Allwinner SoC based boards

For boards using an Allwinner ARM based SoC ("sunxi"), the U-Boot build system generates a single integrated image file: `u-boot-sunxi-with-spl.bin.` This file can be used on SD cards, eMMC devices, SPI flash and for the USB-OTG based boot method (FEL). To build this file:

- For 64-bit SoCs, build Trusted Firmware (TF-A, formerly known as ATF) first, you will need its `bl31.bin`. See below for more details.
- Optionally on 64-bit SoCs, build the crust management processor firmware.
- Build U-Boot:

```
$ export BL31=/path/to/bl31.bin              # required for 64-bit SoCs
$ export SCP=/src/crust/build/scp/scp.bin    # optional for some 64-bit SoCs
$ make <yourboardname>_defconfig
$ make -j5 -s
```

- Transfer to an uSD card (see below for more details):

```
$ dd if=u-boot-sunxi-with-spl.bin of=/dev/sdX bs=8k seek=1
```

- Boot and enjoy!

For more details, and alternative boot locations or installations, see below.

Building Arm Trusted Firmware (TF-A)

Boards using a 64-bit Soc (A64, H5, H6, H616, R329) require the BL31 stage of the Arm Trusted Firmware-A firmware. This provides the reference implementation of secure software for Armv8-A, offering PSCI and SMCCC services. Allwinner support is fully mainlined. To build bl31.bin:

```
$ git clone https://git.trustedfirmware.org/TF-A/trusted-firmware-a.git
$ cd trusted-firmware-a
$ make CROSS_COMPILE=aarch64-linux-gnu- PLAT=sun50i_a64 DEBUG=1
$ export BL31=$(pwd)/build/sun50i_a64/debug/bl31.bin
```

The target platform (PLAT=) for A64 and H5 SoCs is sun50i_a64, for the H6 sun50i_h6, for the H616 sun50i_h616, and for the R329 sun50i_r329. Use:

```
$ find plat/allwinner -name platform.mk
```

to find all supported platforms. docs/plat/allwinner.rst contains more information and lists some build options.

Building the Crust management processor firmware

For some SoCs and boards, the integrated OpenRISC management controller can be used to provide power management services, foremost suspend to RAM. There is a community supported Open Source implementation called crust, which runs on most SoCs featuring a management controller.

This firmware part is optional, setting the SCP environment variable to /dev/null avoids the warning message when building without one.

To build crust's scp.bin, you need an OpenRISC (or1k) cross compiler, then:

```
$ git clone https://github.com/crust-firmware/crust.git
$ cd crust
$ make <yourboard>_defconfig
$ make CROSS_COMPILE=or1k-none-elf- scp
$ export SCP=$(pwd)/build/scp/scp.bin
```

Find a list of supported board configurations in the configs/ directory.

Building the U-Boot image

Find the U-Boot defconfig file for your board first. Those files live in the `configs/` directory; you can grep for the stub name of the devicetree file, if you know that, or for the SoC name to find the right version:

```
$ git grep -l MACH_SUN8I_H3 configs
$ git grep -l sun50i-h6-orangepi-3 configs
```

The linux-sunxi wiki also lists the name of the defconfig file in the respective board page. Then use this defconfig file to create the .config file, and build the image:

```
$ make <yourboard>_defconfig
$ make -j5
```

For 64-bit boards, this requires either the BL31 environment variable to be set (as shown above in the TF-A build example), or it to be supplied on the build command line:

```
$ make BL31=/src/tf-a.git/build/sun50i_h616/debug/bl31.bin -j5 -s
```

The same applies to the (optional) SCP firmware.

The file containing everything you need is called `u-boot-sunxi-with-spl.bin`, you will find it in the root folder of your U-Boot (build) tree. Except for raw NAND flash devices this very same file can be used for any boot source. It will contain the SPL image, fitted with the proper signature recognised by the BROM, and the required checksum. Also it will contain at least U-Boot proper, either wrapped in the legacy U-Boot image format, or in a FIT image. The board's devicetree is also included, either appended to the U-Boot proper image, or contained in the FIT image. If required by the SoC, this FIT file will also include the other firmware images.

