Flashing Instructions For The Cisco Meraki MR33

Revision 0.9.1 - 2018-12-25

https://wiki.openwrt.org/toh/meraki/mr33

Contents

Disclaimer

No promises this won't brick your AP, and no promises that this will even work!

Requirements

- **Old-stock MR33 (U-Boot 2012.07!). Newer version will brick.**
- A standard PC with full network and serial access. (root/admin)
- Torx T5 screwdriver, flathead and philips screwdrivers, knife / prying tool / crowbar
- TFTP client on the PC
- A working Python 2.7 + 3.0 installation on the PC for ubootwrite
- OpenWrt compatible ssh / scp client on the PC
- Be able to define a private network in 192.168.1.0/24 (MR33 will take 192.168.1.1)
- A really good TTL-232R-3V3 Serial Adapter with configurable cable
- Three or four really small clamps or alternatively: a soldering iron, a steady hand and (de-)solderings skills.

Firstboot - Temporary Install - RAM Boot

This is the first step to get OpenWrt on the MR33. The methods described here are required for the Basic and Full installation later on in this document. Furthermore, this method can be used to recover a bricked OpenWrt image or do modifications to the UBI partitions.

Old U-Boot access method for U-Boot 2012.07

This was tested on a stock MR33 running:

```
U-Boot 2012.07-g97ab7f1 [local,local] (Oct 06 2016 - 13:07:25)
Firmware: 25-201610270754-Gd75eda32-Lccc6777d-aacharya-screen
```

Do not attempt to try this method on any later U-Boot release!

Merakis APs are known to auto update whenever possible. It's strongly advised to prevent the AP from connecting to the Internet during the procedure.

1. **Getting the files**

   Download the following files to a temporary directory on your PC:

filename	SHA256 HASH
mr33-uboot.bin	`2f3de799a93704189ca5e060e5a66672dc89b39ca757279521ab6d3d3ffecb75`
ubootwrite.py	`6aed7673cd5afbdfbe119c7532402dd754b1e90906292865461553e31c824807`
openwrt-18.06.01-ipq40xx-meraki_mr33-initramfs-fit-uImage.itb	`See sha256sums file`
openwrt-18.06.01-ipq40xx-meraki_mr33-squashfs-sysupgrade.bin	`See sha256sums file`

   Verify all downloaded files with the provided sha256 hash.

   NOTE: The provided u-boot image are NOT meant to be flashed on the "uboot" partition!

2. Disconnecting and disassembling the MR33

Disconnect all cables. To access the serial header on the PCB, remove the security screw and the four Torx T5 screws that are hiding behind each of the four plastic rubber desk mount feet on the backside.

Once all screws have been removed, carefully pry open the plastic cover. Wedge the flathead screwdriver or knife between the outer plastic housing by the security screw hole and the metal clip right next to it (mount cradle) and apply force. There are a total of four plastic retaining clips on the gray backcover grabbing outwards into the white plastic outer housing. They are organized as Groups of two. One group of clips is located located on the left hand side and the other two clips are on the righthand side.

3. Setting up serial connection

Connect the serial cable. The low profile 1x4 0.1” serial header is located near the center of the PCB, between Ethernet magnetics transformer and SoC.

Pin 1 (the little white arrow is pointing to it)	V_{cc} (3.3V - see Warning!)
Pin 2	RX
Pin 3	TX
Pin 4	Ground

Note:

Depending on the used adaptor or cable, the Pins for TX/RX can swapped. Verify the operation of the serial console with your favourite terminal app. A bad cable connection will make it impossible to continue.

Warning:

Do not connect Pin 1/V_{cc} if you are using a USB-to-TTL/CMOS cable adaptor. Usually these adaptors supply 5V and this will fry U26 on the board. Only connect it, if you are using a MAX3232-based RS232-to-TTL/CMOS adaptor, as these will need a 3.3v power supply in order to work.

Serial port parameter settings: 115200 baud rate, 8 data bits, no parity bit, 1 stop bit.

4. **Starting ubootwrite on the PC**
   (Windows users please follow User Story: Flashing with Windows)

   ```
   # chmod a+x ubootwrite.py
   # ./ubootwrite.py --write=mr33-uboot.bin
   ```

   ubootwrite default serial is ttyUSB0. Use `--serial=/dev/ttyX` parameter to change it something else. The application will wait for messages coming from the MR33.

