, we can verify the burst transfer speed by running HD-Tach, which can reach over 100MB/s.

USB3.0 port

Time taken to write 100MB of

data using a USB 3.0 flash drive: 7.17 6.95 6.08 7.22 8.32 4.45 4.50 5.68 7.58 5.73 4.12 6.72 3.96 9.45 5.65 4.98 5.23 6.26 7.03 4.83 7.21 4.50 5.60 7.57 4.79 4.44 4.17 9.53 7.00 6.15 4.08 6.14

Time taken to read 100MB of data: 1.99 1.97 2.09 2.00 1.95 1.98 1.97 2.15 1.97 2.02 1.98 2.18 2.03 2.03 1.97 2.09 1.98 2.00 1.99 2.00 2.10 2.00 2.06 2.00 2.15 2.00 1.97 2.01 2.05 2.12 2.00 2.03

7621 Firmware Upgrade Help Guide

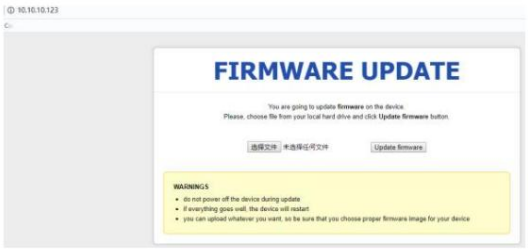
Firmware upgrade

There are several ways to upgrade firmware:

1. Press and hold the button during startup to perform the upgrade.

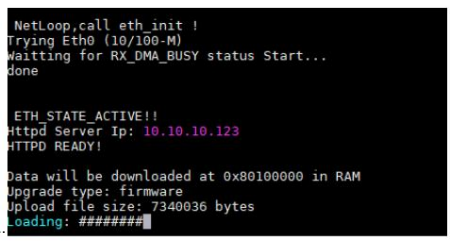
Page address: 10.10.10.123

The upgrade page is shown in the image below:

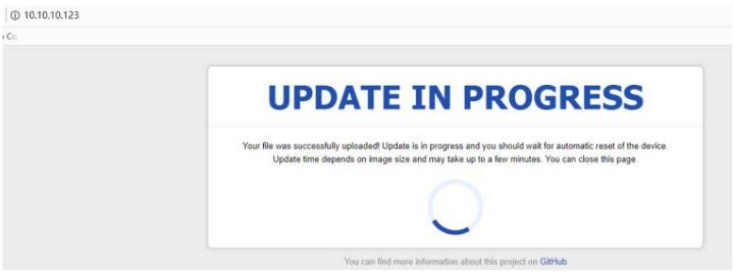


Select the file, and then click Update Firmware. Note that the page will not change while the firmware is being uploaded; you can only tell if the network port light has started by observing whether it is flashing.

If serial port 0 is connected, it will display...



During the upgrade process, a



,

"Note This" message will appear and will remain unchanged until the upgrade is complete

and the system restarts. If the debug serial port is not plugged in, you can only determine if the firmware has started by pinging 192.168.16.254.

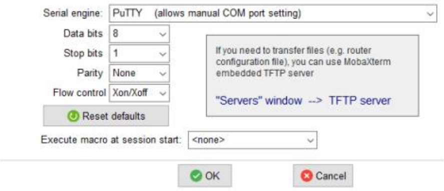
You can check the progress of the upgrade via the debug serial port.

```
Upload file size: 8126468 bytes
Loading: ####################################

HTTP upload is done! Upgrading...

****************************
*    FIRMWARE UPGRADING    *
* DO NOT POWER OFF DEVICE! *
****************************

erase 0x50000 +0x7C0004; cp.b 0x80100000 0x50000 0x7C0004
raspi_erase: offs:50000 len:7c0000
................................
```

2. Upgrade via serial port command line

Requires connection to the debug serial

port: Connect TX and RX to the USB-to-serial chip using jumper wires: chip model CP210X, connect to the module's USB port using a USB cable.



