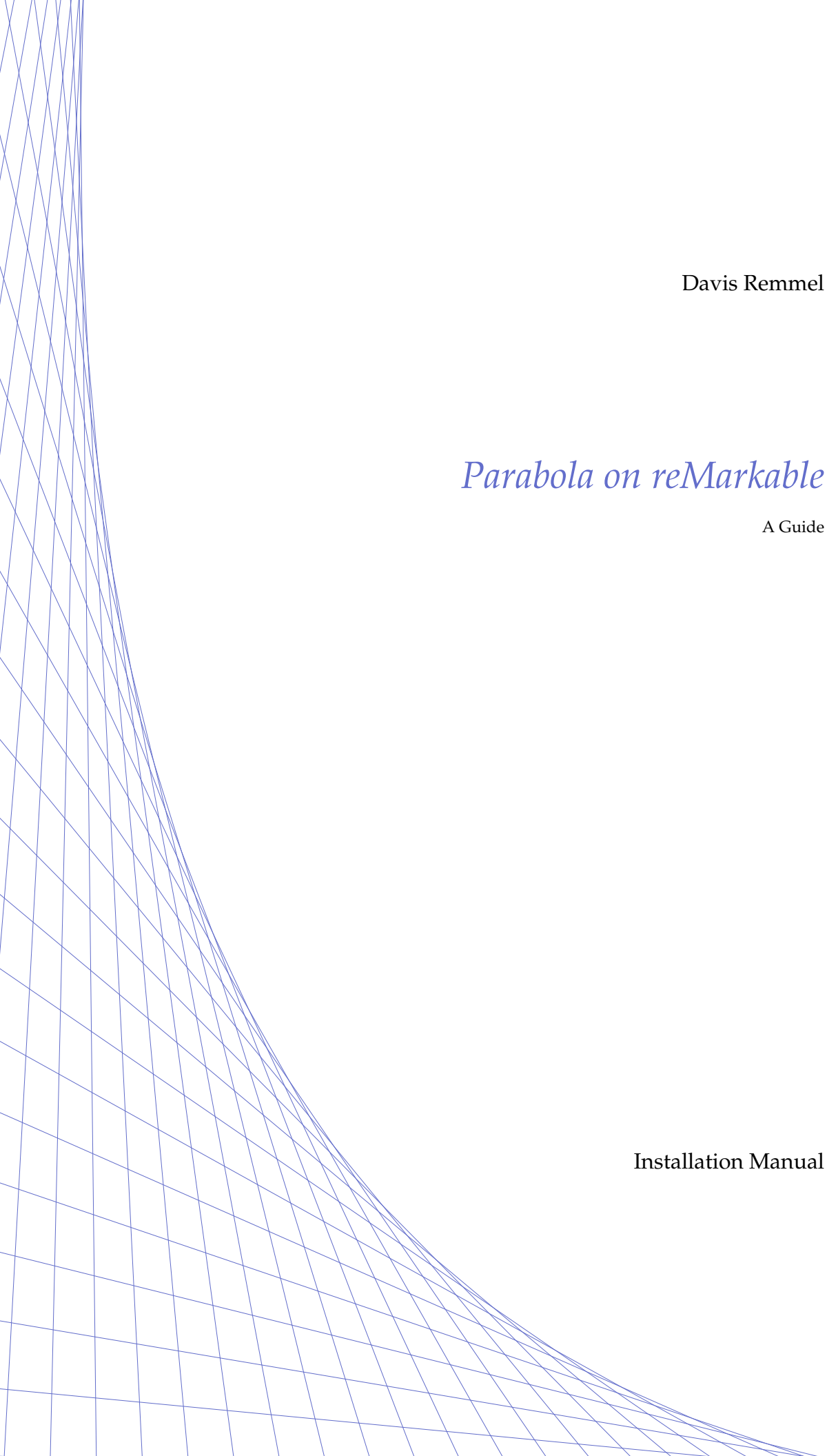

Davis Remmel

Parabola on reMarkable

A Guide

Installation Manual

Parabola on reMarkable: A Guide
August 29, 2020

Contents

Preface

You may have seen my work in the reMarkable tablet hacking community. I'm the one who added a microSD card to the tablet, then later installed a desktop GNU/Linux environment. My efforts focus on making the device usable in a general computing context.

This manual will guide a reMarkable owner through removing its proprietary software, compiling a bootloader and kernel, installing the Parabola operating system, and configuring a desktop environment. A finished OS distribution, named *Parabola-rM*, is suitable for immediate installation and available at the project's official website.

Parabola-rM is *free*, not in price, but in liberty. Under the terms of its license users hold the freedom to share this program with others, or even re-sell it. Anyone can make improvements because the source code is freely available. This viral licensing forms a web of non-proprietary software, leading the world toward transparency and trust, precipitating software *rights*.

If you are a privacy-minded individual who wants to support independent software development that represents the needs of the tablet's community, please buy a copy of this guide and Parabola-rM. The funds generated will support me through writing a non-proprietary handwriting recognition engine, eventually authoring "magic paper" software influenced by Dynabook.

I would greatly appreciate your purchase; thank you.

Davis Remmel
Maintainer

ii Support Information

This manual and the complete Parabola-rM image are available online from the official project page. This work is *not covered by any warranty*. Failure to perform these operations properly may result in a broken tablet, and performing these operations are likely to invalidate the manufacturer's warranty.

ii.1 General Support

Customers who purchase Parabola-rM from its original maintainer are entitled to some email support. The maintainer will try his best to satisfy each customer. Please write an email using the following header fields. Please reference the PayPal transaction ID in the message body.

To: Davis Remmel <d@visr.me>
Subject: Parabola-rM Support

ii.2 Getting Updates

Updates to Parabola-rM are announced via email to eligible customers. People who buy Parabola-rM from the maintainer will receive updates for six months. The OS will never stop working; updates provide improvements, but a user can never be locked out of the software they own.

Recipients may unsubscribe from update announcements by sending an email to the maintainer using the following header fields. Please reference the PayPal transaction ID in the message body.

To: Davis Remmel <d@visr.me>
Subject: Unsubscribe from Parabola-rM Update Announcements

ii.3 Bug Reporting

For advanced users who can identify a fault with the OS, please submit a bug report via email with the following header fields. In the message body, please include: (a) a description of what is seen when using the OS, (b) what is expected to be seen when using the OS, (c) steps to reproduce the problem, and (d) information about the operating system and hardware.

To: Davis Remmel <d@visr.me>
Subject: Parabola-rM Bug: *Short description of problem*

1 Introduction

This guide expects a level of familiarity from the user about using GNU/Linux utilities such as compiling software, partitioning a disk, and configuring system components. This chapter will offer context and outline the process used in this guide.

1.1 Compatibility

To install this OS requires a PC running one of the following operating systems. It is strongly recommended to use a Unix-like OS due to the use of composite USB devices, which are unsupported in Windows.

- FreeBSD 12.1
- GNU/Linux (various)
- Apple macOS 10.13
- Microsoft Windows 10

The following reMarkable models have been tested with Parabola-rM.

- RM100 (reMarkable 1)

1.2 Hardware summary

The reMarkable tablet is a portable computer disguised as a digital notebook. Its design is based on the NXP (Freescale) i.MX6 System-on-Chip (SoC) platform.

Processor	i.MX6SL (SoloLite) 800 MHz ARMv7 processor
Memory	510 MB
Storage	8 GB eMMC
Communication	Micro-USB
	Wi-Fi (with proprietary firmware)
Display	10.3 inch (diagonal), electrophoretic, 16 shades of gray
Input	Electromagnetic resonance (EMR) digitizer with stylus
	Capacitive touch overlay
	Four facial buttons (Power, Left, Center, Right)

1.3 Boot process summary

The SoC may boot from multiple sources depending on its initial state. If only the power button is pressed, the SoC will look for a bootloader on a protected eMMC partition (mounted by the kernel as */dev/mmcblk1boot0*). However, if the user holds the center facial button while the power button is pressed, the SoC will instead enter recovery mode. In this mode, the SoC will await a payload to be delivered over USB with the *imx_usb* utility. Regardless of how the bootloader gets into memory, the sequence below is followed.

1. The SoC ROM loads and executes the bootloader
2. The bootloader loads and executes the kernel
3. The kernel mounts the filesystem and starts the init program as PID 1

1.4 Installing the Parabola-rM image

One may purchase the finished Parabola-rM image from the project's maintainer. This bundle contains files named "mmcblk1boot0" and "mmcblk1" which may be directly copied over the eMMC block devices. The bundle also contains compiled versions of *imx_usb* (64-bit) for FreeBSD 12.1, GNU/Linux, and Windows 10. The rest of this document describes how to create these images from-scratch.

1.5 Installation procedure summary

The procedure to install Parabola is similar to its boot process. First, one must configure and compile the bootloader. Then, one must configure and compile the kernel. They must next partition the eMMC and copy the bootloader, kernel, and Parabola root filesystem (rootfs). Lastly, if one wants to use software designed for a desktop PC they must install a desktop environment.

1.6 Compiling imx_usb

If a user cannot compile and use *imx_usb*, they probably shouldn't follow the proceeding chapters. The reMarkable company offers their version of this utility online and includes a bootable initramfs. This initramfs may be used as a rescue utility should anything go wrong.

To load the initramfs, one must place their tablet into recovery mode. While the tablet is turned off, hold the middle facial button while pressing the power button; continue holding the middle button for five seconds while releasing the power button. The tablet should register on the PC as a USB device, "Freescale SemiConductor Inc SE Blank MEGREZ."

Execute the *imx_usb* utility. The tablet will load the initramfs and boot, then appear as a USB Ethernet device and act as a DHCP and SSH server. If the PC is configured to request a DHCP lease on this network interface, it will automatically be assigned an IP address. The tablet appears at *10.11.99.1*.

1.7 Taking and restoring a backup

SSH into the tablet as *root* and duplicate the following block devices to the local disk. Be sure to verify the checksums of these devices match.