5. **Powering up the MR33**
   Either connect the power cable or power the MR33 via Ethernet (PoE). ubootwrite will initialize almost instantly and upload an unlocked u-boot into the MR33's memory. This however takes about 5 Minutes. ubootwrite will provide progress reports during this time. The tool will exit once the custom u-boot initialized properly and started the tftpserver.

```
# ubootwrite.py --write=mr33-uboot.bin
Uploading image
Prompt is '
STINKBUG # '
Progress 0.4%, 0.6kb/s , ETA 257.0s
[...]
Progress 99.2%, 0.6kb/s, ETA 2.0s
Progress 99.6%, 0.6kb/s, ETA 1.0s
Progress 100%
COM: kernel from Legacy Image at 82000000 ...
  Image Name:   MR33-UBOOT-1
  Image Type:   ARM Linux Kernel Image (gzip compressed)
  Data Size:    168241 Bytes = 164.3 KiB
  Load Address: 86000000
  Entry Point:  86000000
  Verifying Checksum ... OK
  Uncompressing Kernel Image ...  OK
Using machid 0x8010001 from environment

Starting kernel ...

U-Boot 2012.07 [local,local] (Jan 24 2018 - 16:43:16)

DRAM:  242 MiB
machid : 0x8010001
Product: meraki_Stinkbug
[...]
ipq40xx_ess_sw_init done
eth0
Hit any key to stop autoboot:  3
CO 2
CO 1
CO 0
Hello from MR33 U-BOOT
```

Once you see the "Hello from MR33 U-Boot" message, the MR33 has successfully booted and will be listening on IPv4 *192.168.1.1* for incoming TFTP traffic. So please follow with the next step and start the tftp transfer.

6. **Uploading initramfs image**
   Upload the openwrt-ipq40xx-meraki_mr33-initramfs-fit-uImage.itb image to the MR33

```
# echo -e "binary\nput openwrt-ipq40xx-meraki_mr33-initramfs-fit-uImage.itb" | tftp 192.168.1.1
tftp> tftp> Sent 6134136 bytes in 5.2 seconds
```

   Give the MR33 some time to boot. Continue with the next step once the green status LED stops flashing and the LED stays permanently on.

7. **Connecting to the MR33 via SSH**
   use # ssh root@192.168.1.1 to connect to the DUT.

```
# ssh root@192.168.1.1


BusyBox v1.27.2 () built-in shell (ash)

 _______                     ________        __
|       |.-----.-----.-----.|  |  |  |.----.|  |_
|   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
|_______||   __|_____|__|__||________||__|  |____|
         |__| W I R E L E S S   F R E E D O M
-----------------------------------------------------
OpenWrt SNAPSHOT, r5902-9c2ac19b03
-----------------------------------------------------
=== WARNING! =====================================
There is no root password defined on this device!
Use the "passwd" command to set up a new password
in order to prevent unauthorized SSH logins.
--------------------------------------------------
root@OpenWrt:~#
```

Congratulations, you are now “root” on your own hardware. Imagine that! 😀

JTAG

Sure! Let us know. There’s a header next to the board

Secret method 1

It's a secret.

Secret method 2

This is also a secret.

Permanent installation

This section describes how to permanently install OpenWrt on the MR33. The stock firmware has left some free space (51 MiB) on the device. This is plenty for a full OpenWrt install with Luci, IPSec, VPN and more. If this is not enough; once booted into the Firstboot OpenWrt image, you can also delete the stock firmware from your device if you have no intention of reverting back to stock. The main advantage of doing this is freeing up more NAND space for packages/applications. If you would like to do this, please skip the **Fallback** and **Install** section below and go to the **Full Install** section below.

Preparation

It is assumed your device is currently booted into Firstboot “RAM Boot” for this process.
For starters: Please visit the OpenWrt Wiki and follow through the checklist:

https://wiki.openwrt.org/doc/howto/generic.flashing
https://lede-project.org/docs/guide-quick-start/factory_installation

It is strongly recommended to make a full backup of the existing firmware partitions and firmware images at this point.

https://wiki.openwrt.org/doc/howto/generic.backup

Relevant partitions to backup:

Device	Size	Name / contents
/dev/mtd0	1 MiB	sbl1
/dev/mtd1	1 MiB	mibib
/dev/mtd2	1 MiB	bootconfig
/dev/mtd3	1 MiB	qsee
/dev/mtd4	1 MiB	qsee_alt
/dev/mtd5	512 KiB	cdt
/dev/mtd6	512 KiB	cdt_alt
/dev/mtd7	512 KiB	ddrparams
/dev/mtd8	2 MiB	uboot
/dev/mtd9	2 MiB	u-boot-backup
/dev/mtd10	512 KiB	ART
/dev/mtd11	112 MiB	ubi

Fallback

OpenWrt will not replace the “part.old”. If the “part.safe” kernel becomes corrupted. The bootloader will automatically fallback to the “part.old” kernel. Please be aware that this can cause the AP to revert back to the Meraki firmware! Which might do a autoupdate that makes it impossible to install OpenWrt again.

