Suzie Linux https://suzielinux.com/

Suzie Linux was named in memory of my adorable Maine Coon cat Suzie.

Suzie Linux
Pocketbeagle2 board
documentation

Author	Date	Project	Revisions
Michel Catudal	2025-04-22	**Pocketbeagle 2 Linux creation**	**2**

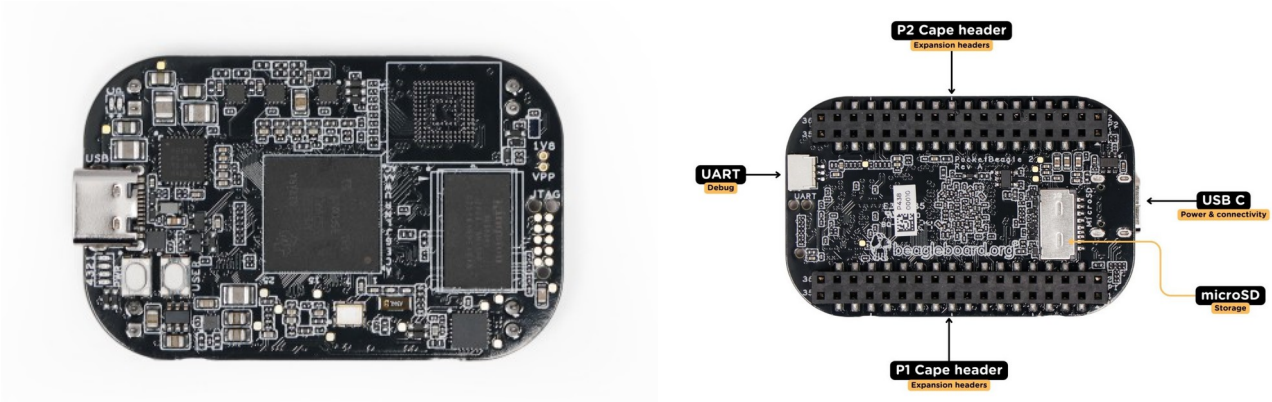
REVISION TRACKING SHEET

Rev	Name	Date	Comment
1	Michel Catudal	2025-04-20	
2	Michel Catudal	2025-04-22	Fixed some typos and replace 2 images

