Suzie Linux https://suzielinux.com/

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

Gentoo Linux for Beagleplay boards documentation

Author	Date	Project	Revisions
Michel Catudal	2025-05-25	**Beagleplay Gentoo Linux Creation**	**1**

REVISION TRACKING SHEET

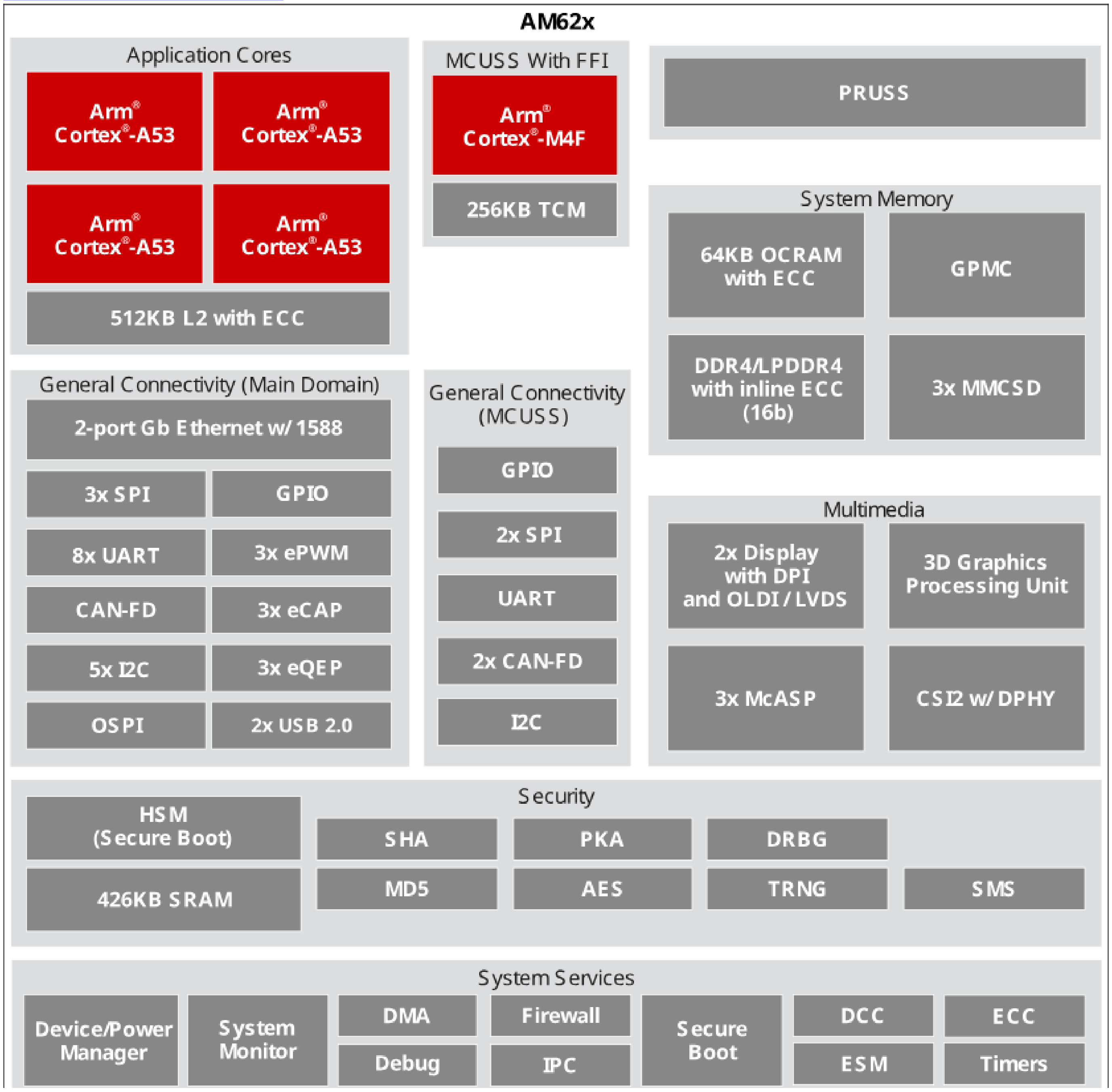
Rev	Name	Date	Comment
1	Michel Catudal	2025-05-25	