In order to avoid the problem and have a working fallback: Install the initramfs image into “part.old”.

```
root@OpenWrt:/# file="path/to/openwrt-ipq40xx-meraki_mr33-initramfs-fit-uImage.itb"
root@OpenWrt:/# size=$(cat "$file" | wc -c)
root@OpenWrt:/# ubirename /dev/ubi0 part.old part.meraki.old
root@OpenWrt:/# ubimkvol /dev/ubi0 --size=$size --type=static --name=part.old
Volume ID 99, size 49 LEBs (6221824 bytes, 5.9 MiB), LEB size 126976 bytes (124.0 KiB), static, name
"part.old", alignment 1

Ubimkvol generates a new Volume ID (marked in red). This ID is used for the next command so please
replace the 99 with the correct value.

root@OpenWrt:/# ubiupdatevol /dev/ubi0_99 "$file"
```

This should be done before the installation, as the OpenWrt installation will request all the available space.

Basic installation

Simply download the *openwrt-ipq40xx-meraki_mr33-squashfs-sysupgrade.bin* image to /tmp of the temporary installation. Once there’s a public release, this image can be easily downloaded with wget or curl from the website. Alternatively the file can be uploaded with scp from the PC. Once the sysupgrade.bin file is in /tmp use the *sysupgrade* tool to install it. This will overwrite the “part.safe” partition! Make sure you have a backup, if you want to go back to the stock firmware.

Or as an alternative: rename the existing “part.safe” partition to get it out of harm's way.

```
root@OpenWrt:/# ubirename /dev/ubi0 part.safe part.meraki
root@OpenWrt:/# sysupgrade -v openwrt-ipq40xx-meraki_mr33-squashfs-sysupgrade.bin
```

This will terminate the ssh session and you get booted from the device. Shortly thereafter the MR33 will reboot on its own.

Full installation

If you prefer removing the Stock Firmware from your device fully, then please skip the **Fallback** and **Basic Install** section above. It is assumed your device is currently booted into Firstboot "RAM Boot" for this process. **Note that if you do this, you CAN NOT go back to stock at any time in the future.**

1. Get a list of all UBI partitions on your device as it stands as we will need to know their UBI ID's and names.

```
root@OpenWrt:~# ubinfo -a | grep -e "Volume ID" -e "Size" -e "Name"
Volume ID:   0 (on ubi0)
Size:        137 LEBs (17395712 bytes, 16.6 MiB)
Name:        diagnostic1
Volume ID:   1 (on ubi0)
Size:        133 LEBs (16887808 bytes, 16.1 MiB)
Name:        storage
Volume ID:   2 (on ubi0)
Size:        19 LEBs (2412544 bytes, 2.3 MiB)
Name:        part.safe
Volume ID:   3 (on ubi0)
Size:        19 LEBs (2412544 bytes, 2.3 MiB)
Name:        part.old
Volume ID:   4 (on ubi0)
Size:        66 LEBs (8380416 bytes, 8.0 MiB)
Name:        rootfs-25-201610270754-Gd75eda32-Lccc6777d-aacharya-screen-1
Volume ID:   5 (on ubi0)
Size:        66 LEBs (8380416 bytes, 8.0 MiB)
Name:        rootfs-25-201610270754-Gd75eda32-Lccc6777d-aacharya-screen-2
Volume ID:   6 (on ubi0)
Size:        5 LEBs (634880 bytes, 620.0 KiB)
Name:        ART
```

2. Using the information from our output, we will want to delete all partitions except for "ART" as this stores a copy of our caldata for the wireless cards. For example, to delete the "Diagnostic1" partition above, you would run:

```
root@OpenWrt:/# ubirmvol /dev/ubi0 -n 0
```

   As 0 is the "Volume ID" for the diagnostic1 partition.

3. Once all but the "ART" partitions are removed, you will then want to go through the **Fallback** section of this tutorial, **skipping** the following command:

```
root@OpenWrt:/# ubirename /dev/ubi0 part.old part.meraki.old
```

4. Now that your fallback is flashed and the stock firmware is removed, you can then go back and follow the Basic Install section of this guide to install the final OpenWrt image.

Restore stock firmware

This section only applies if OpenWrt was installed on the device. If the Firstboot method was tried in order to evaluate the image viability: Simply reboot the AP. Either through the serial console, over ssh or by disconnecting the power from the device.

If the original part.safe partition was renamed during the permanent installation procedure. Simply run the following commands on the MR33 from Firstboot:

```
root@OpenWrt:/# ubirename /dev/ubi0 part.safe part.openwrt part.old part.fallback
root@OpenWrt:/# ubirename /dev/ubi0 part.meraki part.safe part.meraki.old part.old
```

And Finally remove the OpenWrt partition:

```
root@OpenWrt:/# ubirmvol /dev/ubi0 --name=rootfs
root@OpenWrt:/# ubirmvol /dev/ubi0 --name=rootfs_data
```

Note: If the installed OpenWrt Image on the MR33 no longer boots. Use the Firstboot - Temporary Install - RAM Boot to get to the root@OpenWrt:/# shell.

Final words

Thanks to Chris for sending me one of his MR33. Thanks to Jerome C. for sending an MR33 to Chris.

Please be reasonable and do not sent any modified and or bricked MR33 back to Cisco Meraki for Servicing / Replacement.