- /dev/mmcblk1boot0
- /dev/mmcblk1boot1
- /dev/mmcblk1

Should anything go wrong, these copies may be used to restore the tablet in its original condition. It is wise to test this right away. Place the protected eMMC partitions in read-write mode with *echo 0 > /sys/block/mmcblk1bootN/force_ro*, then *dd if=/dev/zero of=/dev/mmcblk1bootN*. Read them back out and verify they contain null data. Overwrite */dev/mmcblk1* with zeros and verify it as well. Then, restore the backups from the PC, verify the checksums match, and place the protected partitions into read-only mode with *echo 1 > /sys/block/mmcblk1bootN/force_ro*. Reboot the tablet and pray that it comes back.

1.8 Compiling code for ARM

In order to compile code for the reMarkable, it is recommended to use the toolchain provided by the reMarkable company. The easiest way to get started is to create a GNU/Linux virtual machine with Trisquel or Ubuntu, and run the *poky-2.1.3.sh* script. This will install all the necessary compilers and libraries. When one wants to compile something, like U-Boot or Linux, they can simply source the toolchain's *environment-setup...gnueabi* file in their shell and the reMarkable's architecture will be targeted.

2 U-Boot Bootloader

The bootloader is loaded by the SoC ROM, and subsequently loads and executes the kernel. The reMarkable company publishes their code for U-Boot. Download it to follow along with this chapter, using the *master* branch.

In a later section, the partition map will be discussed in-depth. However, it is necessary to provide brief context for this chapter. The eMMC will have three partitions.

1. FAT, 20 MiB, U-Boot support files
2. ext4, 2 GiB, Parabola system files
3. ext4, 5.3 GiB, mounting */home*

2.1 Configuration

This section will discuss how to modify U-Boot directly, to supply new kernel boot arguments matching our disk layout[1]. U-Boot passes the kernel boot arguments, so it is necessary to adjust them.

Located within *include/configs/zero-gravitas.h* is the boot configuration. Change the root filesystem to */dev/mmcblk1p2*. Remove the boot-from-memory and boot-from-fallback parts of the *mmcboot* command, since these are no longer necessary. Finally, to prevent U-Boot from reading/saving these environment variables to disk, define *CONFIG_ENV_IS_NOWHERE*. These changes are summarized in Figure 2.1, but a full diff listing is available in the Patch: U-Boot Config appendix.

[1]The maintainer is working on getting reMarkable support upstreamed in the official U-Boot and Parabola repositories, which involves using an extlinux-style configuration file instead of hardcoding the kernel arguments.

```
"mmcargs=setenv bootargs console=${console},${baudrate} " \
     "root=/dev/mmcblk1p2 rootwait rootfstype=ext4 rw por=${por};\0" \
/* ... */
"mmcboot=echo Booting from mmc ...; " \
    "mmc dev ${mmcdev}; " \
    "if mmc rescan; then " \
        "if run loadimage; then " \
            "if run loadfdt; then " \
                "bootz ${loadaddr} - ${fdt_addr}; " \
            "else " \
                "echo WARN: Cannot load the DT; " \
            "fi; " \
        "fi; " \
    "fi;\0"
/* ... */
/* #define CONFIG_ENV_IS_IN_FAT */
#define CONFIG_ENV_IS_NOWHERE
```

Figure 2.1: Replace *mmcargs* and *mmcboot* with new parameters.

2.2 Compilation

Compile U-Boot by running *make* in its source directory. It will produce the *u-boot.imx* bootloader binary, which we will later flash to the eMMC's *boot0* protected partition.

3 Linux Kernel

The Linux kernel is responsible for providing abstraction between the userland programs and the hardware. The reMarkable company publishes their kernel online, and one must download it to proceed with this chapter. Check out the Linux 4.9 version, available in the *lars/zero-gravitas_4.9* branch.

3.1 Removing proprietary blobs

The kernel directory tree contains nonfree software, so that must be removed to respect the user's freedom. This is achieved with help from the Linux-libre deblob scripts.

Purge the *firmware* directory of all files, except for *epdc_ES103CS1.ihex* which is the EPDC (electrophoretic display controller) waveform file. This is a set of voltages, temperatures, and timings—non-executable data—so it is freedom-respecting. An easy way to achieve this is with *find . -type f | grep -v 'epdc_ES103CS1.fw.ihex' | grep -i '\.hex\|\.ihex\|\.asm\|\.H16\|\.S' | xargs rm*.

From the Linux-libre project, run the deblob-check script inside the Linux base directory. This will take a while to complete. It will likely trigger on several unimportant files that hold cryptographic lookup tables. These are fine, but do inspect that it does not trigger on any proprietary code.

3.2 EPDC framebuffer driver

In the kernel provided by the reMarkable company there is a problem with the EPDC framebuffer driver: when making lots of updates, especially in Automatic Mode, it is possible for the EPDC controller to lock up but continue to apply a voltage to the panel. This often happens when one region is updated multiple times very quickly. This condition is called a *TCE underrun*. The TCE is the Timing Control Engine, part of the EPDC, and pushes waveforms to the TFT panel on each scan cycle. TCE underruns can damage the electrophoretic display.

Max Tsai, of NXP, has published two patches that mitigate this condition. The first patch, "EPDC QoS," listed in Appendix B, removes latency for updates by increasing priority to the DDR (memory) controller. This prevents an underrun condition from happening in the first place. The second patch, "EPDC SW Reset," is listed in Appendix C. If an underrun condition occurs, the driver will drop its queue and reinitialize the panel. To the user, the screen will appear to pause for a moment, which is better than having a screen with a permanent disability.

Lastly, the framebuffer driver must be patched to enable partial refreshing with deferred-IO automatic updates[1], and to increase the interval at which the framebuffer's memory region is checked. This can be seen in Figure 3.1. REAGL waveforms are enabled to minimize ghosting.

These patches should be applied to the kernel in the framebuffer driver, located in the *drivers/video/fbdev/mxc* directory[2].

[1] It is possible to force monochrome and dithering in the ...update_pages() routine.

[2] If working from the 4.9 kernel, the Patch: EPDC QoS has been applied by the manufacturer. It is noted here as a historical reference.

```
diff --git a/drivers/video/fbdev/mxc/mxc_epdc_fb.c b/drivers/video/fbdev/mxc/mxc_epdc_fb.c
index 0f762808dd62..1c414c65cc59 100644
--- a/drivers/video/fbdev/mxc/mxc_epdc_fb.c
+++ b/drivers/video/fbdev/mxc/mxc_epdc_fb.c
@@ -3502,10 +3502,10 @@ static void mxc_epdc_fb_update_pages(struct mxc_epdc_fb_data *fb_data,
        update.update_region.top = y1;
        update.update_region.height = y2 - y1;
        update.waveform_mode = WAVEFORM_MODE_AUTO;
-       update.update_mode = UPDATE_MODE_FULL;
+       update.update_mode = UPDATE_MODE_PARTIAL;
        update.update_marker = 0;
        update.temp = TEMP_USE_AMBIENT;
-       update.flags = 0;
+       update.flags = EPDC_FLAG_USE_REGAL;

        mxc_epdc_fb_send_update(&update, &fb_data->info);
 }
@@ -3700,7 +3700,7 @@ static struct fb_ops mxc_epdc_fb_ops = {
 };

 static struct fb_deferred_io mxc_epdc_fb_defio = {
-       .delay = HZ,
+       .delay = HZ / 30,
        .deferred_io = mxc_epdc_fb_deferred_io,
 };
```

Figure 3.1: Enable automatic partial refreshing in the EPDC driver.

3.3 Configuration

To disable the compiler from using (now-deleted) firmware, it is necessary to disable some configuration options. Additionally, the default configuration provided by the reMarkable company doesn't have some desirable features. The diff shown in Figure 3.2 removes nonfree options and adds parameters to cover the EPDC deferred IO mode and new USB communications.

3.4 Compilation

Change into the base directory, run *make zero-gravitas_defconfig* to build *.config*, then run *make* to compile the kernel. The result will be two files: the kernel image at *arch/arm/boot/zImage* and the device tree binary at *arch/arm/boot/dts/zero-gravitas.dtb*.

```
diff --git a/arch/arm/configs/zero-gravitas_defconfig b/arch/arm/configs/zero-gravitas_defconfig
index 196cb9ba6223..03faf33b66d4 100644
--- a/arch/arm/configs/zero-gravitas_defconfig
+++ b/arch/arm/configs/zero-gravitas_defconfig
-CONFIG_BRCMFMAC=m
+CONFIG_BRCMFMAC=n
-CONFIG_TOUCHSCREEN_CYPRESS_CYTTSP5_LOADER=y
-CONFIG_TOUCHSCREEN_CYPRESS_CYTTSP5_BINARY_FW_UPGRADE=y
+CONFIG_TOUCHSCREEN_CYPRESS_CYTTSP5_LOADER=n
+CONFIG_TOUCHSCREEN_CYPRESS_CYTTSP5_BINARY_FW_UPGRADE=n
-CONFIG_IMX_SDMA=y
+CONFIG_IMX_SDMA=n
+CONFIG_FB_MXC_EINK_AUTO_UPDATE_MODE=y
+CONFIG_USB_ACM=y
+CONFIG_USB_F_ACM=y
+CONFIG_USB_U_SERIAL=y
+CONFIG_USB_CDC_COMPOSITE=y
```

Figure 3.2: Make these config changes in *arch/arm/config/zero-gravitas_defconfig*.

4 Parabola OS

This chapter goes through the process of installing the bootloader, kernel, and Parabola system files to the eMMC. These commands are expected to take place on the tablet with an SSH connection to the recovery initramfs.

4.1 Formatting the eMMC

In order to copy files there must be a partition table, and each partition formatted. Write the following DOS-type partition table with *fdisk* on */dev/mmcblk1*.

