iMX233-OLinuXino

This is a page about Olimex's Freescale based i.MX233 Boards; iMX233-OLinuXino.

Availability

Boards:
IMX233-OLINUXINO-MICRO at Digi-Key

Accessories:
USB-SERIAL-CABLE-F at Digi-Key

Kit Notes

iMX233-OLinuXino-Micro:

From: PDF Manual

Important note about owners of revision B of the board: if you are one of the first owners of iMX233-OLinuXino-Micro and you experience random hang-ups (Kernel oops, Kernel panic) it is recommended to unsolder/remove R17 (check the schematic or the board file to locate it easier). Removing R17 fixes the random lock-up.
This problem has been fixed in revision B1 of the board.

Vendor Documentation

- Olimex Documentation: https://www.olimex.com
  - Documentation: https://github.com/OLIMEX/OLINUXINO
  - Forums: https://www.olimex.com/forum/index.php?board=1.0

Basic Requirements

- Running a recent supported release of Debian, Fedora or Ubuntu on a x86 64bit based PC; without OS Virtualization Software.
- Many of the listed commands assume /bin/bash as the default shell.
- ARM Cross Compiler – Linaro: https://www.linaro.org
  - Linaro Toolchain Binaries: https://www.linaro.org/downloads/
- Bootloader
  - Das U-Boot – the Universal Boot Loader: http://www.denx.de/wiki/U-Boot
  - Source: https://github.com/u-boot/u-boot/
- Linux Kernel
  - Linus's Mainline tree: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
- ARM based rootfs
  - Debian: https://www.debian.org

ARM Cross Compiler: GCC

This is a pre-built (64bit) version of GCC that runs on generic linux, sorry (32bit) x86 users, it's time to upgrade...
Download/Extract:

user@localhost:~$

```
wget -c https://releases.linaro.org/components/toolchain/binaries/6.5-2018.12/arm-eabi/gcc-linaro-6.5.0-2018.12-x86_64_arm-eabi.tar.xz
tar xf gcc-linaro-6.5.0-2018.12-x86_64_arm-eabi.tar.xz
export CC=`pwd`/gcc-linaro-6.5.0-2018.12-x86_64_arm-eabi/bin/arm-eabi-
```

Test Cross Compiler:

user@localhost:~$

```
${CC}gcc --version
```

Test Output:

```
arm-eabi-gcc (Linaro GCC 6.5-2018.12) 6.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Bootloader: U-Boot

Das U-Boot – the Universal Boot Loader: http://www.denx.de/wiki/U-Boot
eewiki.net patch archive: https://github.com/eewiki/u-boot-patches
Download:

user@localhost:~$

```
git clone -b v2019.01 https://github.com/u-boot/u-boot --depth=1
cd u-boot/
```

Patches:

user@localhost:~/u-boot$

```
wget -c https://github.com/eewiki/u-boot-patches/raw/master/v2019.01/0001-mx23_olinuxino-uEnv.txt-bootz-n-fixes.patch

patch -p1 < 0001-mx23_olinuxino-uEnv.txt-bootz-n-fixes.patch
```

Configure and Build:

user@localhost:~/u-boot$

```
make ARCH=arm CROSS_COMPILE=${CC} distclean
make ARCH=arm CROSS_COMPILE=${CC} mx23_olinuxino_defconfig
make ARCH=arm CROSS_COMPILE=${CC} u-boot.sb
./tools/mxsboot sd u-boot.sb u-boot.sd
```

Linux Kernel

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

user@localhost:~$

```
git clone https://github.com/RobertCNelson/armv5_devel
cd armv5_devel/
```

For v4.9.x-imxv5 (Longterm 4.9.x):

user@localhost:~/armv5_devel$

```
git checkout origin/v4.9.x-imxv5 -b tmp
```

For v4.14.x-imxv5 (Longterm 4.14.x):

user@localhost:~/armv5_devel$

```
git checkout origin/v4.14.x-imxv5 -b tmp
```

For v4.19.x-imxv5 (Longterm 4.19.x):

user@localhost:~/armv5_devel$

```
git checkout origin/v4.19.x-imxv5 -b tmp
```

Build:

user@localhost:~/armv5_devel$

```
./build_kernel.sh
```

Root File System

Debian 10

User	Password
debian	temppwd
root	root

Download:

user@localhost:~$

```
wget -c https://rcn-ee.com/rootfs/eewiki/minfs/debian-10.4-minimal-armel-2020-05-10.tar.xz
```

Verify:

user@localhost:~$

```
sha256sum debian-10.4-minimal-armel-2020-05-10.tar.xz
```

sha256sum output:

```
2669f496afd6992688f620f30689ef385a8bde2b2a4cddc3582d79f8dad27d5f  debian-10.4-minimal-armel-2020-05-10.tar.xz
```

Extract:

user@localhost:~$