Table of Content

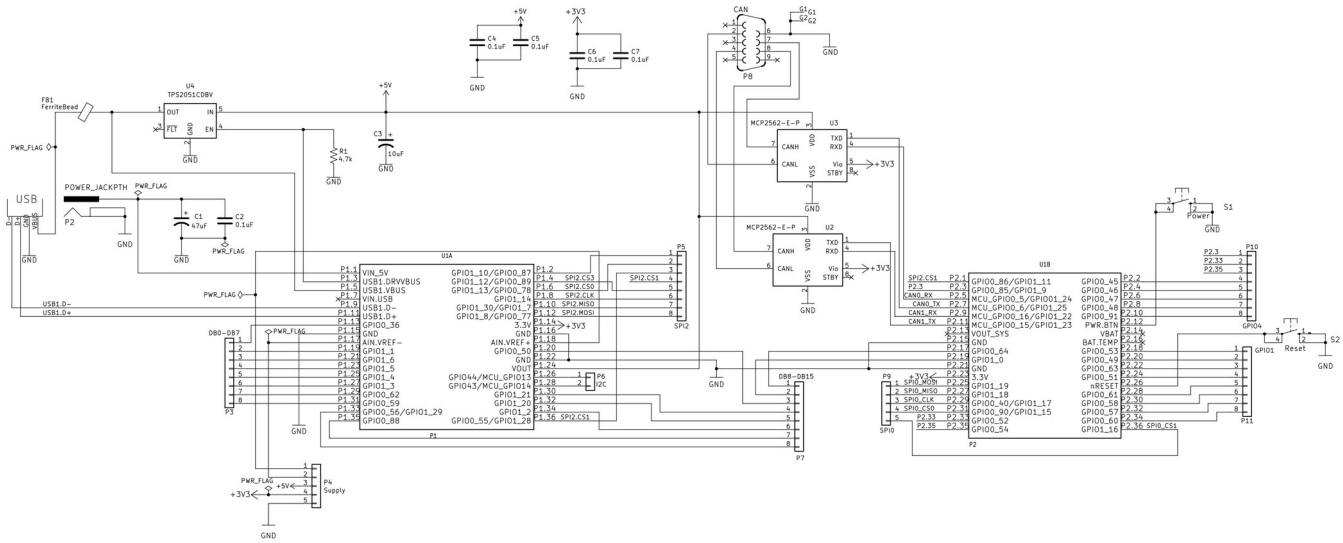
1. Hardware

1.1. Overview of pocketbeagle2 board



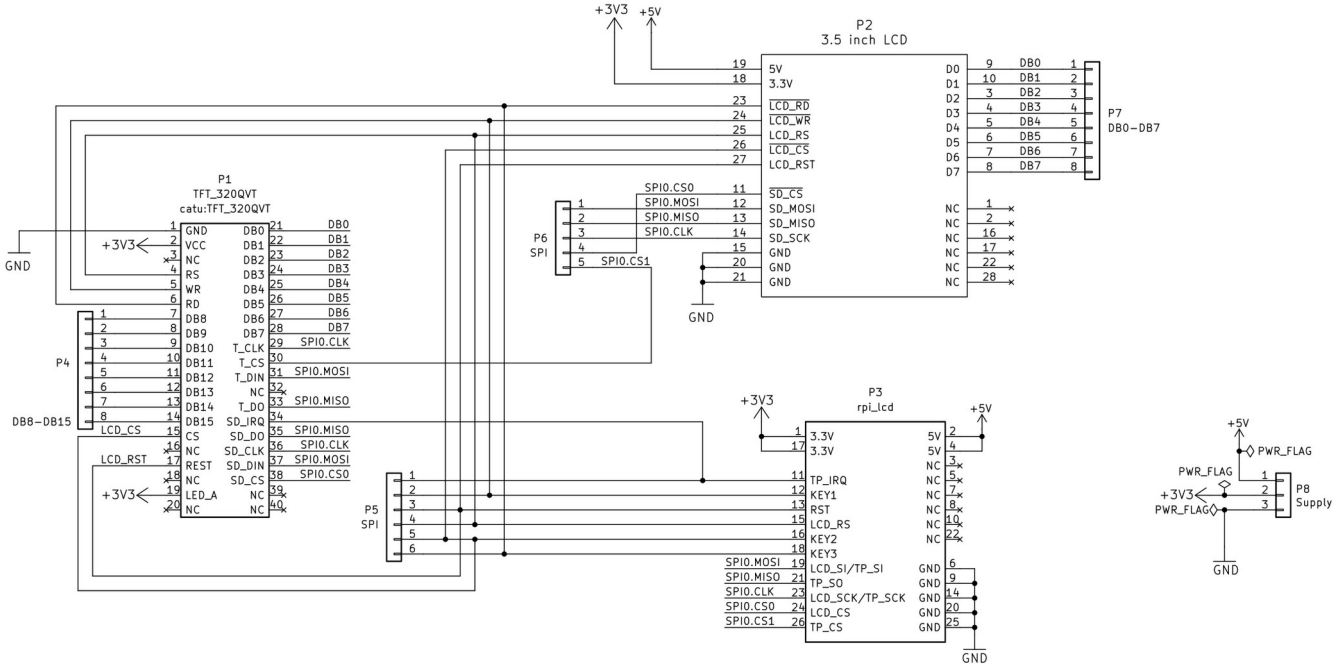
1.2. Pocketbeagle2 cape

Added support for USB so an ethernet or wifi adapter can be used
Added drivers for the two can ports
Added some connectors so an LCD can be used and other components



1.3. LCD adapter board

This board adapts to the TFT_320QVT or RPI_LCD board
It connects to the pocketbeagle2 cape