Installing U-Boot

Installing on a (micro-) SD card

All Allwinner SoCs will try to find a boot image at sector 16 (8KB) of an SD card, connected to the first MMC controller. To transfer the generated image to an SD card, from any Linux device (including the board itself) with an (micro-)SD card reader, type:

```
# dd if=u-boot-sunxi-with-spl.bin of=/dev/sdX bs=1k seek=8
```

`/dev/sdx` needs to be replaced with the block device name of the SD card reader. On some machines this could be `/dev/mmcblkX`. Newer SoCs (starting from H3), also look at sector 256 (128KB) for the signature (after having checked the 8KB location). Installing the firmware there has the advantage of not overlapping with a GPT partition table. Simply replace the "`seek=8`" above with "`seek=128`".

You can also use an existing (mainline) U-Boot to write to the SD card. Load the generated U-Boot image somewhere into DRAM (via `ext4load`, `fatload`, or `tftpboot`), then write to MMC device 0:

```
=> fatload mmc 0:1 $kernel_addr_r u-boot-sunxi-with-spl.bin
=> mmc dev 0
=> mmc write $kernel_addr_r 0x10 0x7f0
```

To use the alternative boot location on newer SoCs:

```
=> mmc write $kernel_addr_r 0x100 0x700
```

Installing on eMMC (on-board flash memory)

Some boards have a soldered eMMC chip, some other boards have an eMMC socket to receive an optional eMMC module. U-Boot can be installed to those chips, to boot without an SD card inserted. The Boot-ROM can boot either from the regular user data partition, or from one of the separate eMMC boot partitions. U-Boot can be installed either from a running Linux instance on the device, from a running (mainline) U-Boot, or via an adapter for the (removable) eMMC module.

Installing on an eMMC user data partition from Linux

If you have a running Linux instance on the device, and have somehow copied over the image file to that device, you can write the image directly into the eMMC device from there. Find the name of the block device file first, it is one of the `/dev/mmcblk<X>` devices. eMMC devices typically also list a `/dev/mmcblk<X>boot0` partition (see below), this helps you to tell it apart from the SD card device. To install onto the user data partition:

```
# dd if=u-boot-sunxi-with-spl.bin of=/dev/dev/mmcblkX bs=1k seek=8
```

Similar to SD cards, the BROM in newer SoCs (H3 and above) also checks sector 256 of an eMMC, so you can use "`seek=128`" as well.

Installing on an eMMC boot partition from Linux

In the following examples, `/dev/mmcblkX` needs to be replaced with the block device name of the eMMC device. The eMMC device can be recognised by also listing the boot partitions (`/dev/mmcblkXboot0`) in `/proc/partitions`.

To allow booting from one of the eMMC boot partitions, this one needs to be enabled first. This only needs to be done once, as this setting is persistent, even though the boot partition can be disabled or changed again any time later:

```
# apt-get install mmc-utils
# mmc bootbus set single_hs x1 x4 /dev/mmcblkX
# mmc bootpart enable 1 1 /dev/mmcblkX
```

The first "1" in the last command points to the boot partition number to be used, typically devices offer two boot partitions.

By default Linux disables write access to the boot partitions, to prevent accidental overwrites. You need to disable the write protection (until the next reboot), then can write the U-Boot image to the *first* sector of the selected boot partition:

```
# echo 0 > /sys/block/mmcblkXboot0/force_ro
# dd if=u-boot-sunxi-with-spl.bin of=/dev/mmcblkXboot0 bs=1k
```

Installing on an eMMC user data partition from U-Boot

You can also write the generated image file to an SD card, boot the device from there, and burn the very same image to the eMMC device from U-Boot. The following commands copy the image from the SD card to the eMMC device:

```
=> mmc dev 0
=> mmc read $kernel_addr_r 0x10 0x7f0
=> mmc dev 1
=> mmc write $kernel_addr_r 0x10 0x7f0
```

You can also copy an image from the 8K offset of an SD card to the 128K offset of the eMMC (or any combination), just change the "`0x10 0x7f0`" above to "`0x100 0x700`", respectively. Of course the image file can be loaded via any other loading method, including `fatload`, `ext4load`, `tftpboot`.

