ChatGPT

Linux Kernel Support for Kobo Touch N905C (Freescale i.MX507)

Current Kernel Support Status

The Kobo Touch N905C currently relies on a very old Linux 2.6.35.3 kernel (as used in Kobo's official firmware and projects like Quill OS) [1]. There is **no out-of-the-box mainline (5.x/6.x) kernel support for the N905C** at this time. In other words, you cannot yet download a vanilla upstream Linux and find a device tree for the N905C. The device's original kernel is heavily customized for the i.MX507 SoC and lacks Device Tree support (using board files instead). Upstream developers have **started adding support for related Kobo devices and the i.MX50 family**, which is encouraging for a future N905C port:

- **Kobo Aura (Model N514)** – a 2013 Kobo e-reader with the same i.MX50x SoC – now has a Device Tree in mainline Linux [2]. This means core support for the i.MX50 SoC (including its E-Ink display controller, SD controllers, etc.) is present in modern kernels. The Aura's DTS (arch/arm/boot/dts/**imx50-kobo-aura.dts**) can serve as a **reference for N905C**, as the devices share many similarities. For example, the Aura DTS defines the SoC, memory, SDIO Wi-Fi, and other peripherals for an i.MX50-based e-reader [3] [4].

- **Freescale/NXP i.MX50 EVK** – mainline Linux also has a device tree for the i.MX50 evaluation kit (imx50-evk.dts) [5]. This can provide additional reference for base SoC features (clocks, I/O, etc.) when porting N905C.

Apart from these, **no official Device Tree for the N905C or its direct variants (Kobo Touch N905/N905B)** exists yet. Community efforts (e.g. PostmarketOS) to run mainline on N905C are in early stages – the device is marked "not booting" on their list [6]. In summary, **a modern kernel will require creating a new Device Tree for the N905C**, merging knowledge from the downstream 2.6.35 kernel and the upstream Kobo Aura i.MX50 support.

Sources: Quill/InkBox OS developer notes [1]; mainline Linux device tree sources for Kobo Aura [2].

Internal Hardware of the Kobo Touch N905C

To port the N905C to a modern kernel (with Device Tree), we must enumerate its hardware components, along with their addresses and connections. The following table summarizes the **internal devices** in the Kobo Touch N905C, including SoC features and peripheral chips:

Component	Description	Connection / Addresses
SoC (CPU + MCUs)	*Freescale/NXP i.MX507* – 800 MHz ARM Cortex-A8 processor, designed for e-readers. It is essentially an i.MX508 **without the OpenVG GPU** acceleration [7]. The SoC integrates many peripherals, notably a hardware EPD (E-ink) controller.	On-chip memory-mapped registers for peripherals (e.g. the EPD controller) are within the SoC's address map. **EPDC (E-Paper Display Controller)** registers start at **0x020F4000** (size 0x4000) [8], with an interrupt line (IRQ 97 on the i.MX50 interrupt controller) [8]. The i.MX507 also provides 3× SDIO (eSDHC) interfaces for storage/Wi-Fi, I^2C and SPI buses, UART, USB OTG, etc. (The N905C uses these buses as detailed below.)
Display (E-Ink Panel)	*6-inch E Ink "Pearl" display*, 600 × 800 resolution, 16-level grayscale [9] [10]. This is the reflective electronic paper screen used for reading. Likely model ED060SC7 or similar from E Ink (PVI) – a classic Pearl EPD panel.	**Connected to SoC EPDC** interface. The i.MX507's integrated EPDC drives the panel timing and waveform. The EPDC bus is wired directly to the E-Ink panel's connector (timing controller). No separate frame buffer RAM on panel – the SoC drives pixels via EPDC registers. (VCOM and EPD voltages are managed by the PMIC – see power section.)
Touchscreen	*Neonode zForce IR Touch* – Infrared touch sensor array around the screen bezel [11]. This system of IR emitters and detectors forms a touch interface (it detects touch by interruption of IR beams, allowing use with gloves, etc.). The N905C uses zForce for touch input instead of a capacitive overlay.	**I^2C-connected controller:** The Neonode sensor board includes a controller that communicates via I^2C. In similar devices it uses I^2C address **0x50** [12]. An **interrupt GPIO** is used to signal touch events to the SoC (the controller pulls an IRQ line low when touch data is ready). For example, on later Kobo devices the zForce IRQ and reset were on GPIO5_6 and GPIO5_9 [12] – N905C likely uses equivalent lines. The Device Tree would include something like `compatible = "neonode,zforce"; reg = <0x50>; interrupts = <…>; reset-gpios = <…>; x-size = <800>; y-size = <600>;` [13].