Partition	*Boot*	*Type*	*Size*	*Description*
1	•	W95 FAT32 (LBA)	20 MiB	U-Boot (offset 4 MiB)
2		Linux	2 GiB	Parabola
3		Linux	(fill)	Home

Format these partitions with *mkfs.vfat* and *mkfs.ext4*. The ARM processor is 32-bit, and therefore cannot use the defaults set. For better performance with the eMMC NAND layout, tune the inode parameters.

mkfs.vfat /dev/mmcblk1p1

mkfs.ext4 -O '^64bit' -O '^metadata_csum' -O 'uninit_bg' -J 'size=4' -b 1024 -g 8192 -i 4096 -I 128 /dev/mmcblk1p2

mkfs.ext4 -O '^64bit' -O '^metadata_csum' -O 'uninit_bg' -J 'size=4' -b 1024 -g 8192 -i 4096 -I 128 /dev/mmcblk1p3

4.2 Installing the OS

If one purchases a copy of this manual, it comes with a bundle of source files which are referenced in this section.

U-boot

Install the compiled bootloader onto the eMMC's *boot0* protected partition. Enable writing with *echo 0 > /sys/block/mmcblk1boot0/force_ro*. Then, zero it out with *dd if=/dev/null of=/dev/mmcblk1boot0*. Finally, install the bootloader with *dd if=u-boot.imx of=/dev/mmcblk1boot0 bs=512 seek=2* and re-enable the write lock with *echo 1 > /sys/block/mmcblk1boot0/force_ro*.

U-Boot will look for two files on the first partition (FAT): *splash.bmp* and *waveform.bin*. The former is a grayscale bitmap image shown when the device is loading. The latter makes it possible to show that image at all, because U-Boot will initialize the EPDC and signal with the waveforms in that file. The waveform file may be found in the compiled kernel target, *epdc_ES103CS1.fw*.

Mount the */dev/mmcblk1p1* partition to */mnt/p1*, and copy these files there.

Linux

Mount the */dev/mmmcblk1p2* partition to */mnt/p2*. On that partition, create the */boot* directory. From the compiled Linux kernel, copy into */boot* the *arch/arm/boot/zImage* and *arch/arm/boot/dts/zero-gravitas.dtb* files.

Parabola rootfs

Download the Parabola armv7h tarball archive from its download page. Since the tablet does not have much space, it is best to extract this through SSH with *pv parabola-tarball.tar.gz | ssh root@10.11.99.1 'tar xpf - -C /mnt/p2'*. Once extracted, most of the system is installed. However, if one tries to boot their tablet in this state, it can't yet communicate with the user.

4.3 Bootstrapping communications

The kernel will look for, and execute, the */sbin/init* script. However, the system won't do anything useful until one is able to communicate with it. Create a systemd service to enable serial communications.

cd /mnt/p2/etc/systemd/system/getty.target.wants

ln -s /usr/lib/systemd/system/serial-getty.service serial-getty@ttyGS0.service

Disable the *pam_securetty.so* module in */mnt/p2/etc/pam.d/login*, as shown in Figure 4.1. Write the */mnt/p2/etc/fstab* file, as defined in Figure 4.2. Finally, reboot, and a login prompt should appear on the USB serial console.

```
1 #%PAM-1.0
2
3 #auth       required     pam_securetty.so
4 auth       requisite    pam_nologin.so
5 auth       include      system-local-login
6 account    include      system-local-login
7 session    include      system-local-login
```

Figure 4.1: */etc/pam.d/login*

```
/dev/mmcblk1p2  /               auto    defaults                    1  1
/dev/mmcblk1p1  /var/lib/uboot  auto    defaults                    0  0
/dev/mmcblk1p3  /home           auto    defaults                    0  2
devpts  /dev/pts        devpts  mode=0620,gid=5                     0  0
proc    /proc           proc    defaults                            0  0
tmpfs   /run            tmpfs   mode=0755,nodev,nosuid,strictatime  0  0
tmpfs   /tmp            tmpfs   defaults                            0  0
tmpfs   /root/.cache    tmpfs   defaults,size=20M                    0  0
```

Figure 4.2: */etc/fstab*

4.4 Configuring the network

Because the Parabola tarball includes out-of-date software, the system must be configured with a network before it can fetch updates. It is expected the reader has some networking knowledge, and is able to set this up on their own since each PC operating system handles it differently. In the following example, FreeBSD 12.1 is used because it packages the necessary software in the base installation, so that it may route traffic. This example assumes the PC is a laptop, connected to the Internet with its *wlan0* (Wi-Fi) interface.

Configure the PC's USB Ethernet interface with a static IP address: *ifconfig ue0 10.11.99.2/24 up*.

Configure the tablet with a static IP address through a systemd service. Move */etc/systemd/network/eth0.network* to */etc/systemd/network/usb0.network*, and write into it the contents of Figure 4.3.

```
[Match]
Name=usb0

[Network]
Address=10.11.99.1/24
```

Figure 4.3: */etc/systemd/network/usb0.network*

Restart the systemd networking service with *systemctl restart systemd-networkd*, and add a temporary route through the PC with *ip route add default via 10.11.99.2*

Configure the PC's firewall to route traffic between the tablet's virtual NIC (*ue0*) and the Wi-Fi NIC (*wlan0*), as shown in Figure 4.4.

Test the network connection and name resolution with *ping fsf.org*. Once successful, set the tablet's time and date with *timedatectl set-ntp true* and *systemctl restart systemd-timedated*.

```
1 nat on wlan0 from {10.11.99.1/24} to any -> (wlan0)
```

Figure 4.4: */etc/pf.conf* on the PC

4.5 Upgrading the system software

Parabola uses *pacman* as its package manager, so users of Arch Linux should feel comfortable. Initialize the keys with *pacman-key --init*, *pacman-key --populate*, and *pacman-key --refresh-keys*[1]. The tablet uses a self-compiled kernel, so remove the *linux-libre* package with *pacman -R linux-libre*. Update the package database with *pacman -Syy*. The keyring programs may be out of date, so upgrade these first with *pacman -S archlinux-keyring parabola-keyring*. Finally, update the rest of the system software with *pacman -Syu*, and reboot.

[1]The refresh-keys command may fail due to a key not being available in a public keyserver. This message can be ignored.

4.6 Auto-networking

It may be convenient for a PC to auto-configure its network settings when the tablet is plugged in. This can be accomplished by running a DHCP server. A good program for this is *dnsmasq*: install it with *pacman -S dnsmasq* and write its configuration file to bind to the *usb0* interface, as shown in Figure 4.5. Make it run with *systemctl start dnsmasq* and set it to start at boot with *systemctl enable dnsmasq*.

```
1 interface=usb0
2 bind-interfaces
3 dhcp-range=10.11.99.2,10.11.99.253,10m
4 dhcp-option=6  # Don't send DNS
```

Figure 4.5: */etc/dnsmasq.conf*

4.7 Configuring graceful shutdowns

If one purchases a copy of this manual, it comes with support files for executing graceful shutdowns. Create a directory at */var/lib/remarkable*, copy there *epdc-show-bitmap*[2] and make that executable. Copy *splash-off.raw* to */var/lib/uboot* (to keep it next to the other splash image).

[2]See the source listing, epdc-show-bitmap.c.

A systemd service will execute before shutdown to show this splash screen and vacuum the system log. Write the primary script to */var/lib/remarkable/shutdown.sh*, as shown in Figure 4.6, and set its execution bit.

Write the contents of Figure 4.7 to */etc/systemd/system/remarkable-shutdown.service* to run it as a systemd service. Then, *systemctl start remarkable-shutdown.service* and *systemctl enable remarkable-shutdown.service*.

```
#!/usr/bin/env bash
pgrep Xorg | xargs wait
sleep 1
journalctl --vacuum-size=100M
/var/lib/remarkable/epdc-show-bitmap /var/lib/uboot/splash-off.raw
```

Figure 4.6: */var/lib/remarkable/shutdown.sh*

```
[Unit]
Description=rM shutdown helper

[Service]
Type=oneshot
RemainAfterExit=true
ExecStop=/var/lib/remarkable/shutdown.sh

[Install]
WantedBy=multi-user.target
```

Figure 4.7: */etc/systemd/system/remarkable-shutdown.service*

4.8 Fixing slow logins

There is a bug in *systemd-logind* where *pam_systemd.so* may prevent a login from happening. An error results stating, "pam_systemd(login:session): Failed to create session: Input/output error." This may be fixed by disabling *pam_systemd.so* from */etc/pam.d/system-login*, as shown in Figure 4.8.

```
...
session    optional   pam_motd.so          motd=/etc/motd
session    optional   pam_mail.so          dir=/var/spool/mail standard quiet
--session   optional   pam_systemd.so
+#-session   optional   pam_systemd.so
session    required   pam_env.so           user_readenv=1
```

Figure 4.8: */etc/pam.d/system-login*

5 Desktop Environment

In the Linux Kernel chapter, the kernel was modified to provide fast deferred IO framebuffer updates. This mechanism pushes screen updates from */dev/fb0* to the EPDC *ex post facto*. Ergo, Xorg may run without any further modifications. The window manager uses Xorg as its interface.

The desktop environment (DE) covered in this chapter is Xfce, but any DE may be used. Xfce was selected because it is relatively light, yet full-featured.

5.1 Xorg

Install Xorg through the package manager with *pacman -S xorg-server xf86-video-fbdev xf86-input-evdev*. Evdev is required to use the EMR digitizer, but libinput (installed with *xorg-server*) may be used with the other input devices.

After installation, a configuration should be written to */etc/X11/xorg.conf*[1].

Before the framebuffer can work with Xorg, it must be placed into Automatic Update mode. Copy the *epdc-init-auto*[2] program to */var/lib/remarkable*, set its execution bit, and run it. The screen should blank (go black), and then turn white again.

Test Xorg by running it with *Xorg -nocursor*. The tablet's screen should blank when it is run. Inspect */var/log/Xorg.0.log* to ensure all input devices are registered.

[1] A sample configuration file is listed in the xorg.conf appendix.

[2] The source of tihs program is listed in epdc-init-auto.c.

5.2 Xfce

Install Xfce through the package manager with *pacman -S exo garcon thunar thunar-volman tumbler xfce4-appfinder xfce4-panel xfce4-session xfce4-settings xfce4-terminal xfconf xfdesktop xfwm4 xfwm4-themes*[3].

Test Xfce by forking an Xorg process, then run *xfce4-session*. Ensure the window manager loads, then log out through *Applications→Log Out* and disable, "Save session for future logins."

