Packages - Freedom Issue #3095

u-boot has nonfree files in arch/x86/dts/microcode/

2021-09-16 05:38 PM - GNUtoo

Status:	fixed	**% Done:**	0%
Priority:	bug		
Assignee:	GNUtoo		
Category:			

Description

We have at least the following files that are nonfree in arch/x86/dts/microcode/:

```
$ git grep "reverse engineering"
arch/x86/dts/microcode/m0220661105_cv.dtsi: *   .No reverse engineering, decompilation, or disasse
mbly of this software is
arch/x86/dts/microcode/m12206a7_00000029.dtsi: *        .No reverse engineering, decompilation, or
 disassembly of this software is
arch/x86/dts/microcode/m12306a9_0000001b.dtsi: *        .No reverse engineering, decompilation, or
 disassembly of this software is
arch/x86/dts/microcode/m7240651_0000001c.dtsi: *        .No reverse engineering, decompilation, or
 disassembly of this software is
arch/x86/dts/microcode/mc0306d4_00000018.dtsi: *        .No reverse engineering, decompilation, or
 disassembly of this software is
```

For the rest they may be documented in a somewhat recent thread in the linux-libre mailing list, so we could remove them in a second time once we understand if these are code or data.

There is also some documentation that contains instructions to build u-boot with nonfree binaries to remove and there too they could be removed once we find them.

In any case I don't know how to properly remove files in PKGBUILDs as I lost track of if the Parabola modifications to build processed source tarballs worked or not if they were removed or not and so on.

So if someone has an example to follow that works today I could do that modification.

If not I don't know what to do or how to do it.

History

#1 - 2021-10-04 05:06 AM - bill-auger

that is the mksource() mechanism described on the wiki https://wiki.parabola.nu/Hacking:mksource()

to use it:

- add a mksource() function to the PKGBUILD, and move all liberation steps into mksource()
- if the above step results in an empty prepare(), delete prepare()
- rename the 'source' array to 'mksource'
- prefix the checksums array(s) with 'mk' (eg: 'mksha256sums')
- repalce the 'source' array, to require a single (libre) source-ball and it's signature,
  each with a 'repo.parabola.nu/other/' URL, with '-libre' appended to the 'pkgname'
  eg:
  pkgname='ruby'
  source=(https ://repo.parabola.nu/other/ruby-libre/ruby-libre-2.4.1.tar.xz)
- if the libre source-ball is not found on the parabola server, it is generated automatically,
  using the mksource mechanism, before starting the normal build()
- otherwise, the build continues normally (bypassing the mksource() function)
- in either case, `librerelease` will ensure that the source-ball(s) are published

it is a rather clumsy mechanism though, and was implemented in only a few packages (although presumably, every libre replacement package should use it) - there has been much discussion about eliminating mksource(), in favor of simply generating the source ball after the standard prepare() has run, which is effectively what mksource() simulates in a round-about way - "source-ball(s)" is plural in the last step because mksource PKGBUILDs also results in two (probably identical) source-balls in the repo ('foo-libre' in other/ and 'foo.srfc.tar.gz' in sources/)

https://labs.parabola.nu/issues/2403

i have an example PKGBUILD, which does nothing useful, but to exercize the mksource functionality, if you'd like to see it

#2 - 2021-11-01 05:21 AM - GNUtoo

Yes that could help me experiment with it.