Component	Description	Connection / Addresses
Wi-Fi Module	*802.11b/g/n Wi-Fi (Wireless LAN).* The N905C has a built-in Wi-Fi radio for connectivity. Kobo devices of this generation use an SDIO Wi-Fi module. Notably, the Kobo Aura (which is similar hardware) uses a **CyberTan WC121** module containing a **Broadcom BCM43362** Wi-Fi chip [14] [15]. The N905C likely uses the same (or a very similar) module supporting 2.4 GHz 802.11n.	**Connected via SDIO (SD bus):** The Wi-Fi module is wired to one of the i.MX507's eSDHC interfaces (4-bit SDIO mode). In the Kobo Aura's DTS (and likely N905C), the Wi-Fi is on `esdhc2` (SD2) with **bus width 4** and a max clock ~50 MHz [14]. A **GPIO-controlled power rail** is used to power-cycle the Wi-Fi: e.g. the Aura uses a GPIO regulator on **GPIO4_12** to enable 3.3 V to the Wi-Fi card [16], and a separate **reset GPIO (GPIO4_17)** via a "mmc-pwrseq" node for proper reset sequencing [17]. These lines would be reflected in the Device Tree for N905C. (The Wi-Fi driver in mainline for BCM43362 is `brcmfmac`, which would require the correct **firmware blob** in `/lib/firmware/brcm`.)
Internal Storage	*Main Flash Storage:* **2 GB** internal memory for the OS and user data. The Kobo Touch models have all storage on a removable microSD in some variants. In fact, the **original N905 used a soldered 2 GB eMMC (Samsung moviNAND)** [18], whereas the **N905C (cost-reduced model)** uses a 2 GB microSD card internally (located in a slot on the PCB) as its internal drive (this has been confirmed by users who found a removable microSD inside the N905C). This contains the Kobo Linux system and books.	**Connected via SDIO:** The internal storage is connected to another eSDHC interface on the i.MX507. Typically the internal eMMC/microSD is on the 8-bit SDIO interface (`esdhc3`). For example, in the Kobo Aura DTS, `esdhc3` is configured as **8-bit, non-removable** for the on-board eMMC [19]. The N905C's microSD would be the same interface. Additionally, the N905C has an **external microSD slot** for user expansion (up to 32 GB cards supported) [20], which is likely wired to `esdhc1` (with a card-detect GPIO) [21]. This allows the user to side-load content or even boot alternative systems from an external card in some hack scenarios.