Appendix

MR33 hardware specification overview

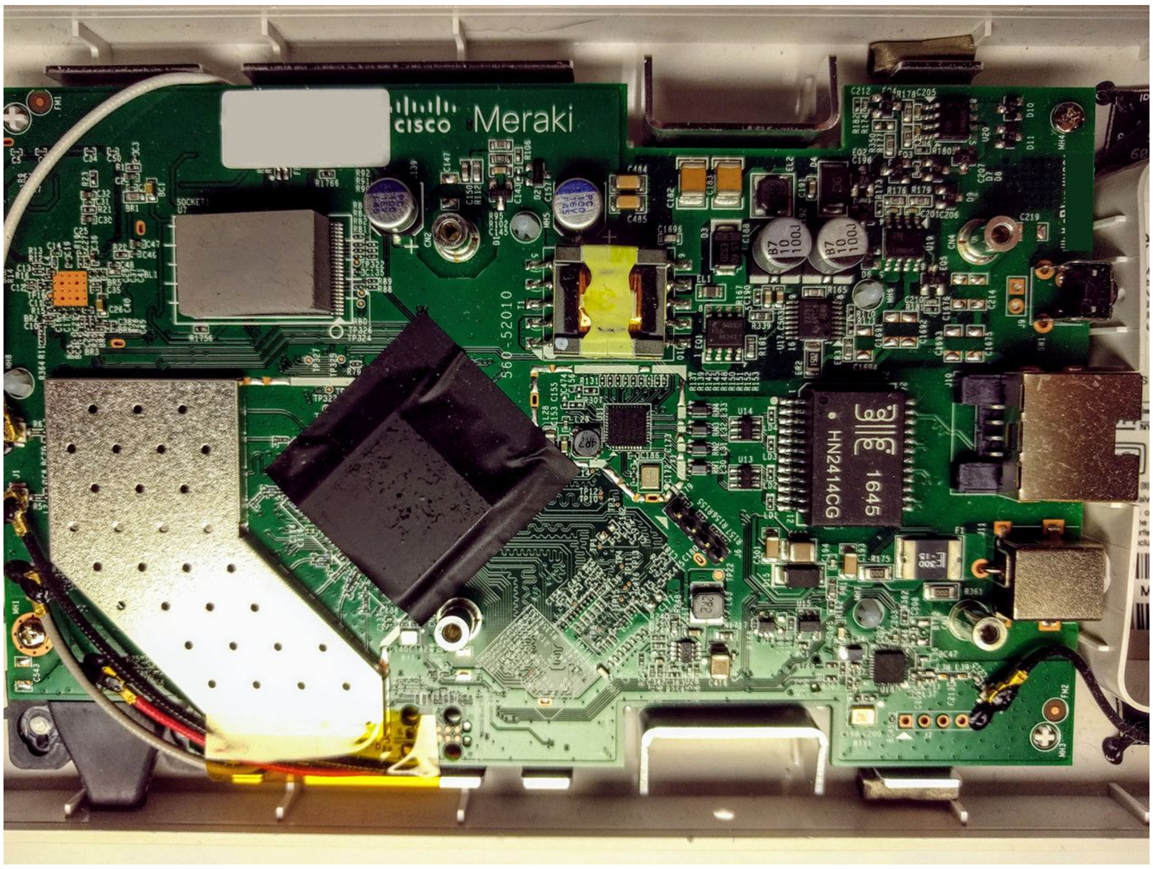
SoC	Qualcomm Atheros QCA4029 / MACHID: 8010001 / AP_DK04_1_C1
RAM	Micron MT41K128M16JT-125IT 256 MiB @ 627 MHz
FLASH	Spansion S34ML01G200TFV00 128 MiB SLC NAND (106 MiB usable)
ETH	1 x 10/100/1000 Mbps Ethernet RJ45 Port supports PoE (AR8035)
WIFI1	Integrated into SoC 2x2:2 802.11bgn 2.4 GHz
WIFI2	Integrated into SoC 2x2:2 802.11ac 5GHz up to VHT80
WIFI3	QCA9887 (168c:0050) PCIe 1x1:1 802.11abgn ac Dualband up to VHT80
LED	1 x Programmable RGB+White Status LED (driven by Ti LP5562 on i2c-1) 2 x LAN Activity / Speed LEDs (On the RJ45 Port)
BUTTON	1 x Reset button
MISC	Bluetooth LE Ti cc2650 PG2.3 4x4mm - BL_CONFIG at 0x0001FFD8 AT24C64 8KiB EEPROM Kensington Lock