Currently makepkg -f gives me that:

```
==> Retrieving sources...
  -> Downloading uboot4extlinux-sunxi-libre-2021.07.tar.xz...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0   153    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
curl: (22) The requested URL returned error: 404
==> ERROR: $PKGEXT does not contain a valid package suffix (needs '.pkg.tar*', got 'uboot4extlinux-sunxi-libre
-2021.07.tar.xz')
==> ERROR: Failure while downloading https://repo.parabola.nu/other/uboot4extlinux-sunxi/uboot4extlinux-sunxi-
libre-2021.07.tar.xz
    Aborting...
```

With the following PKGBUILD (which could be dropped in place of libre/uboot4extlinux-sunxi/PKGBUILD):

```
# U-Boot: sunXi
# Maintainer: Isaac David <isacdaavid@at@isacdaavid@dot@info>
# Contributor: André Silva <emulatorman@hyperbola.info>
# Contributor: Timothy Redaelli <timothy.redaelli@gmail.com>
# Contributor: Denis 'GNUtoo' Carikli <GNUtoo@cyberdimension.org>

# To add a new board (that uses an Allwinner System On a Chip) you need:
# - The package name. Example: uboot4extlinux-a20-olinuxino_micro
# - The u-boot configuration. Example: A20-OLinuXino_MICRO_defconfig
# - The name of the board(s). Example: A20 OLinuXino Micro
#   A single package / u-boot configuration sometimes support multiple
#   boards and the configuration name doesn't always have the boards names
#   in it. Example: uboot4extlinux-am335x_bone supports various BeagleBones,
#   the AM335x GP EVM and the EVM SK all in one u-boot binary and package.
# - If relevant, the packages it replaces, if there are older u-boot packages
#   also supporting that board. Example:uboot-a20-olinuxino-micro
#
# Ideally we'd like to enable people to only add these 4 variables in this
# package to add a new board. These 4 variables could go in something like
# a CVS file that is parsed, but that would require to be able to dynamically
# define packages, but even if eval package_{$_pkgname}() { [...] } works,
# it doesn't make that package function visible to makepkg.
#
# Because of that, contributors wanting to add new boards will have to search
# for example of the information mentioned above and add it in the PKGBUILD
# for their boards.
#
# After adding a new board in this PKGBUILD, to get the board officially
# supported by Parabola.you also need to create a page for it in the
# Parabola wiki for that board and update the ARM installation guide to
# point to it when relevant. Examples of that are available for other
# boards in the ARM installation guide.

pkgbase=uboot4extlinux-sunxi
pkgname=("${pkgbase}"
         'uboot4extlinux-a10-olinuxino-lime'
         'uboot4extlinux-a10s-olinuxino-m'
         'uboot4extlinux-a13-olinuxino'
         'uboot4extlinux-a13-olinuxinom'
         'uboot4extlinux-a20-olinuxino-lime'
         'uboot4extlinux-a20-olinuxino-lime2'
         'uboot4extlinux-a20-olinuxino-lime2-emmc'
         'uboot4extlinux-a20-olinuxino_micro'
         'uboot4extlinux-bananapi'
         'uboot4extlinux-bananapro'
         'uboot4extlinux-chip'
         'uboot4extlinux-cubieboard'
         'uboot4extlinux-cubieboard2'
         'uboot4extlinux-cubietruck'
         'uboot4extlinux-linksprite_pcduino'
         'uboot4extlinux-linksprite_pcduino3'
```

```
         'uboot4extlinux-linksprite_pcduino3_nano'
         'uboot4extlinux-orangepi_2'
         'uboot4extlinux-orangepi_one'
         'uboot4extlinux-orangepi_pc'
         'uboot4extlinux-orangepi_plus')

pkgver=2021.07
pkgrel=3
arch=('armv7h' 'i686' 'x86_64')
url="http://git.denx.de/u-boot.git/"
license=('GPL')