2. Gentoo applications required

```
emerge --ask dev-python/cryptography
emerge --ask dev-python/pyelftools
emerge --ask dev-util/yamllint
emerge --ask dev-python/jsonschema
emerge --ask gnutls
emerge --ask flex
emerge --ask sys-devel/bc
emerge --ask bison
emerge --ask swig
emerge --ask dosfstools
emerge --ask genimage
emerge --ask mtool
emerge --ask arch-chroot
```

```
In order to chroot on a arm64 rootfs a few things have to be done.
First you need to make sure that the kernel supports it and emerge needed support
The build system's kernel must support miscellaneous binary formats.
This can be enabled  with CONFIG_BINFMT_MISC=m
or CONFIG_BINFMT_MISC=y in the the kernel's .config file.

A system restart is required after building this module before it can be used.

Enable CONFIG_BINFMT_MISC
Executable file formats  --->
  <*> Kernel support for MISC binaries

USE=static-user needs to be set

QEMU_SOFTMMU_TARGETS and QEMU_USER_TARGETS are empty by default and must be defined to
utilize user targets.

echo 'app-emulation/qemu static-user QEMU_SOFTMMU_TARGETS: * QEMU_USER_TARGETS: *' >
/etc/portage/package.use/qemu
echo 'dev-libs/glib static-libs' >> /etc/portage/package.use/qemu
echo 'sys-libs/zlib static-libs' >> /etc/portage/package.use/qemu
echo 'sys-apps/attr static-libs' >> /etc/portage/package.use/qemu
echo 'dev-libs/libpcre2 static-libs' >> /etc/portage/package.use/qemu

emerge --ask app-emulation/qemu

All work is done as a user, we go to a directory where we will install the files

cd ~
mkdir PocketBeagle2
cd PocketBeagle2
export work_directory=$(pwd)
```

3. Cross Compiler: 32bit arm-linux-gnueabi-gcc

```
cd $work_directory
```

3.1. Download/Extract

```
wget -c https://mirrors.edge.kernel.org/pub/tools/crosstool/files/bin/x86_64/11.5.0/x86_64-gcc-11.5.0-nolibc-arm-linux-gnueabi.tar.xz
tar-xf x86_64-gcc-11.5.0-nolibc-arm-linux-gnueabi.tar.xz
export CC32=`pwd`/gcc-11.5.0-nolibc/arm-linux-gnueabi/bin/arm-linux-gnueabi-
```

4. Cross Compiler: 64bit aarch64-linux-gcc

4.1. Download/Extract

```
wget -c https://mirrors.edge.kernel.org/pub/tools/crosstool/files/bin/x86_64/11.5.0/x86_64-gcc-11.5.0-nolibc-aarch64-linux.tar.xz

tar -xf x86_64-gcc-11.5.0-nolibc-aarch64-linux.tar.xz

export CC64=`pwd`/gcc-11.5.0-nolibc/aarch64-linux/bin/aarch64-linux-
```

5. Bootloader

5.1. TI Linux Firmware

5.1.1. Download

```
git clone -b 11.00.08 https://github.com/beagleboard/ti-linux-firmware.git --depth=1
```

5.1.2. Build

```
make -C ./trusted-firmware-a/ -j16 CROSS_COMPILE=${CC64} PLAT=k3 ARCH=aarch64 SPD=opteed TARGET_BOARD=lite K3_USART=0x6 all
```

5.2. Trusted Firmware A

5.2.1. Download

```
git clone -b lts-v2.12 https://github.com/TrustedFirmware-A/trusted-firmware-a.git --depth=1
```

5.2.2. Build

```
make -C ./trusted-firmware-a/ -j16 CROSS_COMPILE=${CC64} PLAT=k3 ARCH=aarch64 SPD=opteed TARGET_BOARD=lite K3_USART=0x6 all
```

5.3. OPTEE

5.3.1. Download

```
git clone -b 4.5.0 https://github.com/OP-TEE/optee_os.git --depth=1
```

5.3.2. Build

```
make -C ./optee_os/ -j4 CROSS_COMPILE=${CC32} CROSS_COMPILE64=${CC64} CFG_ARM64_core=y
PLATFORM=k3-am62x CFG_WITH_SOFTWARE_PRNG=y CFG_CONSOLE_UART=0x6 all
```

5.4. u-boot

5.4.1. Download

```
git clone -b v2025.04-rc5-pocketbeagle2 https://github.com/beagleboard/u-boot.git --
depth=1
```

5.4.2. Build Cortex-R4

```
make -C ./u-boot O=../CORTEXR CROSS_COMPILE=${CC32} am6232_pocketbeagle2_r5_defconfig

make -C ./u-boot -j4 O=../CORTEXR CROSS_COMPILE=${CC32} BINMAN_INDIRS=../ti-linux-
firmware
```