```
tar xf debian-10.4-minimal-armel-2020-05-10.tar.xz
```

Setup microSD card

We need to access the External Drive to be utilized by the target device. Run lsblk to help figure out what linux device has been reserved for your External Drive.

Example: for DISK=/dev/sdX

```
lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0 465.8G  0 disk
sda1   8:1    0   512M  0 part /boot/efi
sda2   8:2    0 465.3G  0 part /               <- Development Machine Root Partition
sdb      8:16   1   962M  0 disk                 <- microSD/USB Storage Device
sdb1   8:17   1   961M  0 part                 <- microSD/USB Storage Partition
```

Thus you would use:

```
export DISK=/dev/sdb
```

Example: for DISK=/dev/mmcblkX

```
lsblk
NAME      MAJ:MIN   RM   SIZE RO TYPE MOUNTPOINT
sda         8:0      0 465.8G  0 disk
sda1      8:1      0   512M  0 part /boot/efi
sda2      8:2      0 465.3G  0 part /               <- Development Machine Root Partition
mmcblk0     179:0    0   962M  0 disk                 <- microSD/USB Storage Device
mmcblk0p1 179:1    0   961M  0 part                 <- microSD/USB Storage Partition
```

Thus you would use:

```
export DISK=/dev/mmcblk0
```

Erase partition table/labels on microSD card:

```
sudo dd if=/dev/zero of=${DISK} bs=1M count=20
```

Create Partition Layout:
With util-linux v2.26, sfdisk was rewritten and is now based on libfdisk.

Check the version of sfdisk installed on your pc

```
sudo sfdisk --version
```

Example Output

```
sfdisk from util-linux 2.27.1
```

sfdisk >= 2.26.x

```
sudo sfdisk ${DISK} <<-__EOF__
1M,16M,0x53,-
17M,,,-
__EOF__
```

sfdisk <= 2.25.x

```
sudo sfdisk --unit M ${DISK} <<-__EOF__
1,16,0x53,-
17,,,-
__EOF__
```

Install Bootloader:

user@localhost:~$

```
for: DISK=/dev/mmcblk0
sudo dd if=./u-boot/u-boot.sd of=${DISK}p1

for: DISK=/dev/sdX
sudo dd if=./u-boot/u-boot.sd of=${DISK}1
```

Format Partition:

```
for: DISK=/dev/mmcblkX
sudo mkfs.ext4 -L rootfs ${DISK}p2

for: DISK=/dev/sdX
sudo mkfs.ext4 -L rootfs ${DISK}2
```

Mount Partition:
On most systems these partitions may be auto-mounted...

```
sudo mkdir -p /media/rootfs/

for: DISK=/dev/mmcblkX
sudo mount ${DISK}p2 /media/rootfs/

for: DISK=/dev/sdX
sudo mount ${DISK}2 /media/rootfs/
```

Install Kernel and Root File System

To help new users, since the kernel version can change on a daily basis. The kernel building scripts listed on this page will now give you a hint of what kernel version was built.

```
-----------------------------
Script Complete
eewiki.net: [user@localhost:~$ export kernel_version=5.X.Y-Z]
-----------------------------
```

Copy and paste that "export kernel_version=5.X.Y-Z" exactly as shown in your own build/desktop environment and hit enter to create an environment variable to be used later.

```
export kernel_version=5.X.Y-Z
```

Copy Root File System

user@localhost:~$

```
sudo tar xfvp ./debian-*-*-armel-*/armel-rootfs-*.tar -C /media/rootfs/
sync
sudo chown root:root /media/rootfs/
sudo chmod 755 /media/rootfs/
```

Set uname_r in /boot/uEnv.txt

user@localhost:~$

```
sudo sh -c "echo 'uname_r=${kernel_version}' >> /media/rootfs/boot/uEnv.txt"
```

Copy Kernel Image

Kernel Image:

user@localhost:~$

```
sudo cp -v ./armv5_devel/deploy/${kernel_version}.zImage /media/rootfs/boot/vmlinuz-${kernel_version}
```

Copy Kernel Device Tree Binaries

user@localhost:~$

```
sudo mkdir -p /media/rootfs/boot/dtbs/${kernel_version}/
sudo tar xfv ./armv5_devel/deploy/${kernel_version}-dtbs.tar.gz -C /media/rootfs/boot/dtbs/${kernel_version}/
```

Copy Kernel Modules

user@localhost:~$

```
sudo tar xfv ./armv5_devel/deploy/${kernel_version}-modules.tar.gz -C /media/rootfs/
```

File Systems Table (/etc/fstab)

user@localhost:~/$

```
sudo sh -c "echo '/dev/mmcblk0p2  /  auto  errors=remount-ro  0  1' >> /media/rootfs/etc/fstab"
```

Remove microSD/SD card

```
sync
sudo umount /media/rootfs
```

Comments

Any questions or comments please go to our TechForum: TechForum