Содержание

Кросс-компиляция приложений для Q7_KOMDIV_MK (Buildroot SDK) 3
Параметры платформы 3
Способ 1: Готовый SDK 3
1. Загрузка SDK 3
2. Распаковка и настройка 4
3. Пример: Hello World на C++ 4
4. Проверка бинарника на хосте 5
5. Копирование на плату и запуск 5
Способ 2: Сборка SDK из исходников 5
Частые проблемы и решения 6

Кросс-компиляция приложений для Q7_KOMDIV_MK (Buildroot SDK)

В этой инструкции описаны два сценария сборки собственных приложений для модуля **Q7_KOMDIV_MK**:

1. **Быстрый старт с готовым SDK:** скачать архив mips64el-buildroot-linux-gnu_sdk-buildroot.tar.gz, распаковать и собирать приложения компилятором из SDK.
2. **Сборка SDK из исходников:** собрать SDK вместе с rootfs в Docker-контейнере `komdiv-sdk` (если нужен SDK под изменённую конфигурацию).

> **Важно:** приложения нужно собирать **компилятором из SDK**, а не компилятором из дистрибутива хоста. SDK собран из того же Buildroot, что и rootfs на плате, поэтому приложение гарантированно совместимо с библиотеками модуля (`libstdc++`, `libc` и другими). Бинарник, собранный другим компилятором, может не запуститься.

Параметры платформы

Параметр	Значение
Процессор	K5500BK018 («Комдив-МК»), KOMDIV-64 (MIPS64)
Формат исполняемых файлов	ELF64, little endian, ABI n64
Библиотека C	glibc
Динамический загрузчик	`/lib64/ld.so.1`
Префикс компилятора	`mips64el-buildroot-linux-gnu-`
Версия Buildroot	2025.02.7
Хостовая ОС для SDK	Linux x86_64 (Ubuntu 24.04)

Способ 1: Готовый SDK

1. Загрузка SDK

Скачайте архив SDK (около 225 МБ):

- Ссылка: mips64el-buildroot-linux-gnu_sdk-buildroot.tar.gz
- Пароль: Отправляется на почту.
- Контрольная сумма:

```
sha256:
4839c96f980299625a58cf7d1021dd3c7b4cf35646eba295152b52b93c6881c5
```

Проверка после скачивания:

```
sha256sum mips64el-buildroot-linux-gnu_sdk-buildroot.tar.gz
```

2. Распаковка и настройка

Распакуйте архив в каталог, в пути к которому нет пробелов и кириллицы (например, ~/sdk):

```
mkdir -p ~/sdk && cd ~/sdk
tar xf ~/Downloads/mips64el-buildroot-linux-gnu_sdk-buildroot.tar.gz
cd mips64el-buildroot-linux-gnu_sdk-buildroot
./relocate-sdk.sh
export SDK=$(pwd)
```

Скрипт relocate-sdk.sh запускается **один раз после распаковки**: он прописывает в SDK новые пути. Если каталог с SDK переместить, скрипт нужно запустить заново. Переменная SDK нужна в каждой новой консоли.

Проверьте, что компилятор работает:

```
$SDK/bin/mips64el-buildroot-linux-gnu-g++ --version
```

3. Пример: Hello World на C++

Создайте файл helloworld.cpp:

```
#include <iostream>

int main()
{
    std::cout << "Hello, KOMDIV!" << std::endl;
    return 0;
}
```

Вариант A. Прямой вызов компилятора:

```
$SDK/bin/mips64el-buildroot-linux-gnu-g++ -O2 -o helloworld
helloworld.cpp
```

Вариант B. Через CMake (для проектов из нескольких файлов). Создайте CMakeLists.txt:

```
cmake_minimum_required(VERSION 3.10)
project(helloworld CXX)
add_executable(helloworld helloworld.cpp)
```

Соберите проект, указав файл конфигурации тулчейна из SDK:

```
mkdir build && cd build
cmake -DCMAKE_TOOLCHAIN_FILE="$SDK/share/buildroot/toolchainfile.cmake"
..
```

```
make
```

4. Проверка бинарника на хосте

Перед копированием на плату убедитесь, что получен файл для нужной архитектуры:

```
file helloworld
LC_ALL=C readelf -h -A helloworld | grep -E
"Class|Data|Machine|Flags|FP ABI"
LC_ALL=C readelf -l helloworld | grep interpreter
od -An -tx1 -j18 -N2 helloworld
```

Ожидаемый результат:

Что проверяем	Ожидаемое значение
Class	ELF64
Data	2's complement, little endian
Machine	MIPS R3000 (так утилита называет семейство MIPS)
Flags	0x60000007, noreorder, pic, cpic, mips64
FP ABI	Soft float
interpreter	/lib64/ld.so.1
Байты 18–19 заголовка (od)	08 00

> **Примечание:** LC_ALL=C нужен, чтобы вывод readelf был на английском и совпадал с таблицей.

5. Копирование на плату и запуск

Скопируйте файл на плату любым удобным способом. Запустите приложение на плате (логин/пароль root/root):

```
chmod +x /opt/helloworld
/opt/helloworld
```

Ожидаемый вывод:

```
Hello, KOMDIV!
```

Способ 2: Сборка SDK из исходников

Подходит, если нужен SDK под изменённую конфигурацию rootfs (другие библиотеки, свои пакеты).

1. Выполните сборку образа системы по руководству пользователя (раздел «Сборка

образа системы»): Docker-контейнер `komdiv-sdk`, Buildroot 2025.02.7 и `buildroot-external-inmys` (ветка `nms-q7-k5500vk018`).
2. В контейнере в каталоге `output` выполните:

```
cd output
export FORCE_UNSAFE_CONFIGURE=1
make -j12
make sdk
```

3. После завершения сборки SDK находится в `output/images/`:

```
output/images/mips64el-buildroot-linux-gnu_sdk-buildroot.tar.gz
```

Дальше используйте его так же, как в Способе 1. Первая сборка занимает от одного до нескольких часов.

> **Примечание:** для сборки нужен доступ к приватным репозиториям ядра и загрузчика (см. раздел «Состав программной части» руководства пользователя). Токены доступа нигде не публикуйте и не добавляйте в git файл `.gitlab.netrc`.

Частые проблемы и решения

- **`error while loading shared libraries: libstdc++.so.6: cannot open shared object file: No such file or directory`, хотя файл лежит в `/usr/lib`:** программа собрана не компилятором из SDK. Загрузчик отвергает библиотеки, несовместимые с программой по ABI, и сообщение при этом выглядит так, будто файла нет. Соберите программу компилятором из SDK. Временное решение: добавьте ключи `-static-libstdc++ -static-libgcc`, тогда библиотеки войдут в бинарник.
- **`./helloworld: not found` или `No such file or directory` при запуске:** на плате нет динамического загрузчика, указанного в бинарнике. Проверьте `readelf -l helloworld | grep interpreter` (ожидается `/lib64/ld.so.1`) и наличие файла на плате: `ls -l /lib64/ld.so.1`.
- **`Exec format error`:** другая архитектура или порядок байт. Проверьте `file helloworld`.
- **`Illegal instruction`:** компилятор сгенерировал инструкции, которых нет в процессоре. Не добавляйте в сборку ключи `-march`, `-mhard-float` и подобные, используйте настройки по умолчанию из SDK.
- **Скрипт `relocate-sdk.sh` не найден:** проверьте, что вы находитесь в распакованном каталоге SDK (`ls` должен показать `bin`, `share` и сам скрипт).