

## Содержание

<b>Подготовка программной части</b> .....	3
<b>Сборка Linux при помощи Docker</b> .....	3
<b>Подготовка к сборке</b> .....	3
<b>Сборка spl/MLO/u-boot</b> .....	3
<b>Сборка linux</b> .....	4
<b>Сборка rootfs</b> .....	4
<b>Прошивка пустого модуля</b> .....	4
<b>Настройка rootfs</b> .....	5



# Подготовка программной части

## Сборка Linux при помощи Docker

Для выполнения следующей инструкции, на компьютере должно быть установлено [Docker окружение](#)

Также необходимо скачать следующие файлы:

- Набор скриптов и образ контейнера
  - U-boot patch
  - Kernel patch
    - Rootfs
- [Standalone ARM Toolchain\\*](#)
- [AM335x Linux SDK BSP Source Code\\*](#)

\* файлы скачаны с сайта [ti.com](http://ti.com)

Рекомендация: Удобнее скачать файлы в отдельный каталог и работать в этой папке. Для этого, необходимо переместить скачанные архивы в новую папку и перейти в неё в терминале. Распаковать архивы

```
tar -xf u-boot_mir.tar
tar -xf kernel_mir.tar
tar -xf rootfs.tar.gz
tar -xf utils.tar.gz
tar -xf am335x-evm-linux-sdk-src-06.03.00.106.tar.xz
tar -xf gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf.tar.xz
```

Для подготовки образа контейнера, необходимо перейти в папку `utils` и выполнить команду сборки образа

```
sudo docker build -t am335x utils/
```

После этого в `docker` появится образ **am335x**

## Подготовка к сборке

Скопируйте два скрипта из каталога `utils` в текущую директорию

```
cp utils/kernel.sh .
cp utils/u-boot.sh .
```

## Сборка `spl/MLO/u-boot`

Выполните команду

```
sudo docker run -e USER=$USER -e USERID=$UID -v $(pwd):/BR -t am335x
bash u-boot.sh
```

## Сборка linux

Выполните команду

```
sudo docker run -e USER=$USER -e USERID=$UID -v $(pwd):/BR -t am335x  
bash kernel.sh
```

Сборка может занять длительное время, в зависимости от конфигурации компьютера.

После сборки, появятся файлы

- Board-support/linux-4.19.94+gitAUTOINC+be5389fd85-gbe5389fd85/arch/arm/boot/zImage
- Board-support/linux-4.19.94+gitAUTOINC+be5389fd85-gbe5389fd85/arch/arm/boot/dts/am335x-mir\_mb.dtb

Проверить их наличие можно командой:

```
ls board-support/linux-4.19.94+gitAUTOINC+be5389fd85-  
gbe5389fd85/arch/arm/boot/zImage  
ls board-support/linux-4.19.94+gitAUTOINC+be5389fd85-  
gbe5389fd85/arch/arm/boot/dts/am335x-mir_mb.dtb
```

## Сборка rootfs

Выполните команду

```
sudo docker run -e USER=$USER -e USERID=$UID -v $(pwd):/BR -t am335x  
make -C rootfs
```

Сборка может занять длительное время, в зависимости от конфигурации компьютера.

После сборки, появится файл [rootfs/buildroot-2018.05.1/output/images/rootfs.cpio.uboot](#)

## Прошивка пустого модуля

[собранные файлы](#)

bins

```
cd board-support/u-boot-2019.01+gitAUTOINC+333c3e72d3-g333c3e72d3  
picocom -b 115200 --send-cmd "sx -vv " /dev/ttyUSB1  
#ctrl a+s -> send by Xmodem spl/u-boot-spl.bin  
spl/u-boot-spl.bin  
#ctrl a+s -> send by Xmodem spl/u-boot.bin  
u-boot.bin
```

```
setenv ipaddr 192.168.1.198
setenv serverip 192.168.1.125
tftp ${rdaddr} mir_am33/rootfs.cpio.uboot
tftp ${loadaddr} mir_am33/zImage
tftp ${fdtaddr} mir_am33/am335x-mir_mb.dtb
setenv bootargs "console=ttyS0,115200n8 earlyprintk debug"
bootz ${loadaddr} ${rdaddr} ${fdtaddr}
#login with root/root
fdisk /dev/mmcblk0
n
p
1

+1G
a
1
t
c
n
p
2

w
####
mkfs.vfat /dev/mmcblk0p1
mkfs.vfat -n STORE /dev/mmcblk0p2
mount /dev/mmcblk0p1 /opt/
copy from build machine to /opt on som: am335x-mir_mb.dtb MLO
rootfs.cpio.uboot u-boot.img zImage
#scp <...>/bin/results/ root@192.168.1.198:/opt/umount /opt
sync
```

## Настройка rootfs

Изменение ip адресов

```
savevar.sh kernel NETWORK_IP 192.168.1.198 # ip on eth0
savevar.sh kernel NETWORK_IP_1 192.168.2.198 # ip on eth1
savevar.sh kernel NETWORK_IP_2 192.168.3.198 # ip on eth2
```

Для добавление автозапуска скриптов, их нужно записать в файл `/mnt/store/postup.sh`

**Замечание:** после перезагрузки, данные в `/mnt/store` сохранятся, но всё остальное будет очищено. Соответственно пользовательские файлы должны хранится в `/mnt/store`.

Настройки хранятся в файлах `/mnt/store/settings/kernel` и `/mnt/store/settings/userspace`

*/mnt/store/settings/kernel*

```
NETWORK_MODE=<mode>

#mode=static - use NETWORK_IP, NETWORK_MASK, NETWORK_DEFAULT_GATEWAY
for network setup

#mode=dhcp - get ip/dns via dhcp

#default: static

#example: NETWORK_MODE=static

NETWORK_IP=<ip>

#default: 192.168.1.198

#example: NETWORK_IP=192.168.1.198

NETWORK_MASK=<mask>

#default: 255.255.255.0

#example: NETWORK_MASK=255.255.255.0

NETWORK_DEFAULT_GATEWAY=<ip>

#default: ""

#example: NETWORK_DEFAULT_GATEWAY=192.168.1.1
```

*/mnt/store/settings/userspace*

```
CHANGEROOTPASSWD=<false|true>

# use LOGIN_ROOTPASS or not

#default: false

#example: CHANGEROOTPASSWD=true

LOGIN_ROOTPASS=<enc_passw>

#enc_passw - result of crypt from glibc password manipulation (md5 hash
of the password and salt)

#default: $1$.1o2Wll.$rZsjTFfjqfJAZuLeWkpuj0

# default value - hash of 'root' password
```

```
#example: LOGIN_ROOTPASS='$1$.1o2Wll.$rZsjTFfqj fJAZuLeWkpuj0'
```

Пароль может быть изменён двумя способами:

1. Через `/mnt/store/settings/userspace`

Вызываем из glibc функцию `crypt` и результат записываем в `LOGIN_ROOTPASS` (и `CHANGEROOTPASSWD=true`)

Можно выполнить команду `passwd` из консоли и посмотреть результат в `/etc/passwd`.

2. В `/mnt/store/postup.sh`

```
echo root:123 | chpasswd
```

У пользователя `root` будет пароль `123`