5.4.3. Build Cortex-A53

```
make -C ./u-boot/ O=../CORTEXA CROSS_COMPILE=${CC64}
am6232_pocketbeagle2_a53_defconfig

make -C ./u-boot/ -j16 O=../CORTEXA CROSS_COMPILE=${CC64}
BL31=../trusted-firmware-a/build/k3/lite/release/bl31.bin TEE=../optee_os/out/arm-
plat-k3/core/tee-pager_v2.bin BINMAN_INDIRS=../ti-linux-firmware
```

5.4.4. Copy Build Objects

```
cp -v ./CORTEXA/tispl.bin input
cp -v ./CORTEXA/u-boot.img input

cp trusted-firmware-a/build/k3/lite/release/bl31.bin input
```

6. Linux Kernel

```
cd $work_directory
```

This script will build the kernel, modules, device tree binaries and copy them to the deploy directory.

6.1. Download

```
git clone https://github.com/RobertCNelson/arm64-multiplatform kernelbuildscripts
cd kernelbuildscripts
git checkout origin/v6.14.x-arm64-k3
```

6.2. Build

```
./build_kernel.sh
```

7. Save u-boot and kernel files for later use

7.1. Create definition and directory

```
cd ~/PocketBeagle2
export work_directory=$(pwd)
export kernel_version=6.14.2-suzie-arm64-k3-r12
export deploy_dir=kernelbuildscripts/deploy
export kernel_directory=kernelbuildscripts/KERNEL
export kernel_boot_directory=$kernel_directory/arch/arm64/boot
export input_dir=$work_directory/input
mkdir -p $input_dir
```

7.2. Copy u-boot files

```
cd $input_dir
cp CORTEXR/tiboot3-am62x-hs-fs-evm.bin ./
cp CORTEXA/tispl.bin ./
cp CORTEXA/u-boot.img ./
```

7.3. Create extlinux.conf file

```
echo 'label Linux' > extlinux.conf
echo 'kernel /Image.gz' >> extlinux.conf
echo 'fdtdir /' >> extlinux.conf
echo 'append console=ttyS2,115200n8 earlycon=ns16550a,mmio32,0x02860000
root=/dev/mmcblk1p2 ro rootfstype=ext4 rootwait net.ifnames=0' >> extlinux.conf
```

7.4. Copy kernel files

```
cp $kernel_boot_directory/dts/ti/k3-am6232-pocketbeagle2.dtb ./
cp $deploy_dir/$kernel_version.Image Image
gzip Image
cp $deploy_dir/config-$kernel_version ./

The next part is done as root
su
tar xfv $deploy_dir/$kernel_version-modules.tar.gz --strip-components 2
cd modules/$kernel_version
rm build
ln -s /usr/src/linux build
cd $input_dir
tar cvfJ modules-$kernel_version.tar.xz modules
rm -rf modules
cp -Rp $kernel_directory ./linux-$kernel_version
sync
cd linux-$kernel_version
make mrproper
rm -rf .git
tar cvfJ linux-$kernel_version-source.tar.xz linux-$kernel_version
rm -rf linux-$kernel_version
chown $USER:$USER linux-$kernel_version-source.tar.xz
cd $work_directory
tar cvfJ pocketbeagle2-misc-boot-files.tar.xz input genimage.cfg
chown $USER:$USER pocketbeagle2-misc-boot-files.tar.xz
exit
```

7.5. Content of genimage.cfg used to generate the image

```
image boot.vfat {
        vfat {
            files = {
                    "tispl.bin",
                    "u-boot.img",
                    "Image.gz",
            }
            file tiboot3.bin {
                    image = tiboot3-am62x-hs-fs-evm.bin
            }
            file ti/k3-am6232-pocketbeagle2.dtb {
                    image = k3-am6232-pocketbeagle2.dtb
            }
            file extlinux/extlinux.conf {
                    image = extlinux.conf
            }
        }
        size = 256M
}
image sdcard.img {
        hdimage {
        }
        partition u-boot {
                partition-type = 0xC
                bootable = "true"
                image = "boot.vfat"
        }
        partition rootfs {
                partition-type = 0x83
                image = "rootfs.ext4"
        }
}
```