Installing on an eMMC boot partition from U-Boot

The selected eMMC boot partition needs to be initially enabled first (same as in Linux above), you can do this from U-Boot with:

```
=> mmc dev 1
=> mmc bootbus 1 1 0 0
=> mmc partconf 1 1 1 1
```

The first "1" in both commands denotes the MMC device number. The second "1" in the partconf command sets the required BOOT_ACK option, the last two "1"s selects the active boot partition and the target for the next data access, respectively. So for the next "`mmc write`" command to address one of the boot partitions, the last number must either be "1" or "2", "0" would switch (back) to the normal user data partition.

Then load the `u-boot-sunxi-with-spl.bin` image file into DRAM, either by reading directly from an SD card or eMMC user data partition, or from a file system or TFTP (see above), and transfer it to the boot partition:

```
=> tftpboot $kernel_addr_r u-boot-sunxi-with-spl.bin
=> mmc write $kernel_addr_r 0 0x7f0
```

After that the device should boot from the selected boot partition, which takes precedence over booting from the user data partition.

Installing on SPI flash

Some devices have a SPI NOR flash chip soldered on the board. If it is connected to the SPI0 pins on PortC, the BROM can also boot from there. Typically the SPI flash has the lowest boot priority, so SD card and eMMC devices will be considered first.

Installing on SPI flash from Linux

If the devicetree enables and describes the SPI flash device, you can access the SPI flash content from Linux, using the MTD utils:

```
# apt-get install mtd-utils
# mtdinfo
# mtd_debug erase /dev/mtdX 0 0xf0000
# mtd_debug write /dev/mtdX 0 0xf0000 u-boot-sunxi-with-spl.bin
```

`/dev/mtdX` needs to be replaced with the respective device name, as listed in the output of `mtdinfo`.

Installing on SPI flash from U-Boot

If SPI flash driver and command support (`CONFIG_CMD_SF`) is enabled in the U-Boot configuration, the image file can be installed via U-Boot as well:

```
=> tftpboot $kernel_addr_r u-boot-sunxi-with-spl.bin
=> sf probe
=> sf erase 0 +0xf0000
=> sf write $kernel_addr_r 0 $filesize
```

Installing on SPI flash via USB in FEL mode

If the device is in FEL mode (see below), the SPI flash can also be filled with the sunxi-fel utility, via an USB(-OTG) cable from any USB host machine:

```
$ sunxi-fel -v -p spiflash-write 0 u-boot-sunxi-with-spl.bin
```

Booting via the USB(-OTG) FEL mode

If none of the boot locations checked by the BROM contain a medium or valid signature, the BROM will enter the so-called FEL mode, in which it will listen to commands from a host on the SoC's USB-OTG interface. Those commands allow to read from and write to arbitrary memory locations, also to start execution at any address, which allows to bootstrap a board solely via an USB cable. Some boards feature a "FEL" or "U-Boot" button, which forces FEL mode despite a valid boot location being present. The same can be achieved via a magic binary on an SD card, which allows to enter FEL mode on any board.

To use FEL booting, let the board enter FEL mode, via any of the mentioned methods (no boot media, FEL button, SD card with FEL binary), then connect a USB cable to the board's USB OTG port. Some boards (Pine64, TV boxes) don't have a separate OTG port. In this case mostly one of the USB-A ports is connected to USB0, and can be used via a non-standard USB-A to USB-A cable.

Typically there is no on-board indication of FEL mode, other than a new USB device appearing on the connected host computer. The USB vendor/device ID is 1f3a:efe8. Mostly this will identify as "sunxi SoC OTG connector in FEL/flashing mode", but older distributions might still report "Onda (unverified) V972 tablet in flashing mode".

The sunxi_fel tool implements the proprietary BROM protocol, and allows to bootstrap U-Boot by just providing our venerable u-boot-sunxi-with-spl.bin:

```
$ sudo apt-get install sunxi-tools
$ sunxi-fel -v -p uboot u-boot-sunxi-with-spl.bin
```

Additional binaries like a kernel, an initial ramdisk or a boot script, can also be uploaded via FEL, check the Wiki's FEL page for more details.