makedepends=('bc' 'dtc' 'python' 'python-setuptools' 'python2' 'swig')
makedepends_i686+=('arm-none-eabi-gcc')
makedepends_ppc64le+=('arm-none-eabi-gcc')
makedepends_x86_64+=('arm-none-eabi-gcc')
# TODO: try longer term archival formats than xz as in practice the xz format is
# specific to the xz implementation or compilation options.
# TODO:
#   -> Downloading uboot4extlinux-sunxi-libre-2021.07.tar.xz...
#   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
#                                  Dload  Upload   Total   Spent    Left  Speed
#   0   153    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
# curl: (22) The requested URL returned error: 404
# ==> ERROR: $PKGEXT does not contain a valid package suffix (needs '.pkg.tar*', got 'uboot4extlinux-sunxi-lib
re-2021.07.tar.xz')
# ==> ERROR: Failure while downloading https://repo.parabola.nu/other/uboot4extlinux-sunxi/uboot4extlinux-sunx
i-libre-2021.07.tar.xz
#     Aborting...
source=("https://repo.parabola.nu/other/${pkgname}/${pkgname}-libre-${pkgver}.tar.xz"
        'extlinux.conf'
        "${pkgbase}.hook.in"
        'install-uboot4extlinux.sh.in'
        "generate-${pkgbase}-install-text.sh")
mksource=("ftp://ftp.denx.de/pub/u-boot/u-boot-${pkgver}.tar.bz2")
sha256sums=('312b7eeae44581d1362c3a3f02c28d806647756c82ba8c72241c7cdbe68ba77e'
            'SKIP'
            'SKIP'
            'SKIP'
            'SKIP')
mksha256sums=('312b7eeae44581d1362c3a3f02c28d806647756c82ba8c72241c7cdbe68ba77e')

_defconfigs=('A10-OLinuXino-Lime_defconfig'
             'A10s-OLinuXino-M_defconfig'
             'A13-OLinuXino_defconfig'
             'A13-OLinuXinoM_defconfig'
             'A20-OLinuXino-Lime_defconfig'
             'A20-OLinuXino-Lime2_defconfig'
             'A20-OLinuXino-Lime2-eMMC_defconfig'
             'A20-OLinuXino_MICRO_defconfig'
             'Bananapi_defconfig'
             'Bananapro_defconfig'
             'CHIP_defconfig'
             'Cubieboard_defconfig'
             'Cubieboard2_defconfig'
             'Cubietruck_defconfig'
             'Linksprite_pcDuino_defconfig'
             'Linksprite_pcDuino3_defconfig'
             'Linksprite_pcDuino3_Nano_defconfig'
             'orangepi_2_defconfig'
             'orangepi_one_defconfig'
             'orangepi_pc_defconfig'
             'orangepi_plus_defconfig')

# Offset at which to install u-boot
u_boot_with_spl_offset=8192

case "$CARCH" in
    armv7h)              ARCH=arm;;
    i686|ppc64le|x86_64) ARCH=arm;CROSS_COMPILE=arm-none-eabi-;;
esac

_nr_targets=0
for _defconfig in ${_defconfigs[@]}; do
    _nr_targets=$(expr ${_nr_targets} + 1)
done
```

```
_get_target_name()
{
    _defconfig="$1"

    echo ${_defconfig} | \
        sed 's/_defconfig$//' | \
        awk '{print tolower($0)}'
}

_get_target_destdir()
{
    _defconfig="$1"

    _target_name="$(_get_target_name ${_defconfig})"

    echo "build/uboot4extlinux-${_target_name}"
}

_build_uboot_target()
{
    _defconfig="$1"

    _destdir="$(_get_target_destdir ${_defconfig})"
    _target_name="$(_get_target_name ${_defconfig})"

    unset CFLAGS CXXFLAGS LDFLAGS

    make "ARCH=${ARCH}" distclean
    make "ARCH=${ARCH}" ${_defconfig}

    echo 'CONFIG_IDENT_STRING=" Parabola GNU/Linux-libre"' >> .config

    if [ "$CARCH" = "armv7h" ]; then
        make "ARCH=${ARCH}" EXTRAVERSION=-${pkgrel}
    else
        make "ARCH=${ARCH}" "CROSS_COMPILE=${CROSS_COMPILE}" \
             EXTRAVERSION=-${pkgrel}
    fi

    echo "==> Installing ${_target_name} to ${_destdir}"
    install -d ${_destdir}
    mv -f u-boot-sunxi-with-spl.bin "${_destdir}"

