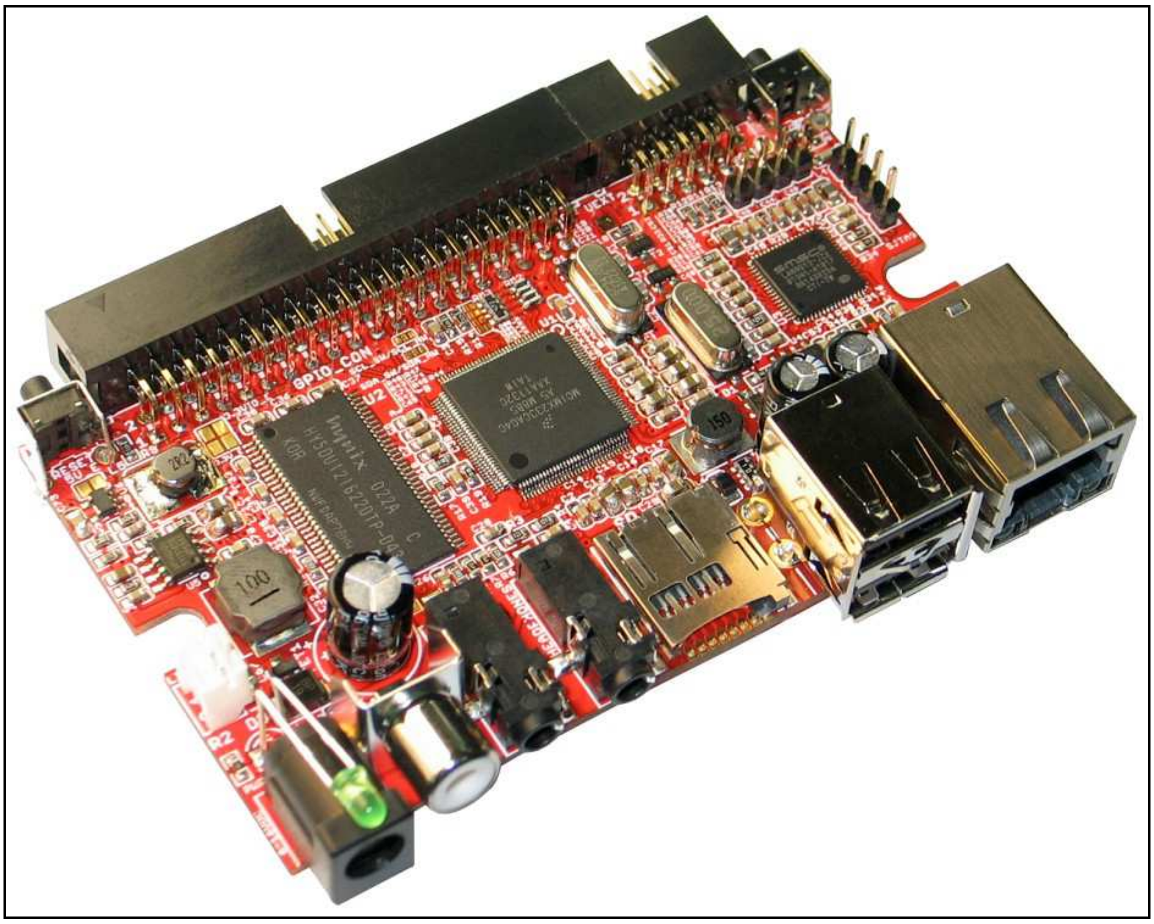
Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



OLinuXino-MAXI Open-source single-board Linux computer USER'S MANUAL

Revision L, March 2013
Designed by OLIMEX Ltd, 2012

All boards produced by Olimex LTD are ROHS compliant

DISCLAIMER

It is possible that the pictures in this manual differ from the latest revision of the board.

Table of Contents

CHAPTER 1: OVERVIEW

1. Introduction to the chapter

Thank you for choosing the OLinuXino single board computer from Olimex! This document provides a user's guide for the Olimex OLinuXino board. As an overview, this chapter gives the scope of this document and lists the board's features. The differences between the members of the OLinuXino family are mentioned. The document's organization is then detailed.