8. Arch Linux Root File System

```
export rootfs_dir=$work_directory/arch_rootfs

cd $work_directory
```

8.1. Download

```
wget http://os.archlinuxarm.org/os/ArchLinuxARM-aarch64-latest.tar.gz
```

8.2. Create a root file System

```
mkdir -p $rootfs_dir
sudo tar xfvp ArchLinuxARM-aarch64-latest.tar.gz -C $rootfs_dir
sudo sync
sudo rm -rf $rootfs_dir/boot/*
sudo cp /usr/bin/qemu-aarch64 $rootfs_dir/usr/bin
sudo cp /etc/locale.gen $rootfs_dir/etc

Create some alias :

cd $rootfs_dir/etc
echo 'alias ll='ls -alF'' > $rootfs_dir/root/.bashrc
echo 'alias la='ls -A''  >> $rootfs_dir/root/.bashrc
echo 'alias l='ls -CF''  >> $rootfs_dir/root/.bashrc
echo ''  >> $rootfs_dir/root/.bashrc
echo 'alias dir='ls -la -N --color''  >> $rootfs_dir/root/.bashrc
echo ''  >> $rootfs_dir/root/.bashrc
echo 'alias rm='rm -i'' >> $rootfs_dir/root/.bashrc
echo 'alias del='rm -i''  >> $rootfs_dir/root/.bashrc
echo ''  >> $rootfs_dir/root/.bashrc
echo 'alias rd=rmdir' >> $rootfs_dir/root/.bashrc
echo 'alias md='mkdir -p''  >> $rootfs_dir/root/.bashrc
cd $work_directory
```

8.3. chroot into archlinux rootfs

```
sudo arch-chroot $rootfs_dir
source /etc/profile
export PS1="(chroot) $PS1"

First we need to uninstall the kernel so archlinux updates won't brick the board

pacman -R linux-aarch64

We need a user to create some missing programs

userdel alarm
useradd -m suzie

Here I create simple passwords, after we boot the micro sd we can change them to
more secured password. For all our settings in chroot this approach makes work simple. In
both case it will ask to confirm the password.

For the root password : passwd
For the suzie user password : passwd suzie


We need to do a system update
```

```
CheckSpace needs to be commented in /etc/pacman.conf

To enable mirrors, edit /etc/pacman. d/mirrorlist and locate your geographic region.
Uncomment mirrors you would like to use.
rm -r /etc/pacman.d/gnupg
pacman-key --init
pacman-key --populate archlinux
pacman -Syy
pacman -Syu
pacman -S base-devel
locale-gen
pacman -S wget subversion git

Set the locale in /etc/locale.conf to your language
Example :

LANG="fr_CA.UTF-8"
LC_COLLATE="C.UTF-8"

Then run this :
source /etc/profile

We need to create some package
One is joe which is similar to wordstar editor
I then create links to ws so simulate the old CPM/80 and dos wordstar

su suzie
cd /home/suzie

mkdir arch_packages
cd arch_packages
wget https://aur.archlinux.org/cgit/aur.git/snapshot/joe.tar.gz
wget https://aur.archlinux.org/cgit/aur.git/snapshot/systemd-gadget.tar.gz

tar xvf joe.tar.gz
rm joe.tar.gz
tar xvf systemd-gadget.tar.gz
rm systemd-gadget.tar.gz

cd joe....

Change arch to arch='aarch64' and run this :
makepkg
cd ../systemd-gadget
makepkg
exit
pacman -U /home/suzie/arch_packages/joe/joe-4.6-2-aarch64.pkg.tar.xz
pacman -U /home/suzie/arch_packages/systemd-gadget/systemd-gadget/systemd-gadget-0.0.1-
1-any.pkg.tar.xz

To leave chroot type exit
```

9. Create Arch Linux micro SD boot disk

```
export kernel_version=6.14.2-suzie-arm64-k3-r12
export input_dir=$work_directory/input
export rootfs_dir=$work_directory/arch_rootfs
export archlinux_dir=$work_directory/archlinux
```

