Wandboard

This is a page about the NXP based i.MX6; Wandboard.

Availability

Boards:
Wandboard SOLO at Digi-Key
Wandboard DUAL at Digi-Key
Wandboard QUAD at Digi-Key
Wandboard SOLO (WB-IMX6S) at Digi-Key
Wandboard DUAL (WB-IMX6U-BW) at Digi-Key
Wandboard QUAD (WB-IMX6Q-BW) at Digi-Key
Wandboard QUADPLUS (WB-IMX6QP-BW) at Digi-Key
Accessories:
Wandboard Antenna Kit at Digi-Key
Wandboard Enclosure at Digi-Key
5V 2A+ DC Power Supply (2.1mm ID x 5.5mm OD positive center plug) at Digi-Key

Vendor Documentation

- WANDBOARD.ORG: https://www.wandboard.org/

Basic Requirements

- Running a recent release of Debian, Fedora or Ubuntu; without OS Virtualization Software.
- 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/cgit/linux/kernel/git/torvalds/linux.git
- ARM based rootfs
  - Debian: https://www.debian.org
  - Ubuntu: https://www.ubuntu.com

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-linux-gnueabihf/gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz
tar xf gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz
export CC=`pwd`/gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
```

Test Cross Compiler:

~/

```
${CC}gcc --version
arm-linux-gnueabihf-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-wandboard-uEnv.txt-bootz-n-fixes.patch

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

Configure and Build:

~/u-boot

```
make ARCH=arm CROSS_COMPILE=${CC} distclean
make ARCH=arm CROSS_COMPILE=${CC} wandboard_defconfig
make ARCH=arm CROSS_COMPILE=${CC}
```

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/armv7-multiplatform
cd armv7-multiplatform/
```

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

~/armv7-multiplatform/

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

For v4.14.x-rt (Longterm 4.14.x + Real-Time Linux):

~/armv7-multiplatform/

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

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

~/armv7-multiplatform/

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

For v4.19.x-rt (Longterm 4.19.x + Real-Time Linux):

~/armv7-multiplatform/

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

Build:

~/armv7-multiplatform/

```
./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-armhf-2018-11-26.tar.xz
```

Verify:

~/

```
sha256sum debian-9.6-minimal-armhf-2018-11-26.tar.xz
c9d9b2623131829eeeefa153274219d89887a1984b8dde72cfbfe29f29c7a10d  debian-9.6-minimal-armhf-2018-11-26.tar.xz
```

Extract:

~/

```
tar xf debian-9.6-minimal-armhf-2018-11-26.tar.xz
```

Ubuntu 18.04 LTS

User	Password
ubuntu	temppwd

Download:

~/

```
wget -c https://rcn-ee.com/rootfs/eewiki/minfs/ubuntu-18.04.1-minimal-armhf-2018-11-26.tar.xz
```

Verify:

~/

```
sha256sum ubuntu-18.04.1-minimal-armhf-2018-11-26.tar.xz
d8190bc858bab9db83bec03707b2636dc23646fca2a0420fb49adbedf7d492c5  ubuntu-18.04.1-minimal-armhf-2018-11-26.tar.xz
```

Extract:

~/

```
tar xf ubuntu-18.04.1-minimal-armhf-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=10
```

Install Bootloader:

~/

```
sudo dd if=./u-boot/SPL of=${DISK} seek=1 bs=1k
sudo dd if=./u-boot/u-boot.img of=${DISK} seek=69 bs=1k
```

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,,L,*
__EOF__
```

sfdisk <= 2.25.x

```
sudo sfdisk --unit M ${DISK} <<-__EOF__
1,,L,*
__EOF__
```

Format Partition:

```
for: DISK=/dev/mmcblkX
sudo mkfs.ext4 -L rootfs ${DISK}p1

for: DISK=/dev/sdX
sudo mkfs.ext4 -L rootfs ${DISK}1
```

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

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

for: DISK=/dev/mmcblkX
sudo mount ${DISK}p1 /media/rootfs/

for: DISK=/dev/sdX
sudo mount ${DISK}1 /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 ./*-*-*-armhf-*/armhf-rootfs-*.tar -C /media/rootfs/
sync
sudo chown root:root /media/rootfs/
sudo chmod 755 /media/rootfs/
```

Set uname_r in /boot/uEnv.txt

~/

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

Video

See other hints: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/fb/modedb.txt

~/

```
sudo sh -c "echo 'cmdline=video=HDMI-A-1:1024x768@60e' >> /media/rootfs
/boot/uEnv.txt"
```

Copy Kernel Image

Kernel Image:

~/

```
sudo cp -v ./armv7-multiplatform/deploy/${kernel_version}.zImage /media
/rootfs/boot/vmlinuz-${kernel_version}
```

Copy Kernel Device Tree Binaries

~/

```
sudo mkdir -p /media/rootfs/boot/dtbs/${kernel_version}/
sudo tar xfv ./armv7-multiplatform/deploy/${kernel_version}-dtbs.tar.gz -C
/media/rootfs/boot/dtbs/${kernel_version}/
```

Copy Kernel Modules

~/