Component	Description	Connection / Addresses
Power Management (PMIC)	*Power Management IC:* The Kobo Touch uses a dedicated PMIC to generate the multiple voltages needed (core, I/O, memory, E-Ink panel supplies, etc.). The **original Touch (N905)** was reported to use the **Freescale MC13892 PMIC** [18] – a power IC originally for i.MX51 – to manage power rails (likely reused from earlier designs). The N905C being later might use a different PMIC (Freescale's recommended PMICs for i.MX50 were the **MC34708** or *MMPF0100* [22], so it's plausible one of these is present). The PMIC provides regulators for VDDCORE, VDDIO, memory, and specialized supplies for the E-Ink panel (e.g. ±VEInk, Vcom).	**I²C interface (for control):** The PMIC is connected via an I²C bus to the SoC for control of regulators. (For instance, MC13892 uses I²C address 0x08 on i.MX51 kits.) In the N905C's downstream kernel, the PMIC would be configured in the board file; in a Device Tree, it would appear as an I²C device with regulator subnodes. If using MC34708 or similar, NXP provides Linux drivers for these. **Battery:** The Kobo Touch has a single-cell Li-ion battery (around *1300 mAh* capacity, though exact varies) providing weeks of runtime. It does **not** use a smart fuel-gauge; battery level is inferred from voltage. **LED:** Next to the power slider is a tri-color LED used for charge/status indication [23]. On N905C, this LED can glow blue or green (and combinations for amber) to show charging states [24]. The LED is controlled by GPIOs from the PMIC or SoC (e.g. the Aura uses a GPIO (GPIO6_24) for its orange LED [25]). The Device Tree can use a `gpio-leds` node to define this indicator.
Buttons & Sensors	*Buttons:* The N905C has **one front button** (Home button) centered below the screen, and a **sliding power switch** on the top edge [23]. The Home button returns to the menu, and the slider handles power and wake/sleep. There is **no frontlight** on this model (that appeared in later Kobos), and **no accelerometer** (orientation is fixed). *Sensors:* Unlike newer models, the N905C does not have a magnetometer Hall sensor (for cover detection) – that feature was introduced in later devices with sleep covers.	**GPIO connections:** The Home button and Power slider are wired to SoC GPIO inputs, configured as keys. For example, the power slider might be on a GPIO that generates KEY_POWER (in the Aura DTS, the power key is on GPIO4_10) [26]. The Home button likely generates a Linux key code (e.g. KEY_HOME) – it would also be listed in a `gpio-keys` DT node. (In the downstream kernel, these were handled in the board file key matrix.) **Reset pin:** There is a pinhole reset (hard reset) switch on the back of the device [27], which physically resets the device – not software-controlled.

Datasheets & References: For more information on these components, you can refer to the *NXP i.MX507* official datasheet/reference manual (which details the memory map and controller registers) [28], the *E Ink Pearl display (ED060SC7)* datasheet for panel specs [10], Neonode's documentation on the zForce IR touchscreen, and NXP's documentation for the PMIC (e.g. MC34708 or MC13892). These will be useful when writing device tree entries (to get correct register addresses, voltages, GPIO active levels, etc.). For example,

knowing the EPDC base address (0x020F4000) [8] and the interrupt number will be needed in the SoC .dtsi file, and knowing the Wi-Fi module's power GPIO helps set up a regulator in the DTS.

Existing Device Tree Files and Porting Resources

Although the N905C lacks a dedicated Device Tree in mainline Linux, **closely related device trees already exist** that can jump-start the porting process:

- **Mainline i.MX50 DTSI:** The file `arch/arm/boot/dts/imx50.dtsi` in the Linux source defines the core i.MX50/507 SoC nodes (clocks, UART, I2C, SPI, EPDC, SDHC, USB, etc.). It will include the addresses and interrupts for all on-chip peripherals. For instance, it defines the EPDC node (`epdc@20f4000`) with reg `<0x020f4000 0x4000>` and IRQ 97 [8], and entries for SDHC controllers, I2C buses, etc. This is the foundation – your N905C DTS will include `imx50.dtsi` and then add board-specific customizations.