[3] While Xfce is available in the *xfce* metapackage, that should not be used because it bundles a power manager which may cause issues.

5.3 Automatic loading

The DE may be configured to start at boot. This process logs *root* in automatically on *tty1*, and that user's shell profile will start the X server and desktop session.

Copy the content of Figure 5.1 into *~/.xserverrc*, Figure 5.2 into *~/.xinitrc*, and Figure 5.3 into *~/.bash_profile*.

To configure the automatic login, it is necessary to edit the *tty1* service to log *root* in without a password. Edit */etc/systemd/system/getty.target.wants/getty@tty1.service* and change it so *agetty* performs the login, shown in Figure 5.4.

```
#!/bin/sh
/var/lib/remarkable/epdc-init-auto
exec /usr/bin/Xorg -nocursor
```

Figure 5.1: *~/.xserverrc*

```
dbus-launch xfce4-session
```

Figure 5.2: *~/.xinitrc*

```
if [[ -z $DISPLAY ]] && [[ $(tty) = /dev/tty1 ]]; then
    startx
fi
```

Figure 5.3: *~/.bash_profile*

```
...
ExecStart=-/sbin/agetty -a root --noclear %I $TERM
...
```

Figure 5.4: */etc/systemd/system/getty.target.wants/getty@tty1.service*

E-paper optimization

A number of tweaks may be made to Xfce to perform better with the tablet's electrophoretic display. These optimizations increase the use of monochrome.

It is recommended to install the Onboard virtual keyboard with *pacman -S onboard ttf-dejavu*. The font is necessary to render Unicode glyphs used in the keys' labels. A virtual keyboard makes the following steps easier.

The default system does not have usable scrollbars because they are not shown full-time. Run this command to edit the GTK3 setting: *gsettings set org.gnome.desktop.interface overlay-scrolling false*. Add *export GTK_OVERLAY_SCROLLING=0* to the beginning of *~/.xinitrc* to cover GTK2.

Set the following preferences through the GUI.

Applications→Settings→Appearance

→Style

Select: High Contrast

→Icons

Select: High Contrast

→Fonts

Set Default Font: System-ui Regular
Disable: Anti-aliasing
Disable: Custom DPI scaling

→Settings

Set Toolbar Style: Text
Disable: Show images on buttons
Disable: Show images in menus

Applications→Settings→Desktop

→Background

Set Style: None
Set Color: Solid Color, White

→Icons

Disable: Show Icon Tooltips
Enable: Single click to activate items
Disable: Default Icons→Removable Devices→Other Devices

Applications→Settings→File Manager Settings

→Behavior

Enable: Single click to activate items

Applications→Settings→Panel

→Display

Set Row Size: 50 px

→Appearance

Set Fixed Icon Size: 32 px

→Items

Remove: Action Buttons
Edit: Window Buttons
Disable: Show button labels
Enable: Show flat buttons
Set Sorting Order: None, allow drag and drop
Set Window Grouping: Never

Applications→Settings→Window Manager

→Style

Set Theme: Default-xhdpi
Set Title Font: System-ui Bold

Applications→Settings→Window Manager Tweaks

→Compositor

Disable: Show shadows under dock windows
Disable: Show shadows under regular windows

Applications→System→Xfce Terminal→Edit→Preferences

→General

Set Cursor Shape: Underline

→Appearance

Enable: Use system font
Set Text Blinks: Never

→Colors

Set Presets: Black on White

5.4 Battery charge indicator

Since it is useful to know the state of charge, one may add a battery monitor to the Xfce panel with *xfce4-genmon-plugin*. After installing the plugin, write the battery monitor script as shown in Figure 5.5, then add an indicator through the GUI with the following sequence.

Applications→Settings→Panel

→Items

Add New Item: Generic Monitor

Edit: Generic Monitor

Command: /usr/sbin/battery-monitor.sh

Disable: Label

```
#!/usr/bin/env bash

# battery-monitor.sh
# Prints the state of charge of the tablet's battery
#
# Parabola-rM is a free operating system for the reMarakble tablet.
# Copyright (C) 2020  Davis Remmel
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <https://www.gnu.org/licenses/>.

# Path for Linux 4.9
battpath="/sys/class/power_supply/bq27441-0"

chargenow="$(cat $battpath/charge_now)"
chargefull="$(cat $battpath/charge_full)"
status="$(cat $battpath/status)"

chargepct="$(echo $chargenow $chargefull \
                  | awk '{printf "%f", $1 / $2 * 100}' \
                  | cut -d'.' -f1)"

symbol=""
if [[ "Charging" == "$status" ]]
then
    symbol=$'\u26a1'  # Lightning symbol
fi

echo "${symbol}${chargepct}%"
```

Figure 5.5: */usr/sbin/battery-monitor.sh*

6 Conclusion

If one followed every step in this manual, they should now have a system identical to the Parabola-rM image. This system conveys the following features.

- Working power management with splash screens
- Fast electrophoretic display refreshing
- Graphical X11 desktop environment optimized for e-paper
- Working EMR digitizer and capacitive touchscreen
- Automatic loading of window manager
- Battery charge indicator
- Facial buttons mapped to Power, Left, Home, and Right
- USB On-the-Go (OTG) peripheral support

The following features will not work because they are incompatible with the free software currently available.

- Embedded Wi-Fi networking
- Smart Direct Memory Access (SDMA)[1]

[1] For more information, see i.MX Linux Reference Manual, Chapter 3.7.

6.1 Further reading

These sites may be useful for someone installing GNU/Linux to i.MX6-based hardware.

- i.MX Linux Reference Manual (IMXLXRM)
- Parabola ARM Installation Guide
- i.MX6 SABRE Lite—Linux on ARM
- NXP Community Forum

Patch: U-Boot Config

A

```
diff --git a/include/configs/zero-gravitas.h b/include/configs/zero-gravitas.h
index 074f171422..2a64e3651c 100644
--- a/include/configs/zero-gravitas.h
+++ b/include/configs/zero-gravitas.h
@@ -71,12 +71,10 @@
        "splashimage=0x80000000\0" \
        "splashpos=m,m\0" \
        "active_partition=2\0" \
-       "fallback_partition=3\0" \
        "bootlimit=1\0" \
        "por=undefined\0" \
-       "mmcargs=setenv bootargs console=${console},${baudrate} " \
-                       "systemd.crash_reboot=true memtest " \
-                       "root=/dev/mmcblk1p${active_partition} rootwait rootfstype=ext4 quiet rw por=${po
+       "mmcargs=setenv bootargs console=${console},${baudrate} " \
+                       "root=/dev/mmcblk1p2 rootwait rootfstype=ext4 rw por=${por};\0" \
        "loadimage=ext4load mmc ${mmcdev}:${mmcpart} ${loadaddr} ${image}\0" \
        "loadfdt=ext4load mmc ${mmcdev}:${mmcpart} ${fdt_addr} ${fdt_file}\0" \
        "mmcboot=echo Booting from mmc ...; " \
@@ -89,41 +87,13 @@
                                        "echo WARN: Cannot load the DT; " \
                                "fi; " \
                        "fi; " \
-               "fi;\0" \
-       "memboot=echo Booting from memory...; " \
-               "setenv bootargs console=${console},${baudrate} " \
-               "g_mass_storage.stall=0 g_mass_storage.removable=1 " \
-               "g_mass_storage.idVendor=0x066F g_mass_storage.idProduct=0x37FF "\
-               "g_mass_storage.iSerialNumber=\"\" rdinit=/linuxrc; "\
-               "bootz ${loadaddr} ${initrd} ${fdt_addr};\0" \
-       "altbootcmd=echo Running from fallback root...; " \
-               "run memboot; " \
-               "if test ${bootcount} -gt 10; then " \
-                       "echo WARN: Failed too much, resetting bootcount and turning off; " \
-                       "setenv bootcount 0; " \
-                       "saveenv; " \
-                       "poweroff; " \
-               "fi; " \
-               "setenv mmcpart ${fallback_partition}; " \
-               "setenv bootargs console=${console},${baudrate} " \
-                               "root=/dev/mmcblk1p${fallback_partition} rootwait rootfstype=ext4 quiet r
-                               "systemd.log_level=debug systemd.log_target=kmsg memtest " \
-                               "log_buf_len=1M printk.devkmsg systemd.journald.forward_to_console=1; " \
-               "run mmcboot;\0" \
+               "fi;\0"

 /* Always try to boot from memory first, in case of USB download mode */
 #define CONFIG_BOOTCOMMAND \
-       "if test ! -e mmc 1:1 uboot.env; then " \
-               "saveenv; " \
-       "fi; " \
-       "run memboot; " \
        "run mmcargs; " \
        "setenv mmcpart ${active_partition}; " \
```

```
55 -       "run mmcboot; " \
56 -       "echo WARN: unable to boot from either RAM or eMMC; " \
57 -       "setenv upgrade_available 1; " \
58 -       "saveenv; " \
59 -       "reset; "
60 +       "run mmcboot; "
61
62  #ifdef CONFIG_BOOTDELAY
63  #undef CONFIG_BOOTDELAY
64 @@ -157,18 +127,8 @@
65  /* Environment organization */
66  #define CONFIG_ENV_SIZE                         SZ_8K
67
68 -#define CONFIG_ENV_IS_IN_FAT
69 -/*#define CONFIG_ENV_IS_NOWHERE*/
70 -
71 -#ifdef CONFIG_ENV_IS_IN_FAT
72 -#define CONFIG_BOOTCOUNT_LIMIT
73 -#define CONFIG_BOOTCOUNT_ENV
74 -
75 -#define FAT_ENV_INTERFACE "mmc"
76 -#define FAT_ENV_DEVICE_AND_PART "1:1"
77 -#define CONFIG_FAT_WRITE
78 -#define FAT_ENV_FILE "uboot.env"
79 -#endif
80 +/*#define CONFIG_ENV_IS_IN_FAT*/
81 +#define CONFIG_ENV_IS_NOWHERE
82
83  #ifdef CONFIG_CMD_SF
84  #define CONFIG_MXC_SPI
```