Pictures

Backcover

Backcover from the inside

PCB



User Story: Meraki MR33 and 5V on the serial connector V_{cc} pin

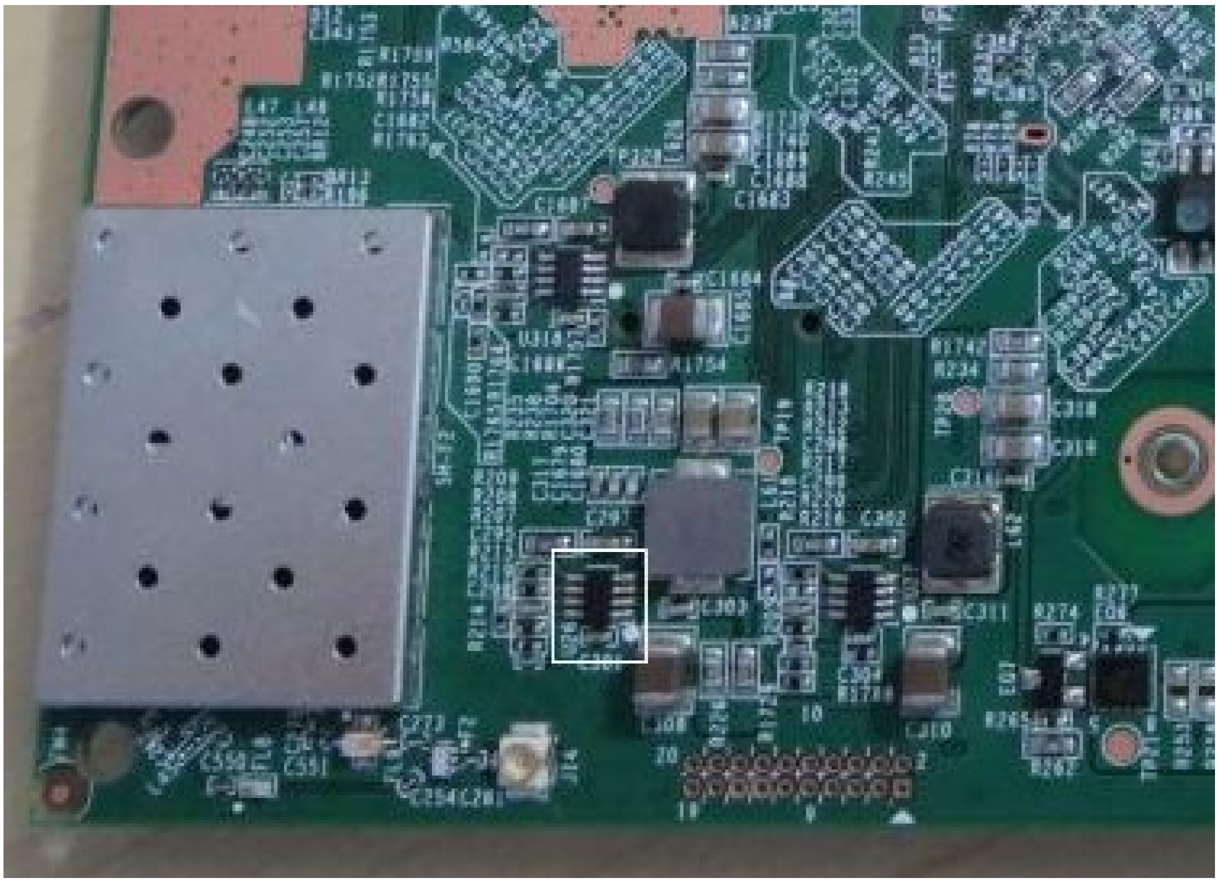
---

This is a copy from the forum post "Meraki MR33 and serial connector Vcc pin" by Fritz Kuhn (@frtz). The full thread can be found on the forum.openwrt.org site: https://forum.openwrt.org/t/meraki-mr33-and-serial-connector-vcc-pin

---

Be sure to never connect a power source to this Vcc pin of the serial connector (which is unnecessary anyway). I did by accident with the USB 5V supply. This broke my MR33. The LED stayed dark, and in fact I got a short circuit between pin 1 (Vcc, normally 3.3V) and pin 4 (ground) of the serial connector on the board.

I investigated the issue and found that the short circuit was made by U26. U26 is tagged AEUH which means that this is a MP2233DJ DC-DC converter made by Monolithic Power Systems. With some SMD reworking skills and some luck to find a supplier for a single piece, you can replace the circuit. I got the MR33 back up and running this way.



User Story: Flashing with Windows

This is a copy from a post on the LEDE-MR33 github project page:
https://github.com/riptidewave93/LEDE-MR33/issues/1#issuecomment-420016794
By the user @ghchesser.

Hi. I'm flashing MR33 using a laptop with Windows 7. The command in CMD supposed to look like:

```
python ubootwrite.py --serial=COM[#] --write=[path to the file]mr33-uboot.bin
```

Where [#] is the numeric value for the COM-port where MR33 is connected to. [path to the file] is the path to the folder with the .bin file.

Python gives messages about inconsistent indentation in the ubootwrite.py file. The easiest way to fix it is to replace all tab-symbols to 8 space-symbols.