Serial port configuration

A printout appears after power-on:

```
U-Boot 1.1.3 (Jan 17 2020 - 14:34:59)

Board: Ralink APSoC DRAM:  256 MB
relocate_code Pointer at: 8ff64000

Config XHCI 40M PLL
flash manufacture id: ef, device id 40 19
info id : ef info->jedec_id :40160000 buf[1]:40 buf[2]:19
info id : ef info->jedec_id :40170000 buf[1]:40 buf[2]:19
info id : ef info->jedec_id :40180000 buf[1]:40 buf[2]:19
info id : ef info->jedec_id :40190000 buf[1]:40 buf[2]:19
find flash: W25Q256FV
*** Warning - bad CRC, using default environment

============================================
Pinney UBoot Version: 1.0
--------------------------------------------
ASIC MT7621A SingleCore (MAC to MT7530 Mode)
DRAM_CONF_FROM: Auto-Detection
DRAM_TYPE: DDR3
DRAM bus: 16 bit
Xtal Mode=3 OCP Ratio=1/3
Flash component: SPI Flash
Date:Jan 17 2020  Time:14:34:59
============================================
icache: sets:256, ways:4, linesz:32 ,total:32768
dcache: sets:256, ways:4, linesz:32 ,total:32768
set LAN/WAN WLL

 HLK7621

------------------------------------------------------------------------
             https://github.com/gnubee-git
----------------hlk7621 build:Jan 17 2020 14:34:59 --------------------
------------------------------------------------------------------------

Please choose the operation:
   1: Load system code to SDRAM via TFTP.
   2: Load system code then write to Flash via TFTP.
   3: Boot system code via Flash (default).
   4: Entr boot command line interface.
   5: Load system code then write to Flash via Httpd.
   7: Load Boot Loader code then write to Flash via Serial.
   9: Load Boot Loader code then write to Flash via TFTP.
```

It will pause for 3 seconds to allow the user to select the corresponding function.

3. After firmware boots, upgrade via the firmware's **LUCI** page.

4. After firmware boot, upgrade via command line.

OpenWRT firmware version

1. Community version **Git** address:

https://github.com/openwrt/openwrt.git

OpenWRT v18.06.6 can start normally, and the network port and debug serial port can be used normally. Other functions have not been tested.

The community version of OpenWRT does not provide any technical support.

.config

The configuration is attached.

Pre-compiled firmware: openwrt-ramips-mt 7621-mediatek_ap-mt7621a-v60-squashfs-sysupgrade.bin

Add the file to the OpenWRT SDK directory and rename the file to .config. Download

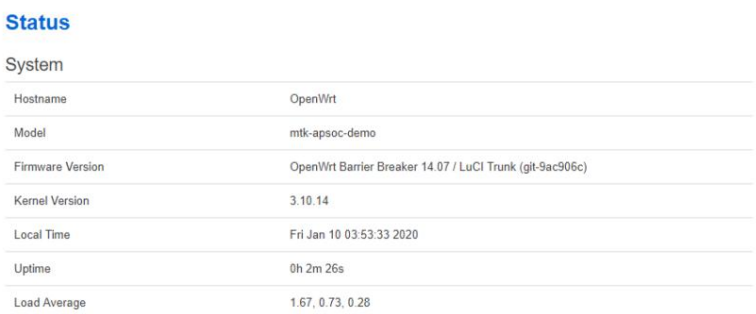
and compile instructions:

Git clone https://github.com/openwrt/openwrt.git; cd OpenWRT; git checkout v18.06.6; ./scripts/feeds update -a; ./scripts/feeds install –a

Pre-compiled firmware:

openwrt-ramips-mt 7621-mt7621-squashfs-sysupgrade.bin

2. OpenWRT version **14.07**



7621.bin

Pre-compiled firmware:

Uboot

The u-boot source code supplied by our company is an open-source version downloaded from GitHub. The address is:

https://github.com/gnubee-git/GnuBee-MT7621-uboot.git

uboot.bin

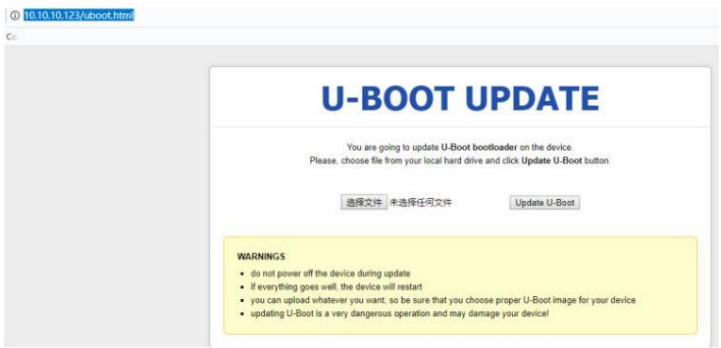
Pre-compiled uboot:

U-Boot upgrade method:

1. Press and hold the **WPS** button during startup to enter upgrade mode **via uboot :**

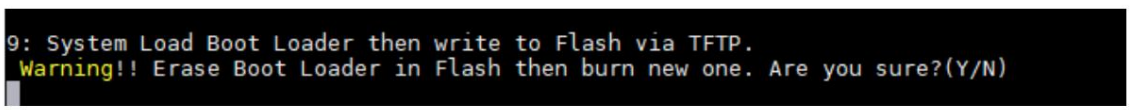
Note: The IP address of the network port connecting the PC to the module needs to be set to an IP address in the same subnet as 10.10.10.123.