Patch: EPDC QoS

B

```
commit a50e901c0ea3270140e4dc6731952e6a166293e3
Author: Max Tsai <max.tsai@nxp.com>
Date:   Tue May 28 16:06:38 2019 +0800

    mx6sl - raises QoS priority to highest on EPDC reading
    stress test can pass with 1920x1920 resolution

    stress test command
    /unit_tests/mxc_epdc_fb_test.out -n 14

diff --git a/arch/arm/boot/dts/imx6sl.dtsi b/arch/arm/boot/dts/imx6sl.dtsi
index c1603dad4640..e9aa8b454c86 100644
--- a/arch/arm/boot/dts/imx6sl.dtsi
+++ b/arch/arm/boot/dts/imx6sl.dtsi
@@ -798,6 +798,11 @@
                                        clock-names = "dcp";
                                        status = "okay";
                                };
+
+                               qosc: qosc@02094000 {
+                                       compatible = "fsl,imx6sl-qosc";
+                                       reg = <0x02094000 0x4000>;
+                               };
                        };

                        aips2: aips-bus@02100000 {
diff --git a/drivers/video/fbdev/mxc/mxc_epdc_fb.c b/drivers/video/fbdev/mxc/mxc_epdc_fb.c
index 1497f728468e..d4f4778377e7 100644
--- a/drivers/video/fbdev/mxc/mxc_epdc_fb.c
+++ b/drivers/video/fbdev/mxc/mxc_epdc_fb.c
@@ -56,6 +56,16 @@

 #include "epdc_regs.h"

+#define QOS_ENABLE
+/*
+ * MMDC_MAARCR[ARCR_RCH_EN] = 1 by default
+ * QoS==’F’ is real time access
+ */
+#ifdef QOS_ENABLE
+#include <linux/of_address.h>
+#define QOS_EPDC_OFFSET        0x1400 // 0x1400 for 6SL, 0x1800 for 6SLL
+#endif
+
 /*
  * Enable this define to have a default panel
  * loaded during driver initialization
@@ -221,6 +231,9 @@ struct mxc_epdc_fb_data {
         dma_cookie_t cookie;
         struct scatterlist sg[2];
         struct mutex pxp_mutex; /* protects access to PxP */
+#ifdef QOS_ENABLE
+        void __iomem *qos_base;
+#endif
```

```
 };

 struct waveform_data_header {
@@ -1177,6 +1190,12 @@ static void epdc_init_settings(struct mxc_epdc_fb_data *fb_data)
         __raw_writel(fb_data->working_buffer_phys, EPDC_WB_ADDR);
         __raw_writel(fb_data->working_buffer_phys, EPDC_WB_ADDR_TCE);

+#ifdef QOS_ENABLE
+        u32 ot_wr, ot_rd;
+        ot_wr = __raw_readl(fb_data->qos_base + QOS_EPDC_OFFSET + 0xd0);
+        ot_rd = __raw_readl(fb_data->qos_base + QOS_EPDC_OFFSET + 0xe0);
+        dev_dbg(fb_data->dev, "EPDC QoS wr 0x%x, rd 0x%x\n", ot_wr, ot_rd);
+#endif
         /* Disable clock */
         clk_disable_unprepare(fb_data->epdc_clk_axi);
         clk_disable_unprepare(fb_data->epdc_clk_pix);
@@ -4746,6 +4765,31 @@ int mxc_epdc_fb_probe(struct platform_device *pdev)

         clk_prepare_enable(fb_data->epdc_clk_axi);
         val = __raw_readl(EPDC_VERSION);
+
+#ifdef QOS_ENABLE
+        /* axi clock must enable for EPDC QoS access */
+        u32 ot_wr, ot_rd;
+        struct device_node *np = of_find_compatible_node(NULL, NULL, "fsl,imx6sl-qosc");
+        if (!np)
+                return -EINVAL;
+        fb_data->qos_base = of_iomap(np, 0);
+        WARN_ON(!fb_data->qos_base);
+        __raw_writel(0, fb_data->qos_base);                /* disable clkgate&soft_reset */
+        __raw_writel(0, fb_data->qos_base + 0x40);        /* enable all masters */
+        __raw_writel(0, fb_data->qos_base + QOS_EPDC_OFFSET);  /* Disable clkgate & soft_rese
+        ot_wr = __raw_readl(fb_data->qos_base + QOS_EPDC_OFFSET + 0xd0);
+        ot_rd = __raw_readl(fb_data->qos_base + QOS_EPDC_OFFSET + 0xe0);
+        dev_dbg(fb_data->dev, "EPDC QoS wr 0x%x, rd 0x%x\n", ot_wr, ot_rd);
+
+        /*
+        __raw_writel(0x0f020f22, fb_data->qos_base + QOS_EPDC_OFFSET + 0xd0);
+        */
+        __raw_writel(0x0f020f22, fb_data->qos_base + QOS_EPDC_OFFSET + 0xe0);
+        ot_wr = __raw_readl(fb_data->qos_base + QOS_EPDC_OFFSET + 0xd0);
+        ot_rd = __raw_readl(fb_data->qos_base + QOS_EPDC_OFFSET + 0xe0);
+        dev_dbg(fb_data->dev, "EPDC QoS wr 0x%x, rd 0x%x\n", ot_wr, ot_rd);
+#endif
+
         clk_disable_unprepare(fb_data->epdc_clk_axi);
         fb_data->rev = ((val & EPDC_VERSION_MAJOR_MASK) >>
                                 EPDC_VERSION_MAJOR_OFFSET) * 10
```