- **Kobo Aura (N514) DTS:** As mentioned, `imx50-kobo-aura.dts` is in mainline (added in Linux 5.4+ by Jonathan Neuschäfer) [2]. The Aura is a 6″ Kobo with i.MX508 (actually i.MX507 at 1 GHz) and many similar components (CPU, PMIC, Wi-Fi, SD). Studying its DTS gives insight into how to describe the N905C. Notable differences: the Aura has a capacitive touchscreen (Atmel controller) instead of IR, and it has frontlight LEDs and a Hall sensor – those entries would not apply to N905C. However, features like the **Wi-Fi SDIO node** (with Broadcom BCM43362) [4], the **SD card slot (esdhc1 with cd gpio)】,** GPIO keys for power [26]**, and** regulator for Wi-Fi power** [16] are very relevant. You can largely reuse and adjust those for N905C (changing GPIO numbers if needed, based on board schematics). The Aura DTS also shows how the memory is declared (256 MB at 0x70000000) [3], which should be the same for N905C if it also has 256 MB RAM.

- **Kobo Glo (N613) / Kobo Mini (N705):** These are other i.MX507-based Kobo models (Glo is basically a Touch with a frontlight and higher-res screen; Mini is a smaller 5″ version). They don't have mainline DTS yet, but community projects (like *postmarketOS*) have interest in them. In fact, all these devices share a lot with N905C. If any unofficial device trees or patches exist for Glo or Touch, they could be found in forums or Git repositories. As of now, the **device tree work for Kobo i.MX50 devices is in early stages**, aside from the Aura which is upstream. It may be necessary to write the N905C device tree from scratch, guided by the older kernel source and Aura DTS.

- **Downstream Kobo Kernel Source:** For detailed hardware info, the *Kobo (Netronix) vendor kernel 2.6.35.3 source* is invaluable. Nickel (the Kobo firmware) had a kernel with board files under "hw/imx507". In fact, the kernel used by Quill/InkBox was based on Kobo's GPL release. As noted by one maintainer, the "original Kobo kernel was copied from kobolabs' repo" [29]. That source (available via Marek's GitHub [29] or the Quill-OS repo) contains board initialization code for the Touch. Look for files like `board-ntx508.c` or similar (NTX = Netronix). Those will list the devices, GPIOs, and I2C addresses in C code. While dated, you can translate that information into a modern DTS. For example, it likely defines the I^2C device for the Neonode touch and sets up the GPIO IRQ, etc., which you can port to a DT node.

In summary, **to get N905C on a modern kernel**, you will create a new DTS file (perhaps `imx50-kobo-touch-n905c.dts`). Start by including `imx50.dtsi`. Then define the **memory node** (likely 128 MB or

256 MB – Kobo Touch is believed to have 256 MB) at the correct address [3]. Add the **GPIO key** for the power slider and home button (similar to Aura's gpio-keys but only one button) [26]. Add a **gpio-leds** node for the tricolor LED (if you plan to drive it; or at least one LED). Next, define the **Wi-Fi SDIO** (you can copy the Aura's `esdhc2` node and adjust if needed) [15]. Then, crucially, add the **Neonode touchscreen** on the I2C bus: e.g. if the IR touch is on I2C2 with interrupt on a certain GPIO, use `compatible="neonode,zforce"` and the properties from the Neonode DT binding [13]. Finally, disable any Aura-specific bits not present (frontlight, etc.). This will result in a DTS that describes the N905C's hardware.

Booting and Testing Custom Kernels on i.MX507 (Kobo N905C)

Getting a new kernel to run on the N905C will require some **bootstrapping techniques** because the device's existing bootloader (U-Boot) and OS are old. Fortunately, the i.MX507 SoC provides a few ways to load and test kernels via USB or other means, without permanently flashing the device:

- **Serial Download Mode (USB Boot):** The i.MX5 series (including i.MX507) has a built-in USB recovery mode in its ROM. If no bootable image is found (or if a specific hardware key combination is used, or if the internal storage is removed/blank), the SoC's ROM will await commands on USB. In this mode, the N905C enumerates as an USB HID device (for i.MX507 it would appear with NXP's USB VID, similar to how an i.MX6SLL shows up as "NXP Semiconductor Inc SE Blank..." device [30]). Using NXP's **USB download tools**, you can send a boot image directly to RAM. For example, the open-source **imx_usb_loader** utility allows downloading and executing code on i.MX5/6/7 via USB (Serial Download Protocol) [31]. You can use this to load a custom U-Boot or even a Linux kernel + minimal initramfs into the N905C's RAM over USB for testing. NXP's own tool **mfgtools (UUU)** is another option – it's a utility that can load U-Boot over USB; it has been used on newer Kobo models and works on i.MX50 as well [32].