}

mksource()
{
  cd u-boot-${pkgver}

  # The list of files was generated with the following command:
  # git grep "open source" | sed 's/:.*//g'
  sed 's/open source/free software/g' -i \
    Kconfig \
    arch/x86/Kconfig \
    board/egnite/ethernut5/ethernut5.c \
    board/synopsys/emsdp/README \
    board/synopsys/hsdk/README \
    board/synopsys/iot_devkit/README \
    cmd/clone.c \
    common/spl/Kconfig \
    doc/README.rockchip \
    doc/README.rockusb \
    doc/arch/x86.rst \
    doc/board/amlogic/libretech-cc.rst \
    doc/board/sipeed/maix.rst \
    doc/develop/uefi/iscsi.rst \
    doc/device-tree-bindings/pinctrl/pinctrl-bindings.txt \
    doc/imx/misc/sdp.txt \
    drivers/gpio/gpio-uclass.c \
    drivers/pinctrl/pinctrl-stmfx.c \
    include/asm-generic/gpio.h \
    include/dm/pinctrl.h \
    include/dm/pinctrl.h \
    include/dt-bindings/gpio/gpio.h \
```

```
    include/dt-bindings/gpio/gpio.h \
    tools/buildman/README \
    tools/sunxi-spl-image-builder.c \

# The licenses of some microcodes are nonfree because the header contains the
# following: ".No reverse engineering, decompilation, or disassembly of this
# software is permitted."
rm -rf arch/x86/dts/microcode/

# The license is nonfree because it contains the following: "Reverse
# engineering, decompilation, or disassembly of this software is not
# permitted."
rm -f Licenses/r8a779x_usb3.txt
rm -f drivers/usb/host/xhci-rcar-r8a779x_usb3_v3.h

# The documentation contains instructions to download and install nonfree
# software. Note that if a board doesn't have such instructions it doesn't
# necessarily means that it can boot with only free software and viceversa.

############
# Amlogic #
############
# Amlogic SOCs Usually have various nonfree components, like the first stages
# of the bootloaders and code that runs in TrustZone. They are most likely
# not signed.
# ---------
# TODO: List the nonfree software of specific documentation
rm -f doc/board/amlogic/beelink-gtkingpro.rst
rm -f doc/board/amlogic/beelink-gtking.rst
rm -f doc/board/amlogic/index.rst
rm -f doc/board/amlogic/khadas-vim2.rst
rm -f doc/board/amlogic/khadas-vim3l.rst
rm -f doc/board/amlogic/khadas-vim3.rst
rm -f doc/board/amlogic/khadas-vim.rst
rm -f doc/board/amlogic/libretech-ac.rst
rm -f doc/board/amlogic/libretech-cc.rst
rm -f doc/board/amlogic/nanopi-k2.rst
rm -f doc/board/amlogic/odroid-c2.rst
rm -f doc/board/amlogic/odroid-c4.rst
rm -f doc/board/amlogic/odroid-n2.rst
rm -f doc/board/amlogic/p200.rst
rm -f doc/board/amlogic/p201.rst
rm -f doc/board/amlogic/p212.rst
rm -f doc/board/amlogic/q200.rst
rm -f doc/board/amlogic/s400.rst
rm -f doc/board/amlogic/sei510.rst
rm -f doc/board/amlogic/sei610.rst
rm -f doc/board/amlogic/u200.rst
rm -f doc/board/amlogic/w400.rst
rm -f doc/board/amlogic/wetek-core2.rst

#####################################
# FPGA toolchains / synthesize tools #
#####################################
# Steer users toward using nonfree Quartus software
rm -f doc/README.socfpga
# Steer users toward using files that can only be produced with the nonfree
# "libero" FPGA toolchain / synthesizer tool.
rm -f doc/board/microchip/mpfs_icicle.rst

#########
# Linux #
#########
# Has intructions to build Linux which is not FSDG compliant.
# TODO: Use linux-libre instead, especially because documentation about vboot
# could be interesting to have. Vboot is a chain of trust that can work with
# only free software. The hardware root of trust can be created by booting on
# a flash chip whose security registers are configured to set the first
# bootloader component read-only.
rm -f doc/uImage.FIT/beaglebone_vboot.txt