C

Patch: EPDC SW Reset

```
From 77f3582c541d91ff864cf7e5ebbaf288daf8dcda Mon Sep 17 00:00:00 2001
From: Max Tsai <max.tsai@nxp.com>
Date: Fri, 31 May 2019 16:05:11 +0800
Subject: [PATCH] 6SL EPDC - Change SW reset flow can make it work.

---
 arch/arm/boot/dts/imx6sl.dtsi         |   4 +-
 drivers/video/fbdev/mxc/mxc_epdc_fb.c | 341 ++++++++++++++++++++++++++++++++++-
 2 files changed, 339 insertions(+), 6 deletions(-)

diff --git a/arch/arm/boot/dts/imx6sl.dtsi b/arch/arm/boot/dts/imx6sl.dtsi
index c1603dad4640..d1e499729420 100644
--- a/arch/arm/boot/dts/imx6sl.dtsi
+++ b/arch/arm/boot/dts/imx6sl.dtsi
@@ -772,8 +772,8 @@
                                 compatible = "fsl,imx6sl-epdc", "fsl,imx6dl-epdc";
                                 reg = <0x020f4000 0x4000>;
                                 interrupts = <0 97 IRQ_TYPE_LEVEL_HIGH>;
-                                clocks = <&clks IMX6SL_CLK_EPDC_AXI>, <&clks IMX6SL_CLK_EPDC_PIX>;
-                                clock-names = "epdc_axi", "epdc_pix";
+                                clocks = <&clks IMX6SL_CLK_EPDC_AXI>, <&clks IMX6SL_CLK_EPDC_PIX>, <&clk
+                                clock-names = "epdc_axi", "epdc_pix", "epdc_pix_podf";
                         };

                         lcdif: lcdif@020f8000 {
diff --git a/drivers/video/fbdev/mxc/mxc_epdc_fb.c b/drivers/video/fbdev/mxc/mxc_epdc_fb.c
index 1497f728468e..5c7dc925fb2a 100644
--- a/drivers/video/fbdev/mxc/mxc_epdc_fb.c
+++ b/drivers/video/fbdev/mxc/mxc_epdc_fb.c
@@ -20,7 +20,6 @@
  * Based on STMP378X LCDIF
  * Copyright 2008 Embedded Alley Solutions, Inc All Rights Reserved.
  */
-
 #include <linux/busfreq-imx.h>
 #include <linux/module.h>
 #include <linux/kernel.h>
@@ -56,6 +55,8 @@

 #include "epdc_regs.h"

+#define SW_RESET
+
 /*
  * Enable this define to have a default panel
  * loaded during driver initialization
@@ -221,6 +222,9 @@ struct mxc_epdc_fb_data {
         dma_cookie_t cookie;
         struct scatterlist sg[2];
         struct mutex pxp_mutex; /* protects access to PxP */
+
+        bool in_recovery;
+        bool is_drop_updates;
 };
```

```
55
56  struct waveform_data_header {
57 @@ -254,6 +258,23 @@ struct mxcfb_waveform_data_file {
58          u32 *data;        /* Temperature Range Table + Waveform Data */
59  };
60
61 +static struct fb_videomode virtual_test_mode = {
62 +        .name="E60_V110",
63 +        .refresh=85,
64 +        .xres=1920,
65 +        .yres=1440,
66 +        .pixclock=160000000,
67 +        .left_margin=36,
68 +        .right_margin=248,
69 +        .upper_margin=4,
70 +        .lower_margin=8,
71 +        .hsync_len=52,
72 +        .vsync_len=1,
73 +        .sync=0,
74 +        .vmode=FB_VMODE_NONINTERLACED,
75 +        .flag=0,
76 +};
77 +
78  static struct fb_videomode e60_v110_mode = {
79          .name = "E60_V110",
80          .refresh = 50,
81 @@ -323,6 +344,19 @@ static struct fb_videomode e97_v110_mode = {
82  };
83
84  static struct imx_epdc_fb_mode panel_modes[] = {
85 +        {
86 +                &virtual_test_mode,
87 +                4,          /* vscan_holdoff */
88 +                10,          /* sdoed_width */
89 +                20,          /* sdoed_delay */
90 +                10,          /* sdoez_width */
91 +                20,          /* sdoez_delay */
92 +                972,          /* GDCLK_HP */
93 +                721,          /* GDSP_OFF */
94 +                0,          /* GDOE_OFF */
95 +                71,          /* gdclk_offs */
96 +                1,          /* num_ce */
97 +        },
98          {
99                  &e60_v110_mode,
100                 4,      /* vscan_holdoff */
101 @@ -413,6 +447,8 @@ static void do_dithering_processing_Y4_v1_0(
102                 unsigned long update_region_stride,
103                 int *err_dist);
104
105 +static void epdc_recover(struct mxc_epdc_fb_data *fb_data);
106 +
107  #ifdef DEBUG
108  static void dump_pxp_config(struct mxc_epdc_fb_data *fb_data,
109                              struct pxp_config_data *pxp_conf)
110 @@ -605,7 +641,7 @@ static void dump_pending_list(struct mxc_epdc_fb_data *fb_data)
111  {
```

```
112          struct update_desc_list *plist;
113
114 -        dev_info(fb_data->dev, "Queue:\n");
115 +        dev_info(fb_data->dev, "Pending Queue:\n");
116          if (list_empty(&fb_data->upd_pending_list))
117                  dev_info(fb_data->dev, "Empty");
118          list_for_each_entry(plist, &fb_data->upd_pending_list, list)
119 @@ -616,6 +652,7 @@ static void dump_all_updates(struct mxc_epdc_fb_data *fb_data)
120  {
121          dump_free_list(fb_data);
122          dump_queue(fb_data);
123 +        dump_pending_list(fb_data);
124          dump_collision_list(fb_data);
125          dev_info(fb_data->dev, "Current update being processed:\n");
126          if (fb_data->cur_update == NULL)
127 @@ -1014,10 +1051,26 @@ static void epdc_init_settings(struct mxc_epdc_fb_data *fb_data)
128          __raw_writel(EPDC_CTRL_SFTRST, EPDC_CTRL_SET);
129          while (!(__raw_readl(EPDC_CTRL) & EPDC_CTRL_CLKGATE))
130                  ;
131 -        __raw_writel(EPDC_CTRL_SFTRST, EPDC_CTRL_CLEAR);
132
133 +        /* Let finish the process */
134 +        udelay(200);
135 +
136 +        /*
137 +         * synchronization from system clock to pixel clock of
138 +         * underrun flag has not included a sw reset.
139 +         * Due to the gap between sw_reset_sys_clock and
140 +         * sw_rest_pixel_clock, there maybe one underrun flag
141 +         * after sw_reset_sys_clock then remain to next restart
142 +         * due to the lack of sw_reset for the crossing signals.
143 +         * This unexpected residual underrun flag will cause
144 +         * the lock-up after restart.
145 +         * change the restart flow to release clock before
146 +         * release sw_reset to workaround the issue.
147 +         */
148          /* Enable clock gating (clear to enable) */
149          __raw_writel(EPDC_CTRL_CLKGATE, EPDC_CTRL_CLEAR);
150 +        udelay(100);
151 +        __raw_writel(EPDC_CTRL_SFTRST, EPDC_CTRL_CLEAR);
152          while (__raw_readl(EPDC_CTRL) & (EPDC_CTRL_SFTRST | EPDC_CTRL_CLKGATE))
153                  ;
154
155 @@ -2401,6 +2454,10 @@ static void epdc_submit_work_func(struct work_struct *work)
156          int *err_dist;
157          int ret;
158
159 +        /* just drop out when reset */
160 +        if (fb_data->is_drop_updates)
161 +                return 0;
162 +
163          /* Protect access to buffer queues and to update HW */
164          mutex_lock(&fb_data->queue_mutex);
165
166 @@ -3197,6 +3254,10 @@ int mxc_epdc_fb_wait_update_complete(struct mxcfb_update_marker_data *marker_dat
167          if (!ret) {
168                  dev_err(fb_data->dev,
```

```
169                         "Timed out waiting for update completion\n");
170 +#ifdef SW_RESET
171 +                 /* controller locked-up recover the HW */
172 +                 epdc_recover(fb_data);
173 +#endif
174                   return -ETIMEDOUT;
175           }
176
177 @@ -3274,6 +3335,10 @@ static int mxc_epdc_fb_ioctl(struct fb_info *info, unsigned int cmd,
178                   {
179                           struct mxcfb_update_data upd_data;
180
181 +                         /* just drop out when reset */
182 +                         if (g_fb_data->is_drop_updates)
183 +                                 return 0;
184 +
185                           if (mutex_lock_interruptible(&hard_lock) < 0)
186                                   return -ERESTARTSYS;
187
188 @@ -3451,9 +3516,14 @@ void mxc_epdc_fb_flush_updates(struct mxc_epdc_fb_data *fb_data)
189                   /* Wait for any currently active updates to complete */
190                   ret = wait_for_completion_timeout(&fb_data->updates_done,
191                                                     msecs_to_jiffies(8000));
192 -                 if (!ret)
193 +                 if (!ret) {
194                           dev_err(fb_data->dev,
195                                   "Flush updates timeout! ret = 0x%x\n", ret);
196 +#ifdef SW_RESET
197 +                         /* Recover EPDC */
198 +                         epdc_recover(fb_data);
199 +#endif
200 +                 }
201
202                   mutex_lock(&fb_data->queue_mutex);
203                   fb_data->waiting_for_idle = false;
204 @@ -3611,6 +3681,7 @@ static bool is_free_list_full(struct mxc_epdc_fb_data *fb_data)
205                   return false;
206  }
207
208 +
209  static irqreturn_t mxc_epdc_irq_handler(int irq, void *dev_id)
210  {
211           struct mxc_epdc_fb_data *fb_data = dev_id;
212 @@ -3661,6 +3732,8 @@ static irqreturn_t mxc_epdc_irq_handler(int irq, void *dev_id)
213                   epdc_eof_intr(false);
214                   epdc_clear_eof_irq();
215                   complete(&fb_data->eof_event);
216 +                 if (fb_data->in_recovery)
217 +                         return IRQ_HANDLED;
218           }
219
220           /*
221 @@ -4480,6 +4553,263 @@ static const struct of_device_id imx_epdc_dt_ids[] = {
222  };
223  MODULE_DEVICE_TABLE(of, imx_epdc_dt_ids);
224
225 +static void drop_all_upates(struct mxc_epdc_fb_data *fb_data)
```

```
+{
+	struct update_data_list *collision_update, *temp_update;
+	struct update_marker_data *next_marker, *temp_marker;
+	struct update_desc_list *next_desc, *temp_desc;
+
+	struct update_data_list *plist;
+	int colli_count = 0, free_count = 0;;
+
+	list_for_each_entry(plist, &fb_data->upd_buf_free_list, list)
+		free_count++;
+
+	/* Clean-up collision list */
+	list_for_each_entry_safe(collision_update, temp_update,
+		&fb_data->upd_buf_collision_list, list) {
+
+		dev_dbg(fb_data->dev, "Flushing collision update!\n");
+		if (collision_update->update_desc) {
+			list_del_init(&collision_update->update_desc->list);
+			kfree(collision_update->update_desc);
+		}
+		collision_update->update_desc = NULL;
+		list_del_init(&collision_update->list);
+		/* Add to free buffer list */
+		list_add_tail(&collision_update->list,
+			&fb_data->upd_buf_free_list);
+		colli_count++;
+	}
+
+	/* clean-up pending update list */
+	list_for_each_entry_safe(next_desc, temp_desc,
+		&fb_data->upd_pending_list, list) {
+		list_del_init(&next_desc->list);
+		kfree(next_desc);
+	}
+	/* Clean-up current update and any queued in upd_buf_queue */
+	if (fb_data->cur_update == NULL) {
+		/* Process next item in update list */
+		if (!list_empty(&fb_data->upd_buf_queue)) {
+			fb_data->cur_update =
+				list_entry(fb_data->upd_buf_queue.next,
+					struct update_data_list, list);
+			list_del_init(&fb_data->cur_update->list);
+		}
+	}
+	while (fb_data->cur_update != NULL) {
+
+		list_for_each_entry_safe(next_marker, temp_marker,
+			&fb_data->cur_update->update_desc->upd_marker_list,
+			upd_list) {
+
+			/* Del from per-update & full list */
+			list_del_init(&next_marker->upd_list);
+			list_del_init(&next_marker->full_list);
+
+			/* Signal completion of update */
+			dev_info(fb_data->dev,
+				"Signaling marker (cancelled) %d\n",
```

```
283 +                                next_marker->update_marker);
284 +                        kfree(next_marker);
285 +                }
286 +
287 +                /* Free marker list and update descriptor */
288 +                kfree(fb_data->cur_update->update_desc);
289 +                fb_data->cur_update->update_desc = NULL;
290 +                /* Add to free buffer list */
291 +                list_add_tail(&fb_data->cur_update->list,
292 +                                &fb_data->upd_buf_free_list);
293 +                /* Process next item in update list */
294 +                if (!list_empty(&fb_data->upd_buf_queue)) {
295 +                        fb_data->cur_update =
296 +                                list_entry(fb_data->upd_buf_queue.next,
297 +                                                struct update_data_list, list);
298 +                        list_del_init(&fb_data->cur_update->list);
299 +                }
300 +                else {
301 +                        /* Clear current update */
302 +                        fb_data->cur_update = NULL;
303 +                }
304 +        }
305 +
306 +        /* Clean-up full_marker_list */
307 +        list_for_each_entry_safe(next_marker, temp_marker,
308 +                        &fb_data->full_marker_list,
309 +                        full_list) {
310 +
311 +                /* Found marker to signal - remove from list */
312 +                list_del_init(&next_marker->full_list);
313 +
314 +                /* Signal completion of update */
315 +                dev_info(fb_data->dev, "Signaling marker %d\n",
316 +                        next_marker->update_marker);
317 +                if (next_marker->waiting)
318 +                        complete(&next_marker->update_completion);
319 +                else
320 +                        kfree(next_marker);
321 +        }
322 +        dev_dbg(fb_data->dev, "cur_update 0x%p, free %d, colli %d\n",
323 +                        fb_data->cur_update,
324 +                        free_count,
325 +                        colli_count);
326 +}
327 +
328 +/* routine to recover EPDC */
329 +static void epdc_recover(struct mxc_epdc_fb_data *fb_data)
330 +{
331 +        struct mxcfb_update_data update;
332 +        int ret = 0;
333 +        struct clk * pixel_clk_src;
334 +        int i;
335 +
336 +        dev_info(fb_data->dev, "In epdc_recover routine.\n");
337 +
338 +        if (fb_data->in_recovery) {
339 +                return;
```

```
340 +         }
341 +
342 +         fb_data->is_drop_updates = true;
343 +
344 +         pixel_clk_src = clk_get(fb_data->dev, "epdc_pix_podf");
345 +         if (IS_ERR(pixel_clk_src)) {
346 +                 dev_info(fb_data->dev, "Unable to get EPDC pixel source clk."
347 +                         "err = %d\n", (int)pixel_clk_src);
348 +         }
349 +
350 +         /* disable IRQ and stop receiving updates */
351 +         disable_irq(fb_data->epdc_irq);
352 +         fb_data->in_recovery = true;
353 +
354 +         /* Enable clocks to access EPDC regs */
355 +         clk_prepare_enable(fb_data->epdc_clk_axi);
356 +         clk_prepare_enable(fb_data->epdc_clk_pix);
357 +
358 +         dev_err(fb_data->dev, "Before EPDC reset\n");
359 +         dump_epdc_reg();
360 +         dev_info(fb_data->dev, " LUT_status=0x%08x/0x%08x\n",
361 +                 __raw_readl(EPDC_STATUS_LUTS), __raw_readl(EPDC_STATUS_LUTS2));
362 +
363 +         /* lock queues */
364 +         mutex_lock(&fb_data->queue_mutex);
365 +         /* check if TCE is still running using FRAME_END IRQ */
366 +         init_completion(&fb_data->eof_event);
367 +         __raw_writel(EPDC_IRQ_FRAME_END_IRQ, EPDC_IRQ_MASK);
368 +         __raw_writel(0xFFFFFFFF, EPDC_IRQ_MASK1_CLEAR);
369 +         __raw_writel(0xFFFFFFFF, EPDC_IRQ_MASK2_CLEAR);
370 +         enable_irq(fb_data->epdc_irq);
371 +
372 +         ret = wait_for_completion_timeout(&fb_data->eof_event,
373 +                 msecs_to_jiffies(50));
374 +         if (!ret) {
375 +                 dev_info(fb_data->dev, "TCE is not active.!\n");
376 +         } else {
377 +                 /* TCE is active so don't reset during VSHOLD_OFF period */
378 +                 udelay(fb_data->eof_sync_period);
379 +         }
380 +         disable_irq(fb_data->epdc_irq);
381 +
382 +         if (!IS_ERR(pixel_clk_src)) {
383 +                 clk_disable_unprepare(pixel_clk_src);
384 +         }
385 +
386 +         /* Hold EPDC under Reset until we clean-up */
387 +         __raw_writel(EPDC_CTRL_SFTRST | EPDC_CTRL_CLKGATE, EPDC_CTRL_SET);
388 +
389 +         /* Disable clock */
390 +         clk_disable_unprepare(fb_data->epdc_clk_axi);
391 +         clk_disable_unprepare(fb_data->epdc_clk_pix);
392 +
393 +         drop_all_upates(fb_data);
394 +
395 +         /* set flags appropriate state */
396 +         fb_data->order_cnt = 0;
```

```
397 +         fb_data->updates_active = false;
398 +         fb_data->luts_complete_wb = ~(0x0ull);
399 +
400 +         for (i = 0; i < fb_data->num_luts; i++)
401 +                 fb_data->lut_update_order[i] = 0;
402 +         fb_data->epdc_colliding_luts = 0;
403 +
404 +         /* Signal completion if submit workqueue needs a LUT */
405 +         if (fb_data->waiting_for_lut ||  fb_data->waiting_for_wb) {
406 +                 complete(&fb_data->update_res_free);
407 +                 fb_data->waiting_for_lut = false;
408 +                 fb_data->waiting_for_wb = false;
409 +         }
410 +
411 +         /* Signal completion if LUT15 free and is needed */
412 +         if (fb_data->waiting_for_lut15) {
413 +                 complete(&fb_data->lut15_free);
414 +                 fb_data->waiting_for_lut15 = false;
415 +         }
416 +         if (fb_data->waiting_for_idle) {
417 +                 complete(&fb_data->updates_done);
418 +                 fb_data->waiting_for_idle = false;
419 +         }
420 +
421 +         /* unlock queue */
422 +         mutex_unlock(&fb_data->queue_mutex);
423 +
424 +         if (!IS_ERR(pixel_clk_src)) {
425 +                 clk_prepare_enable(pixel_clk_src);
426 +         }
427 +
428 +         /* recover HW - reset and init again */
429 +         epdc_init_settings(fb_data);
430 +
431 +         enable_irq(fb_data->epdc_irq);
432 +
433 +         /* Now, request full screen update */
434 +         memset(&update, 0x0, sizeof(struct mxcfb_update_data));
435 +         update.update_region.left = 0;
436 +         update.update_region.width = fb_data->epdc_fb_var.xres;
437 +         update.update_region.top = 0;
438 +         update.update_region.height = fb_data->epdc_fb_var.yres;
439 +         update.update_mode = UPDATE_MODE_FULL;
440 +         update.temp = TEMP_USE_AMBIENT;
441 +         update.waveform_mode = fb_data->wv_modes.mode_gc4;
442 +         update.update_marker = 0;
443 +         update.flags = 0;
444 +
445 +         mxc_epdc_fb_send_update(&update, (struct fb_info *)fb_data);
446 +
447 +         /* restore driver to operational state before returning */
448 +         fb_data->in_recovery = false;
449 +
450 +         fb_data->is_drop_updates = false;
451 +}
452 +
453 +static ssize_t store_reset(struct device *device,
```

```
+                               struct device_attribute *attr,
+                               const char *buf, size_t count)
+{
+        struct fb_info *info = dev_get_drvdata(device);
+        struct mxc_epdc_fb_data *fb_data = (struct mxc_epdc_fb_data *)info;
+        epdc_recover(fb_data);
+        return count;
+}
+
+static ssize_t show_reset(struct device *dev, struct device_attribute *attr, char *buf)
+{
+        struct fb_info *info = dev_get_drvdata(dev);
+        struct mxc_epdc_fb_data *fb_data = (struct mxc_epdc_fb_data *)info;
+        int count = 0;
+        struct update_data_list *plist;
+        //dump_epdc_reg();
+        dump_all_updates(fb_data);
+        /* Count buffers in free buffer list */
+        list_for_each_entry(plist, &fb_data->upd_buf_free_list, list)
+                count++;
+        dev_info(fb_data->dev, "Free list count: %d\n", count);
+        return 0;
+}
+
+static struct device_attribute epdc_attrs[] = {
+        __ATTR(reset, S_IRUGO|S_IWUSR, show_reset, store_reset),
+};
+
 int mxc_epdc_fb_probe(struct platform_device *pdev)
 {
         int ret = 0;
@@ -4933,6 +5263,9 @@ int mxc_epdc_fb_probe(struct platform_device *pdev)
         if (device_create_file(info->dev, &fb_attrs[0]))
                 dev_err(&pdev->dev, "Unable to create file from fb_attrs\n");

+        if (device_create_file(&pdev->dev, &epdc_attrs[0]))
+                dev_err(&pdev->dev, "Unable to create file from epdc_attrs\n");
+
         fb_data->cur_update = NULL;

         mutex_init(&fb_data->queue_mutex);
--
2.14.2
```