9.1. Copy Kernel Files

```
su
we create a directory for Archlinux, first delete the old one if any

mkdir $archlinux_dir
cd $archlinux_dir
tar xvf ../pocketbeagle2-misc-boot-files.tar.xz

cd $rootfs_dir/etc
echo '/dev/mmcblk1p1  /boot vfat user,uid=1000,gid=1000,defaults 0 2' >> /fstab
echo '/dev/mmcblk1p2  /  ext4  noatime,errors=remount-ro  0  1' >> fstab

cd $rootfs_dir/lib
tar xvf $input_dir/modules-$kernel_version.tar.xz

cd $rootfs_dir/usr/src
tar xvf linux-$kernel_version-source.tar.xz linux-$kernel_version
ln -s linux-$kernel_version linux
cp $input_dir/config-$kernel_version ./

cd $rootfs_dir
sudo tar cvfJ $archlinux_dir/archlinux_pocketbeagle2.tar.xz *
exit
```

9.2. Create archlinux rootfs.ext4 image

```
We use some bash scripts to do the image
-------------------------------------------------------------
#!/bin/bash
#
# Script to create Archlinux rootfs.ext4 for the pocketbeagle2 board

# Copyright (C) 2025 Michel Catudal
# Michel Catudal <michelcatudal@gmail.com>
#
# SPDX-License-Identifier:      GPL-2.0+
#

# Force to english

LC_ALL=C

set -x # echo on

work_directory=$(pwd)
rootfs_file="$work_directory/archlinux-pocketbeagle2-rootfs.xz"

uncomp_size=$(xz --robot --list "$rootfs_file" | grep ^totals | cut -f5)
echo $uncomp_size
```

```
COUNT1="$(($uncomp_size/4000000))"
echo $COUNT1
COUNT="$(($COUNT1+150))"
echo $COUNT
dd if=/dev/zero of=$work_directory/input/rootfs.ext4 bs=4M count=$COUNT

# If the rootfs directory does not exist, it will be created
mkdir -p $work_directory/rootfs

mkfs.ext4 $work_directory/input/rootfs.ext4
mount $work_directory/input/rootfs.ext4 $work_directory/rootfs

echo "Extracting filesystem on micro SD image ..."
tar xvf $rootfs_file -C $work_directory/rootfs
sync
-------------------------------------------------------------

sudo ./mk_archlinux_rootfs.sh
sudo chown $USER:$USER input/rootfs.ext4

It creates a file name rootfs.ext4 located in directory input

genimage --rootpath `mktemp` --config genimage.cfg

go on root with su
Change sdd for whatever your micro SD is on
Make sure that it is unmounted

cd images
dd if=sdcard.img of=/dev/sdd status=progress iflag=direct oflag=direct bs=4M

Remove and put the micro SD back in the slot
Use gparted to expand the ext4 partition to fill the sd card
```

10. Gentoo Linux Root File System

```
export rootfs_dir=$work_directory/gentoo_rootfs

cd $work_directory
```

10.1. Download

```
Since this changes often it may be better to go to https://www.gentoo.org/downloads/
and choose the latest arm64 stage 3 openrc
We don't have a display so there is no need for the Destktop

latest_stage3=20250413T230515Z/stage3-arm64-openrc-20250413T230515Z.tar.xz

wget https://distfiles.gentoo.org/releases/arm64/autobuilds/$latest_stage3
```

10.2. Create a root file System

```
mkdir -p $rootfs_dir
sudo tar xfvp stage3-arm64-openrc-20250413T230515Z.tar.xz -C $rootfs_dir
sudo sync
sudo cp /usr/bin/qemu-aarch64 $rootfs_dir/usr/bin
sudo cp /etc/locale.gen $rootfs_dir/etc
```

```
sudo cp /etc/resolv.conf $rootfs_dir/etc

Create some alias :

echo 'alias ll='ls -alF'' > $rootfs_dir/root/.bashrc
echo 'alias la='ls -A''  >> $rootfs_dir/root/.bashrc
echo 'alias l='ls -CF''  >> $rootfs_dir/root/.bashrc
echo ''  >> $rootfs_dir/root/.bashrc
echo 'alias dir='ls -la -N --color''  >> $rootfs_dir/root/.bashrc
echo ''  >> $rootfs_dir/root/.bashrc
echo 'alias rm='rm -i'' >> $rootfs_dir/root/.bashrc
echo 'alias del='rm -i''  >> $rootfs_dir/root/.bashrc
echo ''  >> $rootfs_dir/root/.bashrc
echo 'alias rd=rmdir' >> $rootfs_dir/root/.bashrc
echo 'alias md='mkdir -p''  >> $rootfs_dir/root/.bashrc

echo '>=x11-libs/libxkbcommon-1.8.0 X' > $rootfs_dir/etc/package.use/X
echo '=app-editors/joe-4.6-r2 **' > $rootfs_dir/etc/package.accept_keywords/joe
```