# Steers very strongly users into using Linux as it shows that the only tested
# kernels are Broadcom forks of Linux. We would need to have linux-libre
# versions of these or test it with stock linux-libre instead.
rm -f doc/README.bcm7xxx
```

```
############
# Mediatek #
############
# The instructions uses binaries that lack any corresponding source code.
rm -f doc/README.mediatek

##############
# NXP I.MX8 #
##############
# I.MX8 SOCs require a nonfree firmware for the DDR4 controller. In some
# documentation, I didn't find that requirement mentioned, but instead
# there are still nonfree files mentioned. So I assume that they might
# somehow contain code for that nonfree DDR4 controller, but it might be
# worth checking if it's the case or not. The DDR4 controller firmware is not
# signed. In addition the I.MX8 HDMI controller requires a signed firmware.
# -----------
# nonfree DDR4 controller firmware
rm -f doc/board/freescale/imx8mp_evk.rst
# nonfree DDR4 controller and HDMI firmwares
rm -f doc/board/freescale/imx8mq_evk.rst
# nonfree DDR4 controller firmware
rm -f doc/board/freescale/imx8mn_evk.rst
# nonfree imx-sc-firmware-1.2.7.1.bin and imx-seco-2.3.1.bin firmwares
rm -f doc/board/freescale/imx8qxp_mek.rst
# nonfree DDR4 controller firmware
rm -f doc/board/freescale/imx8mm_evk.rst
# nonfree imx-sc-firmware-1.1.bin and firmware-imx-8.0.bin firmwares
rm -f doc/board/advantech/imx8qm-rom7720-a1.rst
# TODO: List the nonfree software of specific documentation
rm -f doc/board/verdin-imx8mm.rst
rm -f doc/board/toradex/colibri-imx8x.rst
rm -f doc/board/toradex/apalix-imx8x.rst
rm -f doc/board/toradex/apalix-imx8.rst

########################
# NXP nonfree srktool #
########################
# The SRK tool is a tool that is involved in one way or another with
# authenticated or encrypted boot. I'm unsure if free software replacements
# exists or if could easily be replaced with a free software implementation.
# In any case the I.MX6 and I.MX5 can proabably be setup for encrypted or
# authenticated boot with free software tools. The first and second versions
# of the USB Armory has documentation on how to do that.
# ----------------------
rm -f doc/imx/board/toradex/colibri_imx7.rst
rm -f doc/imx/habv4/introduction_habv4.txt

##################
# Samsung Exynos #
##################
# The instructions makes users nonfree components like a nonfree first stage
# bootloaders, and nonfree code that runs in TrustZone.
rm -f doc/README.odroid
# The instructions makes its users download an image and update u-boot in that
# image. Because of that, it's extremely likely that the images contains
# nonfree components that cannot even be redistributed in another form, and
# that the instructions uses that images because of that.
rm -f doc/README.s5p4418

######################
# Texas Instruments #
######################
# Users are expected to use nonfree tools and even sign an NDA to get access
# to them.
rm -f doc/README.ti-secure

##########
# U-boot #
##########
# The tutorial has instructions to download a downstream u-boot, so it might
# have the same issues than u-boot itself if the u-boot is recent enough.
rm -f doc/README.armada-secureboot
# Points to some installations instructions that use u-boot from git (which
# has nonfree software in it). These instructions also have a set of
# unreviewed git repositories.
```

```
rm -f doc/README.marvell
