Содержание

Подготовка программной части для модуля NMS-SM-RK3568 без блобов (opensource), Kernel 6.1

Собирать можно либо в ubuntu(20.04/22.04) либо в docker.

Сборка в Ubuntu 20.04/22.04

Подготовка

```
wget https://buildroot.org/downloads/buildroot-2023.11.1.tar.gz
git clone -b nms-sm-rk3568
https://github.com/inmys/buildroot-external-inmys
tar -xf buildroot-2023.11.1.tar.gz
make BR2_EXTERNAL=$PWD/buildroot-external-inmys -C buildroot-2023.11.1
O=$PWD/output br_defconfig
```

Сборка

```
cd output
make
# ждём несколько часов
```

Если в процессе сборки, лог завис на этапе скачивания qemu, то нужно самостоятельно скачать QEMU и положить в папку buildroot-2023.02.1/dl

Результат:

```
output/images/Image
output/images/rk3568-inmys-smarc-evm.dtb
rootfs.cpio.gz
rootfs.ext2.gz
```

Изменение и пересборка rootfs

```
make menuconfig
make
```

Пересборка kernel (Image/dtb)

```
make linux-rebuild
```

Подготовить toolchain с помощью которого можно кросскомпилировать программы (например Qt)

```
make sdk
```

Результат:

```
output/images/aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz
```

Сборка в Docker

Для выполнения следующей инструкции, на компьютере должно быть установлено Docker окружение

Подготовка

```
wget https://buildroot.org/downloads/buildroot-2023.11.1.tar.gz
git clone -b nms-sm-rk3568
https://github.com/inmys/buildroot-external-inmys
tar -xf buildroot-2023.11.1.tar.gz
# build docker image with name "buildroot-2023.11.1"
docker build -t buildroot-2023.11.1 buildroot-2023.11.1/support/docker
# initial config rootfs
docker run --mount type=bind,source="$(pwd)",target=/workdir --user
"$(id -u):$(id -g)" -it buildroot-2023.11.1 make
BR2_EXTERNAL=/workdir/buildroot-external-inmys -C
/workdir/buildroot-2023.11.1 O=/workdir/output br_defconfig
```

Сборка

```
cd output
docker run --mount type=bind,source="$(pwd)",target=/workdir --user
"$(id -u):$(id -g)" -it buildroot-2023.02.1 make -C /workdir/output
# ждём несколько часов
```

Результат:

```
output/images/Image
output/images/rk3568-inmys-smarc-evm.dtb
rootfs.cpio.gz
rootfs.ext2.gz
```

Изменение и пересборка rootfs

```
docker run --mount type=bind,source="$(pwd)",target=/workdir --user
"$(id -u):$(id -g)" -it buildroot-2023.02.1 make -C /workdir/output
menuconfig
docker run --mount type=bind,source="$(pwd)",target=/workdir --user
"$(id -u):$(id -g)" -it buildroot-2023.02.1 make -C /workdir/output
```

Пересборка kernel (Image/dtb)

```
docker run --mount type=bind,source="$(pwd)",target=/workdir --user
"$(id -u):$(id -g)" -it buildroot-2023.02.1 make -C /workdir/output
```

```
linux-rebuild
```

Подготовить toolchain с помощью которого можно кросскомпилировать программы (например Qt)

```
docker run --mount type=bind,source="$(pwd)",target=/workdir --user
"$(id -u):$(id -g)" -it buildroot-2023.02.1 make -C /workdir/output sdk
```

Результат:

```
output/images/aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz
```

Дополнительно

Создание загрузочной SD карты

Скачать

burn.tar.xz

```
tar -xf burn.tar.gz
```

скопировать в burn файлы: Image rk3568-inmys-smarc-evm.dtb rootfs.cpio.gz

```
sudo ./burn_sd.sh /dev/sdX
```

ГДЕ ВМЕСТО X БУКВА КАРТЫ ПАМЯТИ

Консольный uart

Консольный uart конфигугрируется на скорости 1500000.

Использование toolchain для кросс-компиляции

Для кросс-компиляции нужен компьютер x86_64(amd64) с установленным linux, подойдут: ubuntu,debian.

```
tar -xf aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz
cd aarch64-buildroot-linux-gnu_sdk-buildroot/
./relocate-sdk.sh
```

Теперь:

- aarch64-buildroot-linux-gnu_sdk-buildroot/bin/qmake можно использовать для сборки qt проектов.
- aarch64-buildroot-linux-gnu_sdk-buildroot/share/buildroot/toolchainfile.cmake для сборки cmake проектов.

о том как настроить qt creator для кросс-компиляции написано здесь (host.tar.gz ==

aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz)
https://wiki.inmys.ru/doku.php?id=boards:nms-hh-px30:bsp

Уже собранный aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz : https://disk.yandex.ru/d/P4YAk8zqebVlNg