10.3. chroot into gentoo rootfs

```
cd $work_directory
sudo arch-chroot $rootfs_dir
source /etc/profile
export PS1="(chroot) $PS1"

We need a user for later login thru ssh

useradd -m suzie

Here I create simple passwords, after we boot the micro sd we can change them to
more secured password. For all our settings in chroot this approach makes work simple. In
both case it will ask to confirm the password.

For the root password : passwd
For the suzie user password : passwd suzie

emerge-webrsync
eselect profile set 15
emaint --auto sync

Edit /etc/portage/make.conf
Example between ------:
Blocking of sandbox stuff is needed to be able to compile anything in chroot
You could remove it once you boot the disk and don't plan on using chroot on it in the
future

-----------------------------------
COMMON_FLAGS="-O2 -pipe"
CFLAGS="${COMMON_FLAGS}"
CXXFLAGS="${COMMON_FLAGS}"
FCFLAGS="${COMMON_FLAGS}"
FFLAGS="${COMMON_FLAGS}"

CHOST="aarch64-unknown-linux-gnu"
```

```
LINGUAS="fr fr_CA en en_US es es_AR es_BO es_CL es_CO es_CR es_CU
 es_DO es_EC es_ES es_GT es_HN es_MX es_NI es_PA es_PE
 es_PR es_PY es_SV es_US es_UY es_VE
 zh zh_CN zh_HK zh_SG zh_TW"

L10N="fr fr-CA en en-US es es-AR es-BO es-CL es-CO es-CR es-CU
 es-DO es-EC es-ES es-GT es-HN es-MX es-NI es-PA es-PE
 es-PR es-PY es-SV es-US es-UY es-VE
 zh zh-CN zh-HK zh-SG zh-TW"

ACCEPT_LICENSE="*"

FEATURES="-test -pid-sandbox -network-sandbox -sandbox -usersandbox -ipc-sandbox
-selinux -sesandbox -collision-detect"

USE="${ARCH} -zeitgeist -beagle -pcmcia -selinux -bindist buildpkg -pid-sandbox
-network-sandbox -sandbox -usersandbox -ipc-sandbox -sesandbox -seccomp -systemd dbus
elogind jpeg a52 gif x265 x264 -test pulseaudio qt6 tinfo gtk++ -bindist scanner -audit"

GENTOO_MIRRORS="ftp://mirrors.tera-byte.com/pub/gentoo \
http://gentoo.mirrors.tera-byte.com/ \
rsync://mirrors.tera-byte.com/gentoo \
ftp://mirror.csclub.uwaterloo.ca/gentoo-distfiles/ \
https://mirror.csclub.uwaterloo.ca/gentoo-distfiles/ \
http://mirror.csclub.uwaterloo.ca/gentoo-distfiles/ \
rsync://mirror.csclub.uwaterloo.ca/gentoo-distfiles \
https://mirror.clarkson.edu/gentoo/ \
http://mirror.clarkson.edu/gentoo/ \
rsync://mirror.clarkson.edu/gentoo/ \
http://www.gtlib.gatech.edu/pub/gentoo \
rsync://rsync.gtlib.gatech.edu/gentoo \
https://mirrors.mit.edu/gentoo-distfiles/ \
http://mirrors.mit.edu/gentoo-distfiles/ \
rsync://mirrors.mit.edu/gentoo-distfiles/ \
https://gentoo.osuosl.org/ \
http://gentoo.osuosl.org/ \
https://mirrors.rit.edu/gentoo/ \
http://mirrors.rit.edu/gentoo/ \
ftp://mirrors.rit.edu/gentoo/ \
rsync://mirrors.rit.edu/gentoo/ \
http://gentoo-mirror.flux.utah.edu/"

PORTDIR_OVERLAY="/usr/local/portage/suzie"
LC_MESSAGES=C.utf8
------------------------------------

For the suzie portage overlay

On this overlay there are two directories

suzie and metadata

The suzie repository has has two directories

profile and metadata

Both metadata directories have a file named layout.conf which contains :
```

```
masters = gentoo
auto-sync = false

The profiles has a file name repo_name which contains the word suzie

To set the locale you check which locales are available with:
eselect locale list. If your /etc/locale.gen file has few items the list would be short

[69]  fr_CA.UTF-8 *

If the one with the * is not the one you want you use the set command to the right one

Here I had selected number 69

For example, to set to Mandarin that would be 57

[57]  zh_CN.utf8

eeselect locale set 57

For the time eastern time zone

ln -s /usr/share/zoneinfo/America/Detroit /etc/localtime
emerge --ask joe

Setup some links to simulate the wordstar editor name

cd /usr/bin
ln -s joe ws
cd /etc/joe
cp jstarrc wsrc



This part will take quite a bit of time 115 programs to install

emerge --ask --verbose --update --deep --newuse @world
emerge --ask dev-vcs/git subversion

To leave chroot type exit
cd $work_directory/input/gentoo_rootfs
sudo tar cvfJ $work_directory/gentoo-pocketbeagle2-rootfs.xz *
```