# The tutorial has instructions to download a downstream u-boot, so it might
# have the same issues than u-boot itself if the u-boot is recent enough.
rm -f doc/chromium/run_vboot.rst

############
# Unknown #
############
# The referenced git repository is not present on the web anymore (due to
# Freescale acquisition by NXP). TODO: we need to look if there were
# archives of that software and review it.
rm -f doc/README.mpc85xx-sd-spi-boot

#######
# x86 #
#######
# Unless the computer is supported by Libreboot, or that u-boot runs after
# some other nonfree boot software like a BIOS or UEFI, it's unlikely to be
# able to run with only free software. Though I'm pretty sure that some
# exceptions do exists, but they are probably not supported by u-boot.
# -----
# nonfree Management Engine firmware, RAM intialization code, and video BIOS
rm -f doc/board/google/chromebook_link.rst
# nonfree SDRAM and hardware intialization code
rm -f doc/board/google/chromebook_coral.rst

# nonfree FSP, video BIOS, Management Engine firmware
rm -f doc/board/intel/minnowmax.rst
# nonfree FSP, Chipset Micro Code (CMC), microcode
rm -f doc/board/intel/crownbay.rst

# Steers userstoward using nonfree FSP
rm -f board/intel/slimbootloader.rst

# Steers users and developers toward using nonfree FSP
rm -f doc/device-tree-bindings/fsp/fsp2/apollolake/fsp-m.txt

# Steers users and developers toward using nonfree FSP
rm -f doc/device-tree-bindings/fsp/fsp2/apollolake/fsp-s.txt

#############
# Rockchip #
#############
# rkbin binaries without license nor source code
rm -f doc/board/rockchip/rockchip.rst
rm -f doc/README.rockchip

###################################
# References external source code #
###################################
# - doc/imx/common/mxs.txt: elftosb-10.12.01.tar.gz has a BSD 3 clauses
#   license and the only binaries present seems to be used for testing
#   purposes (it's elftosb generates some headers so it also has tests to
#   check if the generated headers are OK.
# - doc/README.fdt-control: References the device tree compiler which
#   is free software.
# - The official ARM trusted firmware[1], which is often referenced by
#   various documentation files is under the BSD 3 clauses license.
#   It also contains SPDX license identifiers and all of them are free
#   software at commit e33ca7b44 (Merge changes from topic "ck/mpmm" into
#   integration).
#   [1]https://git.trustedfirmware.org/TF-A/trusted-firmware-a.git
# - doc/README.rockusb: References rkdeveloptool which is free software.
#   it has SPDX GPL-2.0+ identifiers and the GPLv2 license so it's GPLv2
#   or later at the 46bb4c0 commit (v1.32: 1.support to upgrade 356x loader).
# - doc/README.OFT: References to the device tree compiler which is free
#   software, and to xxd, which is part of vim.
# - board/intel/edison.rst: xFSTK is LGPL 2.1 or later.
#   TODO: The instructions reference a Google drive that I didn't check,
#   and I don't know how to avoid nonfree JavaScript with Google Drive,
#   so I'm removing the reference until someone checks all that. That
#   comment could also be updated if only one of the requirements is
#   checked (like accessing Google Drive with free software).
sed 's/You might find this `drive`_ helpful.//' -i doc/board/intel/edison.rst
sed 's/.. _drive: .*//' -i doc/board/intel/edison.rst
```

```
}

build()
{
  cd u-boot-${pkgver}

  _target_nr=0
  for _defconfig in ${_defconfigs[@]}; do
      _target_nr=$(expr ${_target_nr} + 1)
      _target_name="$(_get_target_name ${_defconfig})"

      echo "==> Building ${_target_name} " \
           "[${_target_nr} of ${_nr_targets} targets]"

      _build_uboot_target "${_defconfig}"