I execute the command and get the message "Uploading image". After that I connect the power cable to MR33 and see the process going.

```
...
CO 1
CO 0
Hello from MR33 U-BOOT
Creating 1 MTD partitions on "nand0":
0x000000c00000-0x000007c00000 : "mtd=0"
```

User Story: Bricked MR33 due to new U-Boot 2017.07

This is a copy from a post on the LEDE-MR33 github project page:
https://github.com/riptidewave93/LEDE-MR33/issues/13
By the user @0xFelix

Issue #13: Not working on U-Boot 2017.07-RELEASE-g78ed34f31579 (Sep 29 2017 - 07:43:44 -0700)

Seems like Cisco blocks the old method of getting a serial prompt in this uboot version?
The ubootwrite.py script is sending xyzzy, but no prompt appears.
Maybe CONFIG_AUTOBOOT_STOP_STR was changed?
Where can one obtain the latest GPL sources of Meraki's uboot?

Update:
I tried pressing space instead on boot...
Resulted in the device saying:
Secure boot NOT enabled! Blowing fuses... Resetting now.
It is dead now, won't boot and the orange LED stays lit.
Maybe Cisco is not so keen on replacing this device's firmware afterall.

Miscellaneous hardware information

/proc/cpuinfo

```
processor       : 0
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 6.91
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

processor       : 1
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 6.91
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

processor       : 2
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 6.91
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

processor       : 3
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 6.91
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5

Hardware        : Generic DT based system
Revision        : 0000
Serial          : 0000000000000000
```

/proc/interrupts

```
           CPU0       CPU1       CPU2       CPU3
18:      250095     250101     250101     250101     GIC-0  20 Level     arch_timer
40:           1          0          0          0   msmgpio  18 Edge      gpio-keys
122:          7          0          0          0     GIC-0 270 Level     bam_dma
123:        111          0          0          0     GIC-0 129 Level     i2c_qup
124:          0          0          0          0     GIC-0 239 Level     bam_dma
125:          7          0          0          0     GIC-0 139 Level     msm_serial0
127:        132          0          0          0     GIC-0  97 Edge      edma_eth_tx0
128:       1128          0          0          0     GIC-0  98 Edge      edma_eth_tx1
129:          0          0          0          0     GIC-0  99 Edge      edma_eth_tx2
130:          0          0          0          0     GIC-0 100 Edge      edma_eth_tx3
131:        169          0          0          0     GIC-0 101 Edge      edma_eth_tx4
132:         28          0          0          0     GIC-0 102 Edge      edma_eth_tx5
133:          0          0          0          0     GIC-0 103 Edge      edma_eth_tx6
134:          0          0          0          0     GIC-0 104 Edge      edma_eth_tx7
135:         76          0          0          0     GIC-0 105 Edge      edma_eth_tx8
136:         33          0          0          0     GIC-0 106 Edge      edma_eth_tx9
137:          0          0          0          0     GIC-0 107 Edge      edma_eth_tx10
138:          0          0          0          0     GIC-0 108 Edge      edma_eth_tx11
139:         32          0          0          0     GIC-0 109 Edge      edma_eth_tx12
140:         27          0          0          0     GIC-0 110 Edge      edma_eth_tx13
141:          0          0          0          0     GIC-0 111 Edge      edma_eth_tx14
142:          0          0          0          0     GIC-0 112 Edge      edma_eth_tx15
143:       2002          0          0          0     GIC-0 272 Edge      edma_eth_rx0
145:       2249          0          0          0     GIC-0 274 Edge      edma_eth_rx2
147:       1640          0          0          0     GIC-0 276 Edge      edma_eth_rx4
149:       1160          0          0          0     GIC-0 278 Edge      edma_eth_rx6
175:         18          0          0          0     GIC-0 200 Edge      ath10k_ahb
192:         23          0          0          0     GIC-0 201 Edge      ath10k_ahb
193:      20871          0          0          0     GIC-0 133 Level     bam_dma
194:         23          0          0          0     GIC-0 173 Edge      qcom-pcie-msi
195:        133          0          0          0     GIC-0 130 Level     i2c_qup
196:          0          0          0          0   PCI-MSI   0 Edge      aerdrv
197:         23          0          0          0   PCI-MSI   1 Edge      ath10k_pci
IPI0:          0          0          0          0  CPU wakeup interrupts
IPI1:          0          0          0          0  Timer broadcast interrupts
IPI2:       2025       2790       2712       3847  Rescheduling interrupts
IPI3:         19        298       1587        360  Function call interrupts
IPI4:          0          0          0          0  CPU stop interrupts
IPI5:         27       2625       3460       3499  IRQ work interrupts
IPI6:          0          0          0          0  completion interrupts
```