Table of Content

1. Hardware

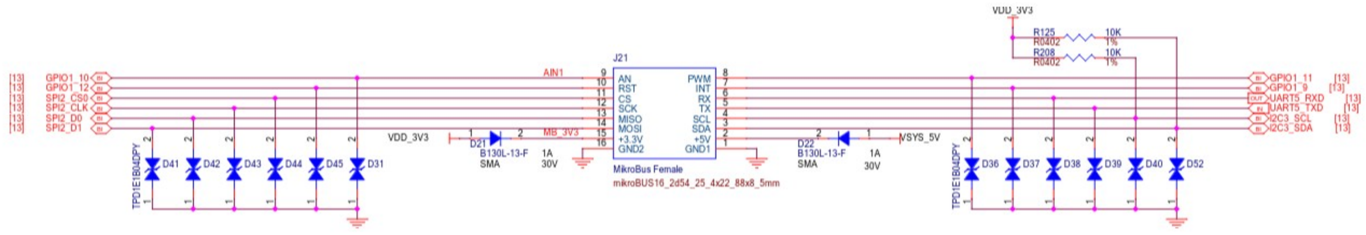
1.1. Overview of the beagleplay board

AM62x Sitara™ Processors from Texas Instruments are Human-machine-interaction SoC with Arm® Cortex®-A53-based edge AI and full-HD dual display. AM6254 which is on your BeaglePlay board has a multi core design with Quad 64-bit Arm® Cortex®-A53 microprocessor subsystem at up to 1.4 GHz, Single-core Arm® Cortex®-M4F MCU at up to 400MHz, and Dedicated Device/Power Manager. Talking about the multimedia capabilities of the processor you can connect upto two display monitors with 1920x1080 @ 60fps each, additionally there is a OLDI/LVDS (4 lanes - 2x) and 24-bit RGB parallel interface for connecting external display panels. One 4 Lane CSI camera interface is also available which has support for 1,2,3 or 4 data lane mode up to 2.5Gbps speed. The list of features is very long and if you are interested to know more about the AM62x SoC you may take a look at AM62x Sitara™ Processors datasheet.