  done
}

_check_uboot_target()
{
    _defconfig="$1"

    _image="$(_get_target_destdir ${_defconfig})/u-boot-sunxi-with-spl.bin"
    _image_size="$(du --bytes --apparent-size ${_image} | awk '{print $1}')"
    _offset="${u_boot_with_spl_offset}"

    _image_end=$(expr ${_image_size} + ${_offset})

    # This test comes from install-uboot4extlinux.sh.in
    if [ ${_image_end} -gt $(expr 1024 \* 1024) ] ; then
        echo "Error: ${_image} is too big:"
        echo "       offset: ${_offset}"
        echo "       size:   ${_image_size}"
        echo "       By default, "\
             "partitioing tools makes the first partition start at 1MiB"
        echo "       Instaling ${_image} "\
             "would overwrite that first partition (if it's present)."

        false # Fail the test
    fi
}

check()
{
  cd u-boot-${pkgver}

  _target_nr=0
  for _defconfig in ${_defconfigs[@]}; do
      _target_nr=$(expr ${_target_nr} + 1)
      _target_name="$(_get_target_name ${_defconfig})"

      echo "==> Testing target ${_target_name} " \
           "[${_target_nr} of ${_nr_targets} targets]"
      _check_uboot_target "${_defconfig}"
  done

}

_make_uboot_package()
{
    _pkgname="$1"

    if [ -n "${pkgdir}" ] ; then
        cd "u-boot-${pkgver}"

        # Install the u-boot binary
        install -d "${pkgdir}/usr/lib/u-boot/${_pkgname}/"
        install -Dm644 "build/${_pkgname}/u-boot-sunxi-with-spl.bin" \
                "${pkgdir}/usr/lib/u-boot/${_pkgname}"

        # Install the u-boot installation script
        sed < "${srcdir}/install-uboot4extlinux.sh.in" > \
            "${pkgdir}/usr/lib/u-boot/${_pkgname}/install-uboot4extlinux.sh" \
              -e "s|@u_boot_with_spl[@]|/usr/lib/u-boot/${_pkgname}/u-boot-sunxi-with-spl.bin|g" \
              -e "s|@u_boot_with_spl_offset[@]|${u_boot_with_spl_offset}|g"
```

```
        chmod +x \
              "${pkgdir}/usr/lib/u-boot/${_pkgname}/install-uboot4extlinux.sh"

        # Install what is required for the pacman hook
        install -d "${pkgdir}/usr/share/libalpm/hooks/"
        sed < "${srcdir}/${pkgbase}.hook.in" > \
            "${pkgdir}/usr/share/libalpm/hooks/${_pkgname}.hook" \
            -e "s|@pkgname[@]|${_pkgname}|g"

        install -d "${pkgdir}/usr/share/doc/u-boot/${_pkgname}/"
        # If we install several uboot4extlinux, we need a way to clearly
        # separate each postinstall message. To do that we wrapped the
        # text in an ASCII art square, but doing that is complicated when
        # using sed as the package name as well as the installation script
        # path both have variable length.
        sh "${srcdir}/generate-${pkgbase}-install-text.sh" \
           "${_pkgname}" \
           "${pkgbase}" \
           "/usr/lib/u-boot/${_pkgname}/install-uboot4extlinux.sh" > \
           "${pkgdir}/usr/share/doc/u-boot/${_pkgname}/install-uboot4extlinux.txt"
    fi
}

_make_pkgdesc()
{
    board_name="$1"
    echo "U-Boot with Extlinux support for ${board_name}"
}

package_uboot4extlinux-sunxi()
{
    pkgdesc="Scripts for managing U-Boot installations for computers with Allwinner System On a Chip"

    depends=('uboot-tools')

    # Users are expected to use this as a base for /boot/extlinux/extlinux.conf
    install -d "${pkgdir}/usr/lib/u-boot/${pkgname}/"
    install -Dm644 \
            "${srcdir}/extlinux.conf" \
            "${pkgdir}/usr/lib/u-boot/${pkgname}/"
}

package_uboot4extlinux-a10-olinuxino-lime()
{
    pkgdesc=$(_make_pkgdesc "A10 OLinuXino Lime")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-a10s-olinuxino-m()
