STEP TO STEP FOR BUILD DEBIAN ON iMX233 OLINUXINO USING VIRTUAL MACHINE

# Availability

Boards:  
[IMX233-OLINUXINO-MICRO](https://www.digikey.com/product-search/en?keywords=1188-1089-ND) at Digi-Key

Accessories:  
[USB-SERIAL-CABLE-F](https://www.digikey.com/product-search/en?keywords=1188-1018-ND) at Digi-Key

# Kit Notes

iMX233-OLinuXino-Micro:

From: [PDF Manual](https://www.olimex.com/Products/OLinuXino/iMX233/iMX233-OLinuXino-MICRO/resources/iMX233-OLINUXINO-MICRO.pdf)

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](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 release of Debian, Fedora or Ubuntu; without OS Virtualization Software.
* ARM Cross Compiler – Linaro: [https://www.linaro.org](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/cgit/linux/kernel/git/torvalds/linux.git>
* ARM based rootfs
  + Debian: [https://www.debian.org](https://www.debian.org/)

# ARM Cross Compiler: GCC

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

**~/**

|  |
| --- |
| wget -c https://releases.linaro.org/components/toolchain/binaries/6.4-2018.05/arm-eabi/gcc-linaro-6.4.1-2018.05-x86\_64\_arm-eabi.tar.xz  tar xf gcc-linaro-6.4.1-2018.05-x86\_64\_arm-eabi.tar.xz  export CC=`pwd`/gcc-linaro-6.4.1-2018.05-x86\_64\_arm-eabi/bin/arm-eabi- |

Test Cross Compiler:

**~/**

|  |
| --- |
| ${CC}gcc --version  arm-eabi-gcc (Linaro GCC 6.4-2018.05) 6.4.1 20180425 [linaro-6.4-2018.05 revision 7b15d0869c096fe39603ad63dc19ab7cf035eb70]  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:

**~/**

|  |
| --- |
| git clone https://github.com/u-boot/u-boot  cd u-boot/  git checkout v2019.01 -b tmp |

Patches:

**~/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:

**~/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:

**~/**

|  |
| --- |
| git clone https://github.com/RobertCNelson/armv5\_devel  cd armv5\_devel/ |

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

**~/armv5\_devel/**

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

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

**~/armv5\_devel/**

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

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

**~/armv5\_devel/**

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

Build:

**~/armv5\_devel/**

|  |
| --- |
| ./build\_kernel.sh |

# Root File System

#### **Debian 9**

|  |  |
| --- | --- |
| **User** | **Password** |
| debian | temppwd |
| root | root |

Download:

**~/**

|  |
| --- |
| wget -c https://rcn-ee.com/rootfs/eewiki/minfs/debian-9.6-minimal-armel-2018-11-26.tar.xz |

Verify:

**~/**

|  |
| --- |
| sha256sum debian-9.6-minimal-armel-2018-11-26.tar.xz  7f93f169764664d60588c904e23acb5ac81b9e14f8023bb5bb2c548f9b33fe39  debian-9.6-minimal-armel-2018-11-26.tar.xz |

Extract:

**~/**

|  |
| --- |
| tar xf debian-9.6-minimal-armel-2018-11-26.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.**

|  |
| --- |
| sudo sfdisk --version  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:

**~/**

|  |
| --- |
| **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 will 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=4.X.Y-Z]  ----------------------------- |

Copy and paste that "export kernel\_version=4.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=4.X.Y-Z |

### **Copy Root File System**

**~/**

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

### **Emulate SO on VM**

|  |
| --- |
| add in /etc/apt/apt.conf.d/70debconf  APT::Default-Release "stable";    add in /etc/apt/sources.list  deb http://ftp.debian.org/debian/ testing main contrib  update the list of the packages  $ sudo apt-get update  install qemu-user-static  $ sudo apt-get install -t testing qemu-user-static  install binfmt-support  $ sudo apt-get install binfmt-support  $ sudo chroot rootfs/ //run SO on VM |
|  |

### **Set uname\_r in /boot/uEnv.txt**

**~/**

|  |
| --- |
| sudo sh -c "echo 'uname\_r=${kernel\_version}' >> /media/rootfs/boot/uEnv.txt" |

### **Copy Kernel Image**

Kernel Image:

**~/**

|  |
| --- |
| sudo cp -v ./armv5\_devel/deploy/${kernel\_version}.zImage /media/rootfs/boot/vmlinuz-${kernel\_version} |

### **Copy Kernel Device Tree Binaries**

**~/**

|  |
| --- |
| 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**

**~/**

|  |
| --- |
| sudo tar xfv ./armv5\_devel/deploy/${kernel\_version}-modules.tar.gz -C /media/rootfs/ |

### **File Systems Table (/etc/fstab)**

|  |
| --- |
| 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 |