11. Create Gentoo Linux micro SD boot disk

```
export kernel_version=6.14.2-suzie-arm64-k3-r12
export input_dir=$work_directory/input
export rootfs_dir=$work_directory/gentoo_rootfs
export gentoo_dir=$work_directory/gentoo
```

11.1. Copy Kernel Files

```
cd $work_directory

su

we create a directory for Gentoo, first delete the old one if any

mkdir $gentoo_dir
cd $gentoo_dir
tar xvf ../pocketbeagle2-misc-boot-files.tar.xz

cd $rootfs_dir/etc
echo '/dev/mmcblk1p1  /boot vfat user,uid=1000,gid=1000,defaults 0 2' >> fstab
echo '/dev/mmcblk1p2  /  ext4  noatime,errors=remount-ro  0  1' >> fstab

cd $rootfs_dir/lib
tar xvf $input_dir/modules-$kernel_version.tar.xz

cd $rootfs_dir/usr/src
tar xvf linux-$kernel_version-source.tar.xz linux-$kernel_version
ln -s linux-$kernel_version linux
cp $input_dir/config-$kernel_version ./

exit
```

11.2. Create gentoo rootfs.ext4 image

```
We use a bash script to do the image
-------------------------------------------------------------
#!/bin/bash
#
# Script to create Gentoo rootfs.ext4 for the pocketbeagle2 board

# Copyright (C) 2025 Michel Catudal
# Michel Catudal <michelcatudal@gmail.com>
#
# SPDX-License-Identifier:      GPL-2.0+
#

# Force to english

LC_ALL=C

set -x # echo on

work_directory=$(pwd)
rootfs_file="$work_directory/gentoo-pocketbeagle2-rootfs.xz"

uncomp_size=$(xz --robot --list "$rootfs_file" | grep ^totals | cut -f5)
echo $uncomp_size
COUNT1="$(($uncomp_size/4000000))"
echo $COUNT1
COUNT="$(($COUNT1+150))"
echo $COUNT
dd if=/dev/zero of=$work_directory/input/rootfs.ext4 bs=4M count=$COUNT
```

```
# If the rootfs directory does not exist, it will be created
mkdir -p $work_directory/rootfs

mkfs.ext4 $work_directory/input/rootfs.ext4
mount $work_directory/input/rootfs.ext4 $work_directory/rootfs

echo "Extracting filesystem on micro SD image ..."
tar xvf $rootfs_file -C $work_directory/rootfs
sync
------------------------------------------------------------

sudo ./mk_gentoo_rootfs.sh
sudo chown $USER:$USER input/rootfs.ext4

It creates a file name rootfs.ext4 located in directory input

genimage --rootpath `mktemp` --config genimage.cfg

go on root with su
Change sdd for whatever your micro SD is on
Make sure that it is unmounted

cd images

dd if=sdcard.img of=/dev/sdd status=progress iflag=direct oflag=direct bs=4M

Remove and put the micro SD back in the slot
Use gparted to expand the ext4 partition to fill the sd card
```