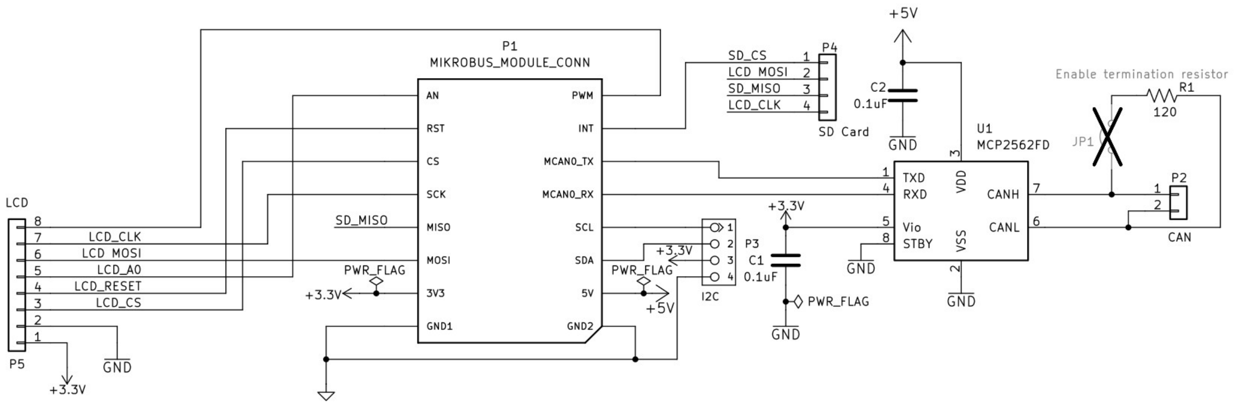


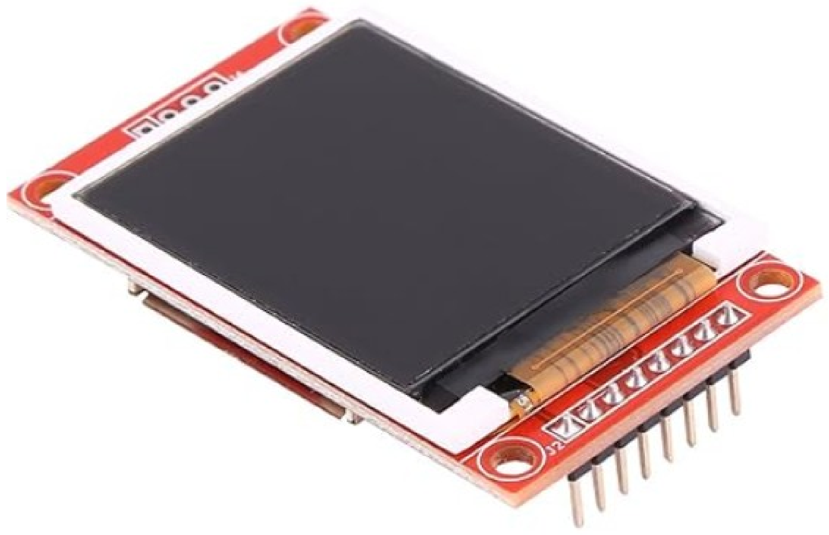
1.1.1. MIKROBUS

mikroBUS is a standard specification by MikroElektronika that can be freely used by anyone following the guidelines. It includes SPI, I2C, UART, PWM, ADC, reset, interrupt, and power (3.3V and 5V) connections to common embedded peripherals.



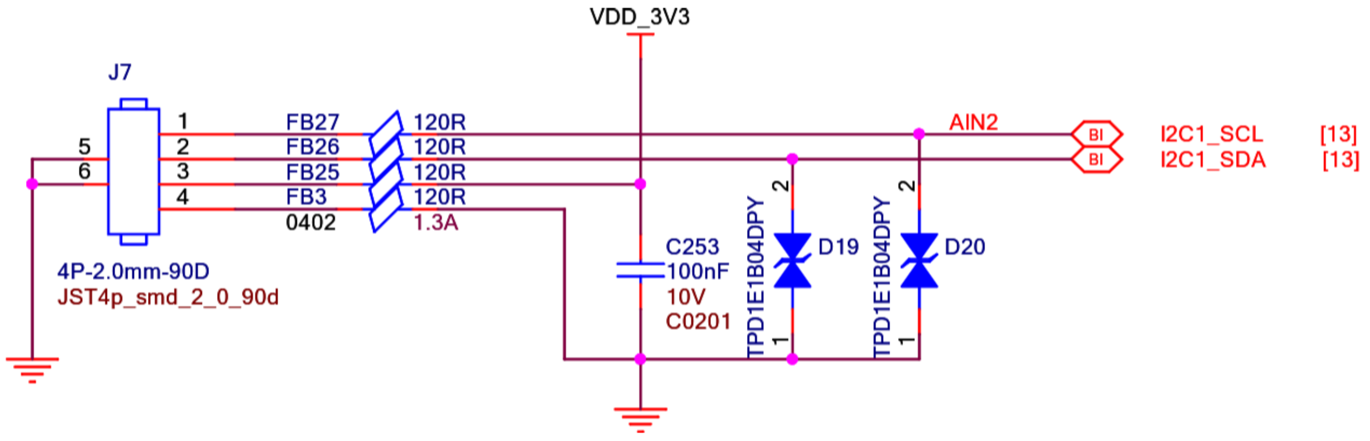
Adapter board to add small LCD and CAN drivers. The unused I2C port is connected to a grove connector.