D

xorg.conf

```
Section "ServerLayout"
        Identifier     "reMarkable Tablet RM100"
        Screen      0  "Screen0"
        InputDevice    "wacom" "CorePointer"
EndSection

Section "ServerFlags"
    Option "BlankTime" "0"
    Option "StandbyTime" "0"
    Option "SuspendTime" "0"
    Option "OffTime" "0"
EndSection

Section "Monitor"
        Identifier "Monitor0"
        DisplaySize 210 158 # mm, sets DPI
        Modeline "1872x1404_30.00" 104.26 1872 1960 2152 2432 1404 1405 \
                 1408 1429 -HSync +Vsync
        Option "PreferredMode" "1872x1404_30.00"
EndSection

Section "Screen"
        Identifier "Screen0"
        Monitor "Monitor0"
        Device "epdc0"
        DefaultDepth 16
        SubSection "Display"
                Depth 16
                Modes "1872x1404_30.00"
        EndSubSection
EndSection

Section "Device"
    Identifier "epdc0"
    Driver "fbdev"
    Option "fbdev" "/dev/fb0"
    Option "Rotate" "CW"
EndSection

Section "InputDevice"
    Identifier "wacom"
    Driver "evdev"
    Option "Protocol" "Auto"
    Option "Device" "/dev/input/event0"
    Option "SwapAxes" "1"
    Option "InvertY" "1"
EndSection

Section "InputClass"
    Identifier "touchscreen"
    MatchIsTouchscreen "on"
    MatchDevicePath "/dev/input/event1"
    Driver "libinput"
    Option "CalibrationMatrix" "-1 0 1 0 -1 1 0 0 1"  # Rot 180
```

```
55 EndSection
56
57 Section "InputClass"
58     Identifier "facialbuttons"
59     MatchIsKeyboard "on"
60     MatchDevicePath "/dev/input/event2"
61     Driver "libinput"
62 EndSection
```