Page noodle land site : http://10.10.10.123/uboot.html

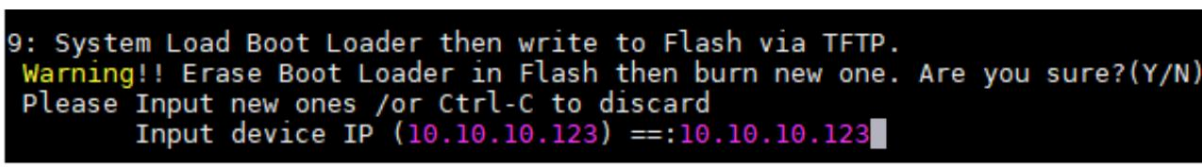


2. Upgrade via serial port command line

Type the number 9 to enter TFTP upgrade uboot mode:

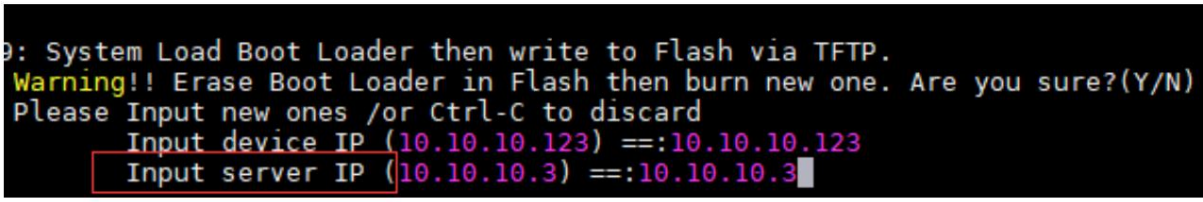


Type Y to enter Parameter configuration:



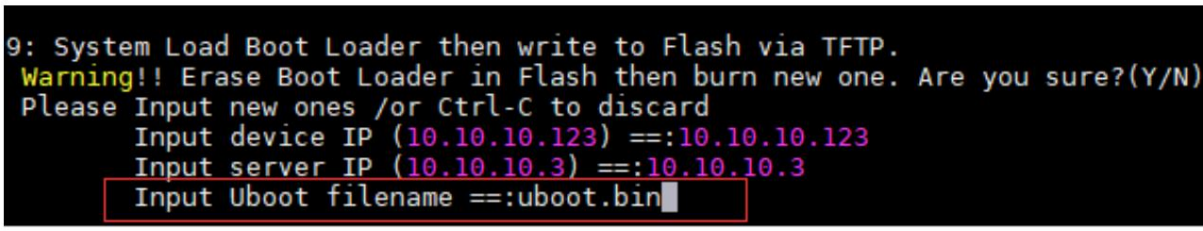
Input device IP: Configure the IP address of module

7621. car



:

Input Server IP: Configure the IP address on the PC's network port connected to the 7621.



Input Uboot filename: Enter the filename of the Uboot system to be upgraded.

3. Hardware upgrade for older **7621** models shipped from the factory.

Older versions of the 7621 come with a uboot in the flash memory, but it cannot be manually entered into firmware upgrade mode by typing numbers. It can only be entered into firmware TFTP upgrade mode by pressing and holding a button for 5 seconds during reboot. The following steps are required on the PC:

1. The PC's network port must be connected to the module's WAN port and must have the IP address 10.10.10.3.

2. Rename the firmware that needs to be upgraded to tim_uImage.

3. Open the TFTP server and change the file directory to the directory where the tim_uImage firmware is located.

7621 partition information

OpenWRT Community Edition can be viewed by examining the DTS file:

This uses AP-MT7621A-V60.dts, which shows the configuration of one of its flash partitions.

```
&spi0 {
        status = "okay";

        mx25l6405d@0 {
                #address-cells = <1>;
                #size-cells = <1>;
                compatible = "mx25l6405d","jedec,spi-nor";
                reg = <0 0>;
                spi-max-frequency = <10000000>;
                m25p,chunked-io = <32>;

                partition@0 {
                        label = "u-boot";
                        reg = <0x0 0x30000>;
                        /*read-only;*/
                };

                partition@30000 {
                        label = "u-boot-env";
                        reg = <0x30000 0x10000>;
                        read-only;
                };

                factory: partition@40000 {
                        label = "factory";
                        reg = <0x40000 0x10000>;
                        read-only;
                };

                partition@50000 {
                        label = "firmware";
                        reg = <0x50000 0x1fb0000>;
                };
        };
};
```

OpenWRT version 14.07:

This version cannot configure flash
partitioning via DTS; it can
only be done by modifying the source code. The source code is located in the SDK's build_dir/target-mipsel_24kec+dsp_uClibc-0.9.33.2/linux-ramips_mt7621/linux-3.10.14/drivers/mtd/ralink directory.

7621 flash image

7621_firmware.bin