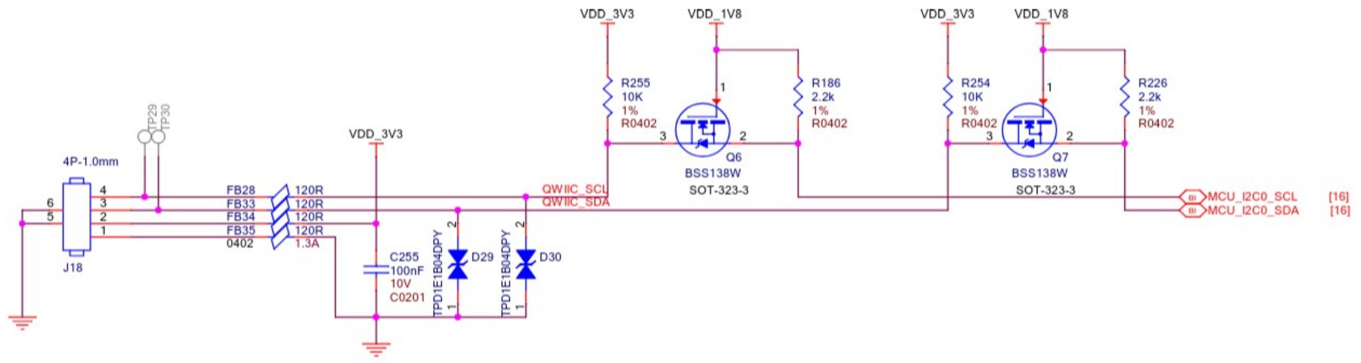
1.1.2. GROVE

Seeed Studio Grove System is a modular, standardized connector prototyping ecosystem. The Grove System takes a building block approach to assembling electronics. Compared to the jumper or solder based system, it is easier to connect devices to an application, simplifying the learning system



1.1.3. QWIIC

Qwiic, or STEMMA QT are 4pin JST SH 1.00 connectors for easy I2C connection.



2. Gentoo applications required for chroot

```
cd ~
mkdir beagleplay
cd beagleplay
export work_directory=$(pwd)
Get misc files needed for the bootloader and rootfs
git clone https://github.com/SuzieLinux/Beagleplay.git files
```

```
2.1. Gentoo applications required
```

```
emerge --ask dev-python/cryptography
emerge --ask dev-python/pyelftools
emerge --ask dev-util/yamllint
emerge --ask dev-libs/libyaml
emerge --ask dev-python/jsonschema
emerge --ask sys-block/bmap-tools
emerge --ask sys-fs/genimage
emerge --ask sys-fs/mtools
emerge --ask gnutls
emerge --ask flex
emerge --ask sys-devel/bc
emerge --ask bison
emerge --ask swig
emerge --ask sys-fs/dosfstools
cd /usr/bin
ln -s mkfs.vfat mkdosfs
emerge --ask sys-apps/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

Add this to /etc/portage/package.use/qemu :

```
# Enable static-user and add the arm64 and other targets
app-emulation/qemu static-user QEMU_SOFTMMU_TARGETS: * QEMU_USER_TARGETS: *
# required by app-emulation/qemu::gentoo[static,static-user]
# required by qemu (argument)
dev-libs/glib static-libs
# required by app-emulation/qemu::gentoo[-static,static-user]
# required by qemu (argument)
sys-libs/zlib static-libs
# required by app-emulation/qemu::gentoo[-static,static-user,xattr]
# required by qemu (argument)
sys-apps/attr static-libs
# required by dev-libs/glib::gentoo
# required by app-emulation/qemu::gentoo[-static,static-user]
# required by qemu (argument)
dev-libs/libpcre2 static-libs
```