E

epdc-init-auto.c

```c
/*
  epdc-init-auto.c
  Initializes the EPDC framebuffer into a deferred-IO automatic-update
  mode

  Parabola-rM is a free operating system for the reMarakble tablet.
  Copyright (C) 2020  Davis Remmel

  This program is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include <sys/ioctl.h>
#include <linux/fb.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/mman.h>
#include "mxcfb.h"
#include <stdio.h>

int main()
{
        int ret;
        int fb = open("/dev/fb0", O_RDWR);
        struct fb_var_screeninfo vinfo;
        ret = ioctl(fb, FBIOGET_VSCREENINFO, &vinfo);
        if (0 != ret) {
                fprintf(stderr, "FBIOGET_VSCREENINFO failed with error "
                        "%d, aborting\n", ret);
                return 1;
        }

        vinfo.xres = 1872;
        vinfo.yres = 1404;
        vinfo.pixclock = 160000000;
        vinfo.left_margin = 32;
        vinfo.right_margin = 326;
        vinfo.upper_margin = 4;
        vinfo.lower_margin = 12;
        vinfo.hsync_len = 44;
        vinfo.vsync_len = 1;
        vinfo.sync = 0;
        vinfo.vmode = FB_VMODE_NONINTERLACED;
```

```
55          vinfo.accel_flags = 0;
56          vinfo.activate = FB_ACTIVATE_FORCE;
57
58          // Put screen info. Sometimes this fails when trying to set the
59          // pixclock. This may be a bug in the driver's arithmetic.
60          ret = ioctl(fb, FBIOPUT_VSCREENINFO, &vinfo);
61          if (0 != ret) {
62                  fprintf(stderr, "FBIOPUT_VSCREENINFO failed with error "
63                          "%d, attempting to reset pixclock\n", ret);
64                  vinfo.pixclock = 6250;
65                  ioctl(fb, FBIOPUT_VSCREENINFO, &vinfo);
66                  vinfo.pixclock = 160000000;
67                  ret = ioctl(fb, FBIOPUT_VSCREENINFO, &vinfo);
68                  if (0 != ret) {
69                          fprintf(stderr, "FBIOPUT_VSCREENINFO failed "
70                                  "with error %d, aborting\n", ret);
71                          return 1;
72                  }
73          }
74
75          // Pull the screeninfo agian
76          ret = ioctl(fb, FBIOGET_VSCREENINFO, &vinfo);
77          if (0 != ret) {
78                  fprintf(stderr, "FBIOGET_VSCREENINFO failed with error "
79                          "%d, aborting\n", ret);
80                  return 1;
81          }
82
83          printf("x:%d y:%d activate:%d bpp:%d rotate:%d hsync_len:%d"
84                 "vsync_len: %d sync:%d\n",
85                 vinfo.xres, vinfo.yres, vinfo.activate,
86                 vinfo.bits_per_pixel, vinfo.rotate, vinfo.hsync_len,
87                 vinfo.vsync_len, vinfo.sync);
88
89          struct fb_fix_screeninfo finfo;
90          ret = ioctl(fb, FBIOGET_FSCREENINFO, &finfo);
91          if (0 != ret) {
92                  fprintf(stderr, "FBIOGET_FSCREENINFO failed with error "
93                          "%d, aborting\n", ret);
94                  return 1;
95          }
96
97          // In case the EPDC wasn't accessible
98          ret = ioctl(fb, MXCFB_ENABLE_EPDC_ACCESS);
99          if (0 != ret) {
100                 fprintf(stderr, "MXCFB_ENABLE_EPDC_ACCESS failed with "
101                         "error %d, aborting\n", ret);
102                 return 1;
103         }
104
105         // Set auto update mode
106         __u32 aumode = AUTO_UPDATE_MODE_AUTOMATIC_MODE;
107         ret = ioctl(fb, MXCFB_SET_AUTO_UPDATE_MODE, &aumode);
108         if (0 != ret) {
109                 fprintf(stderr, "MXCFB_SET_AUTO_UPDATE_MODE failed "
110                         "with error %d, aborting\n", ret);
111                 return 1;
```

```
112         }
113
114         // Queue-and-merge is best-performing
115         __u32 uscheme = UPDATE_SCHEME_QUEUE_AND_MERGE;
116         ret = ioctl(fb, MXCFB_SET_UPDATE_SCHEME, &uscheme);
117         if (0 != ret) {
118                 fprintf(stderr, "MXCFB_SET_UPDATE_SCHEME failed with "
119                         "error %d, aborting\n", ret);
120                 return 1;
121         }
122
123         close(fb);
124         return 0;
125 }
```

F

epdc-show-bitmap.c

```c
/*
  epdc-show-bitmap.c
  Displays a raw image to the EPDC framebuffer

  Parabola-rM is a free operating system for the reMarakble tablet.
  Copyright (C) 2020  Davis Remmel

  This program is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#include <sys/ioctl.h>
#include <linux/fb.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/mman.h>
#include "mxcfb.h"
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
        if (argc < 2) {
                printf("Must pass an image as an argument.\n");
                return 1;
        }

        int ret;
        int fb = open("/dev/fb0", O_RDWR);
        struct fb_var_screeninfo vinfo;
        ret = ioctl(fb, FBIOGET_VSCREENINFO, &vinfo);
        if (0 != ret) {
                printf("FBIOGET_VSCREENINFO failed with error %d"
                       ", aborting\n", ret);
                return 1;
        }

        vinfo.xres = 1872;
        vinfo.yres = 1404;
        vinfo.pixclock = 160000000;
        vinfo.left_margin = 32;
```

```
55          vinfo.right_margin = 326;
56          vinfo.upper_margin = 4;
57          vinfo.lower_margin = 12;
58          vinfo.hsync_len = 44;
59          vinfo.vsync_len = 1;
60          vinfo.sync = 0;
61          vinfo.vmode = FB_VMODE_NONINTERLACED;
62          vinfo.accel_flags = 0;
63          vinfo.activate = FB_ACTIVATE_FORCE;
64
65          // Put screen info. Sometimes this fails when trying to set the
66          // pixclock. This may be a bug in the driver's arithmetic.
67          ret = ioctl(fb, FBIOPUT_VSCREENINFO, &vinfo);
68          if (0 != ret) {
69                  fprintf(stderr, "FBIOPUT_VSCREENINFO failed with error "
70                          "%d, attempting To reset pixclock\n", ret);
71                  vinfo.pixclock = 6250;
72                  ioctl(fb, FBIOPUT_VSCREENINFO, &vinfo);
73                  vinfo.pixclock = 160000000;
74                  ret = ioctl(fb, FBIOPUT_VSCREENINFO, &vinfo);
75                  if (0 != ret) {
76                          fprintf(stderr, "FBIOPUT_VSCREENINFO failed "
77                                  "with error %d, aborting\n", ret);
78                          return 1;
79                  }
80          }
81
82          printf("x:%d y:%d activate:%d bpp:%d rotate:%d hsync_len:%d"
83                 "vsync_len: %d sync:%d\n",
84                 vinfo.xres, vinfo.yres, vinfo.activate,
85                 vinfo.bits_per_pixel, vinfo.rotate, vinfo.hsync_len,
86                 vinfo.vsync_len, vinfo.sync);
87
88          struct fb_fix_screeninfo finfo;
89          ret = ioctl(fb, FBIOGET_FSCREENINFO, &finfo);
90          if (0 != ret) {
91                  fprintf(stderr, "FBIOGET_FSCREENINFO failed with error "
92                          "%d, aborting\n", ret);
93                  return 1;
94          }
95
96          // In case the EPDC wasn't accessible
97          ret = ioctl(fb, MXCFB_ENABLE_EPDC_ACCESS);
98          if (0 != ret) {
99                  fprintf(stderr, "MXCFB_ENABLE_EPDC_ACCESS failed with "
100                         "error %d, aborting\n", ret);
101                 return 1;
102         }
103
104         // Set to partial mode to control update parameters
105         __u32 aumode = AUTO_UPDATE_MODE_REGION_MODE;
106         ret = ioctl(fb, MXCFB_SET_AUTO_UPDATE_MODE, &aumode);
107         if (0 != ret) {
108                 fprintf(stderr, "MXCFB_SET_AUTO_UPDATE_MODE failed "
109                         "with error %d, aborting\n", ret);
110                 return 1;
111         }
```

```
112
113         // No artifacts in display output
114         __u32 uscheme = UPDATE_SCHEME_SNAPSHOT;
115         ret = ioctl(fb, MXCFB_SET_UPDATE_SCHEME, &uscheme);
116         if (0 != ret) {
117                 fprintf(stderr, "MXCFB_SET_UPDATE_SCHEME failed with "
118                         "error %d, aborting\n", ret);
119                 return 1;
120         }
121
122         // Set up update (same region for all writes, gets reused)
123         struct mxcfb_update_data bupdate;
124         bupdate.update_region.left = 0;
125         bupdate.update_region.top = 0;
126         bupdate.update_region.width = 1872;
127         bupdate.update_region.height = 1404;
128         bupdate.waveform_mode = WAVEFORM_MODE_AUTO;
129         bupdate.update_mode = UPDATE_MODE_FULL;
130         bupdate.update_marker = 0;
131         bupdate.temp = TEMP_USE_AMBIENT;
132         bupdate.flags = 0;
133
134         struct mxcfb_update_marker_data updm;
135         updm.update_marker = 0;
136
137         // mmap to framebuffer
138         int buflength = vinfo.yres_virtual * finfo.line_length;
139         printf("buflength %d\n", buflength);
140         char * region = mmap(0, buflength, PROT_READ | PROT_WRITE,
141                              MAP_SHARED, fb, (off_t)0);
142         if (region == MAP_FAILED) {
143                 fprintf(stderr, "map failed!\n");
144                 return 1;
145         }
146
147         // Write black
148         memset(region, 0x00, buflength);
149         ioctl(fb, MXCFB_SEND_UPDATE, &bupdate);
150         ioctl(fb, MXCFB_WAIT_FOR_UPDATE_COMPLETE, &updm);
151
152         // Write white
153         memset(region, 0xff, buflength);
154         ioctl(fb, MXCFB_SEND_UPDATE, &bupdate);
155         ioctl(fb, MXCFB_WAIT_FOR_UPDATE_COMPLETE, &updm);
156
157         // Write image
158         FILE *pattern = fopen(argv[1], "rb");
159         fseek(pattern, 0, SEEK_END);
160         long psize = ftell(pattern);
161         printf("psize is %d\n", psize);
162         fseek(pattern, 0, SEEK_SET);
163
164         if (psize != buflength) {
165                 fprintf(stderr, "Image must match framebuffer size\n");
166                 return 1;
167         }
168
```

```
169         char *buffer = malloc(psize);
170         fread(buffer, psize, 1, pattern);
171         fclose(pattern);
172 
173         memcpy(region, buffer, psize);
174         ret = ioctl(fb, MXCFB_SEND_UPDATE, &bupdate);
175         ioctl(fb, MXCFB_WAIT_FOR_UPDATE_COMPLETE, &updm);
176         if (0 != ret) {
177                 fprintf(stderr, "MXCFB_SEND_UPDATE failed with error "
178                         "%d, aborting\n", ret);
179                 return 1;
180         }
181 
182         close(fb);
183         return 0;
184 }
```