  How to use it: Typically, you would short a "recovery" pin or remove internal eMMC to trigger USB boot. Then on a host PC, run `uuu` or `imx_usb` with the appropriate configuration to send an SPL/U-Boot. The ROM will load that into RAM and execute it. Once you have a custom U-Boot running in RAM, you can use **U-Boot's features** (e.g. fastboot or bootp) to load a kernel. This is essentially a way to bypass the old bootloader for testing. For instance, developers have used UUU to load Kobo bootloaders into RAM and then interact via the serial console [32]. On N905C, you would need to solder a serial console connection (there are TX/RX pads on the board) to see U-Boot's output and interact, since the device has no display output until the EPDC is initialized.

- **Fastboot Protocol:** The stock Kobo Touch bootloader (U-Boot) in older models does **not** provide an Android fastboot interface by default. However, if you replace or upgrade U-Boot (or chainload one via the USB method above), you can enable *fastboot mode*. U-Boot has support for fastboot USB protocol (commonly used on Android devices for flashing images) [33]. By enabling this, the Kobo could appear as a fastboot device on USB, allowing you to flash kernels or boot images by running `fastboot boot <kernel>` etc. This isn't something Kobo provided out-of-the-box, but it's a technique you can use in your development workflow once you have a custom U-Boot. In practice, many Kobo hackers find it simpler to use the serial download mode or to physically swap SD cards, but fastboot is an option if you invest in a custom bootloader.

- **MicroSD Boot / Dual-boot:** Another approach (not strictly USB, but useful) is to use the external microSD slot to boot alternative kernels. On some Kobo models, if you put a prepared bootable SD card in the slot, the device can boot from it (depending on boot ROM settings – some i.MX50 devices try SD cards in a certain order). At the very least, you can use the internal microSD (since it's removable) – by cloning the internal card and modifying the kernel on it. This method is a bit tedious (opening the device), but it allows testing a new kernel by swapping cards rather than touching the original. It's often used for unbricking (e.g. writing a known-good image to the internal SD).

- **Kexec from Running System:** Interestingly, the custom kernel used in InkBox/Quill OS included **kexec support** [1]. This means from the running 2.6.35 system, one could attempt to load a new kernel into memory and kexec into it (essentially performing a soft reboot into a new kernel). In theory, you could use this to test a mainline kernel without changing the bootloader at all – by copying the new kernel image to the Kobo and using a kexec utility. However, given the huge kernel version jump and device initialization needed, this may be challenging (device states might not reinitialize correctly). It's a more experimental approach compared to USB boot, but worth mentioning as a possibility for developers.

Summary of USB/Fastboot Methods: The **most direct route** for kernel development is using the i.MX507's USB recovery. Developers have successfully used it on newer Kobos; for example, on the Kobo Libra (i.MX6) one can load U-Boot into RAM via USB and then continue booting [32]. The same concept applies to N905C's i.MX507. Ensure you have a **serial console** hooked up (to view boot logs) – the Kobo Touch has UART pads internally (you'll need a 3.3V USB-UART adapter and to solder wires as documented by the community [34] [32]). Then, use the Boundary Devices **imx_usb_loader** or NXP **uuu** tool on your PC to send over an SPL or kernel. With this, you can rapidly iterate: if the kernel hangs, just power-cycle and try again without risking the on-device flash.

Lastly, keep in mind that **booting a new kernel on e-reader hardware** often requires a matching minimal **initramfs or userspace** to observe results (since there's no GUI without proper userspace). Initially, try booting to a serial console or simple init environment. Once you have the Device Tree and kernel working (you see the kernel log on serial and maybe framebuffer blanking the e-ink), you can move on to integrating it into a full OS image for the N905C.