```
emerge --ask app-emulation/qemu --update --newuse --deep
```

2.2. Applications required for chroot on mac vmware fusion debian

```
sudo apt upgrade
sudo apt install build-essential git vim
sudo apt install gfortran gpc
sudo apt install debhelper fakeroot
sudo apt install python3-cryptography
sudo apt install python3-pyelftools
sudo apt install yamllint
sudo apt install libyaml
sudo apt install libyaml-dev
sudo apt install python3-pyelftools
sudo apt install python3-jsonschema
sudo apt install python-jsonschema
sudo apt install bmap-tools
sudo apt install genimage
sudo apt install dosfstools
sudo apt install mtools
sudo apt install gnutls-dev
sudo apt install flex
sudo apt install bc
sudo apt install bison
sudo apt install swig
sudo apt install arch-chroot-scripts
```

3. Bootloader

3.1. Get the cross compilers

We compile the bootloader as a user on gentoo
we go to a directory where we will install the files

If not allready installed

```
cd $HOME
mkdir -p toolchains
cd toolchains

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

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

cd $work_directory
cp files/scripts/build_u-boot.sh ./
```

3.2. Build

```
chmod a+x build_u-boot.sh
./build_u-boot.sh
```

The generated files can be found on the directory public:

```
bl31.bin
tee-pager_v2.bin
tiboot3.bin
tispl.bin
u-boot.img
```

4. Gentoo Linux Root File System

```
export rootfs_dir=$work_directory/gentoo_rootfs
cd $ rootfs_dir
```

Since this changes often it may be better to go to https://www.gentoo.org/downloads/ and choose the latest arm64 stage 3 openrc

```
latest_stage3=20250427T235504Z/stage3-arm64-desktop-openrc-20250427T235504Z.tar.xz
wget https://distfiles.gentoo.org/releases/arm64/autobuilds/$latest_stage3
```

4.1. Create a root file System

```
su
mkdir -p $rootfs_dir
tar xfvp stage3-arm64-desktop-openrc-20250427T235504Z.tar.xz -C $rootfs_dir
sync
```

You may want to edit the locale env and keymaps if your language is not French

```
cp /usr/bin/qemu-aarch64 $rootfs_dir/usr/bin
cd $rootfs_dir/etc
cp $work_directory/files/etc/locale.gen ./
cp $work_directory/files/etc/env.d/02locale env.d
cp $work_directory/files/etc/conf.d/keymaps conf.d
cp /etc/resolv.conf ./
cp $work_directory/files/scripts/16-set-alias.bash bash/bashrc.d
cp $work_directory/files/etc/fstab ./
cp $work_directory/files/etc/inittab ./
cd portage
cp $work_directory/files/misc/etc/portage/make.conf ./
cd package.accept_keywords
cp $work_directory/files/misc/etc/portage/package.accept_keywords/* ./
cd ../package.use
cp $work_directory/files/misc/etc/portage/package.use ./

kernel_version=linux-6.15-rc7-catu

cd $work_directory/$rootfs_dir/usr/src
wget https://git.kernel.org/torvalds/tar.gz
tar xvf linux-6.15-rc7.tar.gz $kernel_version
cd $kernel_version
cp $work_directory/files/misc/config-$kernel_version .config
```

4.2. chroot into gentoo rootfs

```
cd $work_directory
```

It is assumed here that you are still under root

```
arch-chroot $rootfs_dir
source /etc/profile
```

This is needed when chroot on gentoo but is not needed on the mac
It actually crashes chroot if you do

```
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 20
emaint --auto sync
```

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

For the time eastern time zone

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

Setup some links to simulate the cpm-80 wordstar editor

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

To remove the annoying wordwrap bug delete all mentions of wordwrap in wsrc
It gets to be a pain when you update a script and it cuts a line and you didn't notice
Your script has no chance of working with this ridiculous behavior of the editor.

This part will take quite a bit of time if many programs need to be installed
Which is why it is always important to download the latest
It can be much faster if you chroot on a fast arm64 board or mac with vmware fusion.