```
sudo tar xfv ./armv7-multiplatform/deploy/${kernel_version}-modules.tar.gz
-C /media/rootfs/
```

File Systems Table (/etc/fstab)

```
sudo sh -c "echo '/dev/mmcblk2p1  /  auto  errors=remount-ro  0  1' >>
/media/rootfs/etc/fstab"
```

WiFi

Dual/Quad only (Solo has no WiFi module installed)
see: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/drivers/net/wireless/brcm80211/brcmfmac/dhd_sdio.c?id=refs/tags/v3.11-rc3#n317

~/

```
wget -c https://git.kernel.org/cgit/linux/kernel/git/firmware/linux-
firmware.git/plain/brcm/brcmfmac4329-sdio.bin
wget -c https://git.kernel.org/cgit/linux/kernel/git/firmware/linux-
firmware.git/plain/brcm/brcmfmac4330-sdio.bin

wget -c https://rcn-ee.com/repos/git/meta-fsl-arm-extra/recipes-bsp
/broadcom-nvram-config/files/wandboard/brcmfmac4329-sdio.txt
wget -c https://rcn-ee.com/repos/git/meta-fsl-arm-extra/recipes-bsp
/broadcom-nvram-config/files/wandboard/brcmfmac4330-sdio.txt

sudo mkdir -p /media/rootfs/lib/firmware/brcm/

sudo cp -v ./brcmfmac43*-sdio.bin /media/rootfs/lib/firmware/brcm/
sudo cp -v ./brcmfmac43*-sdio.txt /media/rootfs/lib/firmware/brcm/
```

Status:

```
debian@arm:~$ dmesg | grep brcm
[    4.798229] brcmfmac: F1 signature read @0x18000000=0x9934329
[    4.799328] brcmfmac: brcmf_sdio_chip_drivestrengthinit: No SDIO Drive
strength init done for chip 4329 rev 3 pmurev 6
[    5.058286] brcmfmac: brcmf_c_preinit_dcmds: Firmware version = wl0:
Sep  2 2011 14:48:19 version 4.220.48
[    5.082280] brcmfmac: brcmf_fil_cmd_data: Failed err=-23
[    5.087654] brcmfmac: brcmf_fws_init: failed to set bdcv2 tlv signaling
debian@arm:~$ /sbin/ifconfig -a
wlan0     Link encap:Ethernet  HWaddr 40:2c:f4:ae:17:91
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Remove microSD/SD card

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

2D/3D Video Acceleration via Etnaviv Project

Vivante GC320/GC880 (quad GC320/GC355/GC2000) 2D/3D Acceleration via Etnaviv
This sections assumes you have already installed your favorite xorg based window manager, such as lxde, lxqt, xfce, kde, gnome, etc... These are packages that need to be installed on top of your selected windows manager and an xorg.conf needed to correctly setup the video interface.
Verify your kernel has etnaviv support:

SOLO/DUAL

```
debian@arm:~$ dmesg | grep etnaviv
[    4.047928] etnaviv gpu-subsystem: bound 134000.gpu (ops gpu_ops)
[    4.047955] etnaviv gpu-subsystem: bound 130000.gpu (ops gpu_ops)
[    4.047968] etnaviv-gpu 134000.gpu: model: GC320, revision: 5007
[    4.090180] etnaviv-gpu 130000.gpu: model: GC880, revision: 5106
```

QUAD

```
debian@arm:~$ dmesg | grep etnaviv
[    4.025841] etnaviv gpu-subsystem: bound 134000.gpu (ops gpu_ops)
[    4.025868] etnaviv gpu-subsystem: bound 130000.gpu (ops gpu_ops)
[    4.025887] etnaviv gpu-subsystem: bound 2204000.gpu (ops gpu_ops)
[    4.025899] etnaviv-gpu 134000.gpu: model: GC320, revision: 5007
[    4.070139] etnaviv-gpu 130000.gpu: model: GC2000, revision: 5108
[    4.120479] etnaviv-gpu 2204000.gpu: model: GC355, revision: 1215
[    4.120492] etnaviv-gpu 2204000.gpu: Ignoring GPU with VG and FE2.0
```

edid parsing still seems hit and miss on the quad board, override default 1024x768 resolution via the video variable in uEnv.txt.
See other hints: https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/fb/modedb.txt

```
cmdline=video=HDMI-A-1:1024x768@60e
```

1080p

```
cmdline=video=HDMI-A-1:1920x1080@60e
```

```
sudo apt-get update
sudo apt-get install xserver-xorg-video-armada-etnaviv
```

/etc/X11/xorg.conf

```
Section "Monitor"
        Identifier      "Builtin Default Monitor"
EndSection
Section "Device"
        Identifier      "Builtin Default fbdev Device 0"
        Driver          "armada"
EndSection
Section "Screen"
        Identifier      "Builtin Default fbdev Screen 0"
        Device          "Builtin Default fbdev Device 0"
        Monitor         "Builtin Default Monitor"
EndSection
Section "ServerLayout"
        Identifier      "Builtin Default Layout"
        Screen          "Builtin Default fbdev Screen 0"
EndSection
```

Comments

Comments, feedback, and questions can be sent to: eewiki@digikey.com
Please use the Digi-Key's TechForum: TechForum