References and Datasheets

- Freescale/NXP i.MX507 Product Summary (integrated Cortex-A8 + EPD controller) [28].
- Kobo Touch hardware information (Wikipedia and community wikis) [35] [36].
- E Ink Pearl 6″ ED060SC7 display specifications (600×800) [10].
- Neonode zForce IR touchscreen driver documentation (Device Tree binding example) [13].
- Mainline Linux Device Trees for Kobo Aura (imx50 Kobo Aura DTS) [4] [26] and i.MX50 EVK.
- Kobo Aura (N514) board features (GPIO assignments for LEDs, keys, Wi-Fi) [25] [16].
- Kobo Touch (N905) original components: PMIC (MC13892) and Samsung eMMC [18].
- Quill/InkBox OS kernel source notes (based on 2.6.35.3, difficult to upgrade) [1].
- NXP i.MX Serial Download Protocol and imx_usb utility (USB boot tool) [31].
- PostmarketOS Wiki (USB recovery with `uuu` on Kobo, serial access) [30] [32].

---

1 I made a new OS for my Kobo eReader : r/linux
https://www.reddit.com/r/linux/comments/mcv5fi/i_made_a_new_os_for_my_kobo_ereader/

2 Documenting devices with mainline Linux support - Help needed [Archive] - maemo.org - Talk
https://talk.maemo.org/archive/index.php/t-99357.html

3 4 15 16 17 19 21 25 26 Linux v6.6.1 - arch/arm/boot/dts/nxp/imx/imx50-kobo-aura.dts
https://sbexr.rabexc.org/latest/sources/3e/34ccbca812f804.html

5 14 Linux-Kernel Archive: [PATCH v2 3/3] ARM: dts: imx50: Add Kobo Aura DTS
https://lkml.iu.edu/hypermail/linux/kernel/1903.3/08386.html

6 Kobo Glo (kobo-kraken) - postmarketOS Wiki
https://wiki.postmarketos.org/wiki/Kobo_Glo_(kobo-kraken)

7 18 20 23 27 35 36 Kobo Touch – Wikipédia
https://hu.wikipedia.org/wiki/Kobo_Touch

8 [1/2] arm: dts: fix unit-address leading 0s - Patchwork
https://patches.linaro.org/project/linux-devicetree/patch/20171013175452.1198-1-robh@kernel.org/

9 11 Kobo Touch - Wikipedia
https://en.wikipedia.org/wiki/Kobo_Touch

10 EPDiy is a driver board for affordable e-Paper (or E-ink) displays.
https://github.com/vroland/epdiy

12 13 www.kernel.org
https://www.kernel.org/doc/Documentation/devicetree/bindings/input/touchscreen/zforce_ts.txt

22 28 i.MX507 Processors|Low Power Optimization | NXP Semiconductors
https://www.nxp.com/products/i.MX507

24 Kobo Touch N905 charging LED cycling between solid green and ...
https://www.reddit.com/r/kobo/comments/fh53s8/kobo_touch_n905_charging_led_cycling_between/

29 GitHub - marek-g/kobo-kernel-2.6.35.3-marek: Kernel for kobo glo, mini, touch (imx507) with changes for running X11
https://github.com/marek-g/kobo-kernel-2.6.35.3-marek

30 32 34 Kobo Libra H2O (kobo-librah2o) - postmarketOS Wiki
https://wiki.postmarketos.org/wiki/Kobo_Libra_H2O_(kobo-librah2o)

31 GitHub - boundarydevices/imx_usb_loader: USB & UART loader for i.MX5/6/7/8 series
https://github.com/boundarydevices/imx_usb_loader

33 pmsourcedump/kobo_clara_uboot: u-boot for the kobo clara hd ...
https://github.com/pmsourcedump/kobo_clara_uboot