{
    pkgdesc=$(_make_pkgdesc "A10s OLinuXino Micro")

    replaces=('uboot4extlinux-a10s-olinuxino-micro')

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-a13-olinuxino()
{
    pkgdesc=$(_make_pkgdesc "A13 OLinuXino")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}
```

```
package_uboot4extlinux-a13-olinuxinom()
{
    pkgdesc=$(_make_pkgdesc "uboot4extlinux-a13-olinuxino-micro")

    replaces=('uboot4extlinux-a13-olinuxino-micro')

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-a20-olinuxino-lime()
{
    pkgdesc=$(_make_pkgdesc "A20 OLinuXino Lime")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-a20-olinuxino-lime2()
{
    pkgdesc=$(_make_pkgdesc "A20 OLinuXino Lime2")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-a20-olinuxino-lime2-emmc()
{
    pkgdesc=$(_make_pkgdesc "A20 OLinuXino Lime2 with eMMC")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-a20-olinuxino_micro()
{
    pkgdesc=$(_make_pkgdesc "uboot-a20-olinuxino-micro")

    replaces=('uboot-a20-olinuxino-micro')

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-bananapi()
{
    pkgdesc=$(_make_pkgdesc "Banana Pi")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-bananapro()
{
    pkgdesc=$(_make_pkgdesc "Banana Pro")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-chip()
```

```
{
    pkgdesc=$(_make_pkgdesc "C.H.I.P")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-cubieboard()
{
    pkgdesc=$(_make_pkgdesc "Cubieboard")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-cubieboard2()
{
    pkgdesc=$(_make_pkgdesc "Cubieboard 2")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-cubietruck()
{
    pkgdesc=$(_make_pkgdesc "Cubietruck")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-linksprite_pcduino()
{
    pkgdesc=$(_make_pkgdesc "uboot4extlinux-pcduino")

    replaces=('uboot4extlinux-pcduino')

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-linksprite_pcduino3()
{
    pkgdesc=$(_make_pkgdesc "uboot4extlinux-pcduino3")

    replaces=('uboot4extlinux-pcduino3')

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-linksprite_pcduino3_nano()
{
    pkgdesc=$(_make_pkgdesc "uboot4extlinux-pcduino3-nano")

    replaces=('uboot4extlinux-pcduino3-nano')

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}
```

```
package_uboot4extlinux-orangepi_2()
{
    pkgdesc=$(_make_pkgdesc "Orange Pi 2")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-orangepi_one()
{
    pkgdesc=$(_make_pkgdesc "Orange Pi One")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-orangepi_pc()
{
    pkgdesc=$(_make_pkgdesc "Orange Pi PC")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}

package_uboot4extlinux-orangepi_plus()
{
    pkgdesc=$(_make_pkgdesc "Orange Pi Plus")

    # util-linux is needed for blkid for install-uboot4extlinux.sh
    depends=("${pkgbase}" 'util-linux')

    _make_uboot_package "${pkgname}"
}
```

#3 - 2022-01-19 01:53 PM - GNUtoo

- Tracker changed from Bug to Freedom Issue

As I didn't manage to use mksource, I've started to work on upstreaming the script to cleanup u-boot in Libreboot. Once this is done it would be easier to work with as we could reuse their released u-boot tarballs.

#4 - 2022-03-31 12:18 AM - GNUtoo

- Assignee set to GNUtoo

- Status changed from unconfirmed to confirmed

#5 - 2022-03-31 12:19 AM - GNUtoo

That's now fixed. I did the fix by deblobbing u-boot in Libreboot and generating a tarball like it is done with linux-libre.

#6 - 2022-03-31 12:19 AM - GNUtoo

- Status changed from confirmed to fixed