```
emerge --ask --verbose --update --deep --newuse @world
emerge --ask dev-vcs/git subversion
emerge --ask openssh
rc-update add sshd default
gpasswd -a suzie wheel
```

If you want to be able to ssh as root add this line to /etc/ssh/sshd_config :

```
PermitRootLogin yes
emerge --ask wireless-regdb
emerge --ask linux-firmware
emerge --ask lightdm
emerge --ask display_manager
emerge --ask xorg-server
emerge --ask caja libmatekbd mate mate-applets mate-applets-meta mate-common mate-
control-center mate-desktop mate-menus mate-panel mate-session-manager mate-settings-
daemon caja-actions caja-extensions mate-calc mate-indicator-applet mate-media mate-
polkit mate-power-manager mate-screensaver mate-sensors-applet mate-system-monitor
mate-user-share mate-utils
emerge --ask dbus libdbus
emerge --ask dbus-monitorrc-update add dbus default
rc-update add display-manager default
emerge --ask net-misc/ntp
rc-update add hwclock boot
rc-update add ntp-client default
```

In /etc/init.d/ntp-client add sleep 10, this will delay trying to get the time for 10 seconds.
It could be lowered if the network is on quicker than that.

```
start() {
        checkconfig || return $?
        # Delay NTP client startup by 10 seconds
        sleep 10

        ebegin "Setting clock via the NTP client '${NTPCLIENT_CMD}'"
        "${NTPCLIENT_CMD}" ${NTPCLIENT_OPTS}
        eend $? "Failed to set clock"
}
```

This part could take 8-16 hours if not done on the mac
It will take close to an hour on the mac, maybe a little bit more.

```
cd /usr/src/$kernel_version
make menuconfig
make
make dtbs
make modules_install
make dtbs_install
make install
```

You may want to delete the compiled files to reduce the size of the image

```
make mrproper
```

```
cd /boot
cp vmlinuz-$kernel_version Image.gz
gunzip Image.gz
```

To leave chroot type exit

```
cd $work_directory/input/gentoo_rootfs
NOW=$(date +"%Y%m%d%H%M")
sudo tar cvfJ $work_directory/gentoo-beagleplay-rootfs-$NOW.xz *
```

5. Create Gentoo Linux micro SD boot disk

If chroot was not done on gentoo it needs to be copied to the gentoo disk
Make sure your running as root

```
su
cd ~/beagleplay
export work_directory=$(pwd)
export NOW=$(date +"%Y%m%d%H%M")
linux_image=gentoo-beagleplay-mate-$NOW.img
```

Replace the date and time with the one used when compressing

```
tar xvf gentoo-beagleplay-rootfs-202505261217.xz

export rootfs=gentoo_rootfs
cd $gentoo_rootfs/boot
mkdir -p extlinux
cp $work_directory/files/misc/boot/extlinux/extlinux.conf extlinux
```

If you changed the name of the kernel you need to edit the file extlinux.conf

Copy the u-boot files on the boot directory

```
cp $work_directory/public/* ./
```

Leave root mode

```
exit

cd $work_directory
cp files/scripts/mk_gentoo_rootfs.sh ./
chmod a+x mk_gentoo_rootfs.sh
mkdir -p input
sudo ./mk_gentoo_rootfs.sh
sudo chown $USER:$USER input/rootfs.ext4
cp files/scripts/genimage.cfg ./
genimage --rootpath `mktemp` --config genimage.cfg
cd images
mv images/sdcard.img $linux_image
```

change sdd for whatever your microsd is

```
sudo dd if=$linux_image of=/dev/sdd status=progress iflag=direct oflag=direct bs=1M
```