/proc/iomem

```
00022000-0002213f : /soc/rng@22000
00080000-00081fff : parf
00090000-00090063 : /soc/mdio@90000
004ab000-004ab003 : /soc/restart@4ab000
01000000-012fffff : /soc/pinctrl@1000000
01800000-0185ffff : /soc/clock-controller@1800000
01949000-019490ff : /soc/tcsr@1949000
01953000-01953fff : /soc/ess_tcsr@1953000
01957000-019570ff : /soc/tcsr@1957000
07884000-078a6fff : /soc/dma@7884000
078af000-078af1ff : msm_serial
078b0000-078b01ff : msm_serial
078b7000-078b75ff : /soc/i2c@78b7000
078b8000-078b85ff : /soc/i2c@78b8000
07984000-0799dfff : /soc/dma@7984000
079b0000-079b0fff : /soc/qpic-nand@79b0000
08e04000-08e23fff : /soc/dma@8e04000
08e3a000-08e3ffff : /soc/crypto@8e3a000
0a000000-0a1fffff : /soc/wifi@a000000
0a800000-0a9fffff : /soc/wifi@a800000
0b017000-0b01703f : /soc/watchdog@b017000
0c000000-0c07ffff : essedma
0c080000-0c087fff : /soc/edma@c080000
40000000-40000f1c : dbi
40000f20-40000fc7 : elbi
40300000-405fffff : MEM
 40300000-405fffff : PCI Bus 0000:01
   40300000-4030ffff : 0000:01:00.0
   40400000-405fffff : 0000:01:00.0
     40400000-405fffff : ath
80000000-87dfffff : System RAM
 80208000-80722fff : Kernel code
 8075e000-807d9867 : Kernel data
```

/proc/mtd

```
dev:    size   erasesize  name
mtd0: 00100000 00020000 "sbl1"
mtd1: 00100000 00020000 "mibib"
mtd2: 00100000 00020000 "bootconfig"
mtd3: 00100000 00020000 "qsee"
mtd4: 00100000 00020000 "qsee_alt"
mtd5: 00080000 00020000 "cdt"
mtd6: 00080000 00020000 "cdt_alt"
mtd7: 00080000 00020000 "ddrparams"
mtd8: 00200000 00020000 "u-boot"
mtd9: 00200000 00020000 "u-boot-backup"
mtd10: 00080000 00020000 "ART"
mtd11: 07000000 00020000 "ubi"
```

/proc/crypto

```
name         : hmac(sha256)
driver       : hmac-sha256-qce
type         : ahash
async        : yes
blocksize    : 64
digestsize   : 32

name         : hmac(sha1)
driver       : hmac-sha1-qce
type         : ahash
async        : yes
blocksize    : 64
digestsize   : 20

name         : sha256
driver       : sha256-qce
type         : ahash
async        : yes
blocksize    : 64
digestsize   : 32

name         : sha1
driver       : sha1-qce
type         : ahash
async        : yes
blocksize    : 64
digestsize   : 20

name         : cbc(des3_ede)
driver       : cbc-3des-qce
type         : ablkcipher
async        : yes
blocksize    : 8
min keysize  : 24
max keysize  : 24
ivsize       : 8
geniv        : <default>

name         : ecb(des3_ede)
driver       : ecb-3des-qce
type         : ablkcipher
async        : yes
blocksize    : 8
min keysize  : 24
max keysize  : 24
ivsize       : 0
geniv        : <default>

name         : cbc(des)
driver       : cbc-des-qce
type         : ablkcipher
async        : yes
blocksize    : 8
min keysize  : 8
max keysize  : 8
ivsize       : 8
geniv        : <default>
```

```
name         : ecb(des)
driver       : ecb-des-qce
type         : ablkcipher
async        : yes
blocksize    : 8
min keysize  : 8
max keysize  : 8
ivsize       : 0
geniv        : <default>

name         : xts(aes)
driver       : xts-aes-qce
type         : ablkcipher
async        : yes
blocksize    : 16
min keysize  : 16
max keysize  : 32
ivsize       : 16
geniv        : <default>

name         : ctr(aes)
driver       : ctr-aes-qce
type         : ablkcipher
async        : yes
blocksize    : 16
min keysize  : 16
max keysize  : 32
ivsize       : 16
geniv        : <default>

name         : cbc(aes)
driver       : cbc-aes-qce
type         : ablkcipher
async        : yes
blocksize    : 16
min keysize  : 16
max keysize  : 32
ivsize       : 16
geniv        : <default>

name         : ecb(aes)
driver       : ecb-aes-qce
type         : ablkcipher
async        : yes
blocksize    : 16
min keysize  : 16
max keysize  : 32
ivsize       : 16
geniv        : <default>
```

OpenSSL speed test

Testsetup is defined at https://wiki.openwrt.org/doc/howto/benchmark.openssl

```
OpenSSL 1.0.2n  7 Dec 2017
built on: reproducible build, date unspecified
options:bn(64,32) rc4(ptr,char) des(idx,cisc,2,long) aes(partial) blowfish(ptr)
compiler: arm-openwrt-linux-muslgnueabi-gcc -I. -I.. -I../include  -fPIC -DOPENSSL_PIC
-DOPENSSL_THREADS -D_REENTRANT -DDSO_DLFCN -DHAVE_DLFCN_H -I[...] -znow -zrelro
-DOPENSSL_SMALL_FOOTPRINT -DOPENSSL_NO_ERR -DTERMIOS -Os -pipe -mcpu=cortex-a15 -mfpu=neon-vfpv4
-fno-caller-saves -fno-plt -fhonour-copts -Wno-error=unused-but-set-variable -Wno-error=unused-result
-mfloat-abi=hard -i[...] -Wformat -Werror=format-security -fstack-protector -D_FORTIFY_SOURCE=1
-Wl,-z,now -Wl,-z,relro -fpic -ffunction-sections -fdata-sections -fomit-frame-pointer -Wall
-DOPENSSL_BN_ASM_MONT -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -DAES_ASM -DBSAES_ASM
-DGHASH_ASM
The 'numbers' are in 1000s of bytes per second processed.
type             16 bytes     64 bytes    256 bytes   1024 bytes   8192 bytes
md5               3616.85k    12728.24k    33591.81k    58226.82k    74100.86k
sha1              3561.18k    11070.60k    25805.91k    38857.55k    45276.45k
des cbc           6187.52k     6361.00k     6422.95k     6413.44k     6417.52k
des ede3          2190.62k     2213.76k     2219.12k     2230.27k     2226.26k
aes-128 cbc      12486.43k    14036.48k    14445.46k    14564.61k    14650.03k
aes-192 cbc      11110.66k    12314.90k    12608.55k    12714.27k    12787.71k
aes-256 cbc      10147.67k    11105.34k    11431.00k    11468.12k    11490.57k
sha256            4425.98k    10450.56k    18634.67k    23256.03k    25152.17k
sha512            2353.08k     9393.84k    14187.26k    19659.44k    22306.82k
                 sign    verify    sign/s verify/s
rsa 2048 bits 0.050969s 0.001216s     19.6    822.4
                 sign    verify    sign/s verify/s
dsa 2048 bits 0.013550s 0.014327s     73.8     69.8

| r5944 | ARMv7 Processor rev 5 (v7l) | 103.32 | ARMv7 Processor rev 5 (v7l) | 103.32 | ARMv7 Processor
rev 5 (v7l) | 103.32 | ARMv7 Processor rev 5 (v7l) | 103.32 | Generic DT based system | 1.0.2n |
58226820 | 38857550 | 23256030 | 19659440 | 6413440 | 2230270 | 14564610 | 12714270 | 11468120 | 19.6 |
822.4 73.8 | 69.8 |
```

dmesg extracts

```
[    0.000000] arm_arch_timer: Architected cp15 timer(s) running at 48.00MHz (virt).
[    0.000000] clocksource: arch_sys_counter: mask: 0xffffffffffffff max_cycles: 0xb11fd3bfb,
max_idle_ns: 440795203732 ns
[    0.000008] sched_clock: 56 bits at 48MHz, resolution 20ns, wraps every 4398046511096ns
[    0.000022] Switching to timer-based delay loop, resolution 20ns
[    0.000420] Calibrating delay loop (skipped), value calculated using timer frequency.. 96.00
BogoMIPS (lpj=480000)

[    0.005319] Brought up 4 CPUs
[    0.005338] SMP: Total of 4 processors activated (384.00 BogoMIPS).
[    0.005346] CPU: All CPU(s) started in SVC mode.
[    0.017754] VFP support v0.3: implementor 41 architecture 2 part 30 variant 7 rev 5
[    0.017953] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns:
19112604462750000 ns

[    1.115753] nand: device found, Manufacturer ID: 0x01, Chip ID: 0xf1
[    1.116099] nand: AMD/Spansion S34ML01G2
[    1.122692] nand: 128 MiB, SLC, erase size: 128 KiB, page size: 2048, OOB size: 64

[    1.544080] at24 0-0050: 8192 byte 24c64 EEPROM, read-only, 0 bytes/write

[    1.754516] UBI: auto-attach mtd11
[    1.754808] ubi0: attaching mtd11
[    2.332143] ubi0: scanning is finished
[    2.340788] ubi0: attached mtd11 (name "ubi", size 112 MiB)
[    2.340818] ubi0: PEB size: 131072 bytes (128 KiB), LEB size: 126976 bytes
[    2.345203] ubi0: min./max. I/O unit sizes: 2048/2048, sub-page size 2048
[    2.352112] ubi0: VID header offset: 2048 (aligned 2048), data offset: 4096
[    2.358986] ubi0: good PEBs: 896, bad PEBs: 0, corrupted PEBs: 0
[    2.365754] ubi0: user volume: 10, internal volumes: 1, max. volumes count: 128
[    2.371992] ubi0: max/mean erase counter: 5/2, WL threshold: 4096, image sequence number: 2086049366
[    2.379041] ubi0: available PEBs: 0, total reserved PEBs: 896, PEBs reserved for bad PEB handling:
20
[    2.388435] ubi0: background thread "ubi_bgt0d" started, PID 102

[    2.425767] This architecture does not have kernel memory protection.
```