The OLinuXino development board enables code development of applications running on the microcontroller i.MX233, manufactured by FreeScale Semiconductor.

OLinuXino is an open-source, open-hardware project and all documentation is available to the customer.

1.1 Features

- iMX233 ARM926J processor at 454Mhz
- 64 MB RAM
- SD-card connector for booting the Linux image
- TV PAL/NTSC video output
- 3 USB High Speed Host
- optional WIFI RTL8188CU module
- Stereo Audio Input
- Stereo Headphones Audio Output
- two Buttons
- UEXT connector for connection of different peripherial modules
- 40 pin GPIO for connection of other hardware
- Board is in shape for fit inside Pactec JM42 plastic box
  http://www.pactecenclosures.com/pdfs/drw_JM-42.pdf
- Power supply input 6-16VDC
- PCB dimensions: 3.70" x 2.15" (94.0mm x 54.6mm)
- Nominal dimensions: 3.70" x 2.65" (94.0mm x 67.3mm)

1.2 The OLinuXino family

Table of comparison			
	OLinuXino-MICRO	OLinuXino-MINI	OLinuXino-MAXI
Processor	iMX233 @ 454Mhz	iMX233 @ 454Mhz	iMX233 @ 454Mhz
Ram [MB]	64	64	64
# USB hosts	1	3	2
100/150 Mbit Ethernet*	No/WIFI option**	No/WIFI option***	Yes/WIFI option**
GPIO connector	60pins	40pins	40pins
# Buttons	3	2	2
Reset button	Yes	Yes	Yes
DC power supply	5V	6V-16V	6V-16V
Dimensions	3.40'' x 1.70''	3.70'' x 2.65''	3.70'' x 2.65''
Breadboarding	Yes	No	No
Audio IN connector	No	Yes	Yes
Audio OUT connector	No	Yes	Yes
UEXT connector	No	Yes	Yes

* 100Mbit Ethernet for the wired network of OLinuXino-MAXI. 150Mbit for the WIFI following 811.02n standard.

** All three boards have the option to work with MOD-WIFI_RTL8188, which is USB WIFI modem with RTL8188CU chip and can be purchased separately. MOD-WIFI_RTL8188 can be connected to any of the OLinuXino boards via the USB.

*** OLinuXino-MINI has additional option of having RTL8188CU hardware mounted! If you wish RTL8188CU embedded in the device you should purchase OLinuXino-MINI-WIFI. Choosing the embedded WIFI option will leave your USB-HOSTs available for use.

1.2 Target market and purpose of the board

The boards from the OLinuXino family are ready to use, easy to setup and are suitable for embedded programming enthusiasts, Linux hobbyists, gadget fans and also professionals (since its low cost makes it very good solution for application orientated embedded systems). The main usage of the board is software embedded development without the urge of understanding perfectly the hardware.

The strong points of the boards are the processor speed, the mobility of the board and the low price.

Customers have full access to the technical documentation of the board. The software is released under General Purpose License and the board is considered open-hardware.

1.3 Organization

Each section in this document covers a separate topic, organized as follow:

- Chapter 1 is an overview of the board usage and features
- Chapter 2 provides a guide for quickly setting up the board and software notes
- Chapter 3 contains the general board diagram and layout
- Chapter 4 describes the component that is the heart of the board: the iMX233 microcontroller
- Chapter 5 is an explanation of the control circuitry associated with the microcontroller to reset. Also shows the clocks on the board
- Chapter 6 covers the connector pinout, peripherals and jumper description
- Chapter 7 shows the memory map
- Chapter 8 provides the schematics
- Chapter 9 contains the revision history, useful links and support information

CHAPTER 2: SETTING UP THE OLINUXINO BOARD

2. Introduction to the chapter

This section helps you set up the OLinuXino development board for the first time. Please consider first the electrostatic warning to avoid damaging the board, then discover the hardware and software required to operate the board.

The procedure to power up the board is given, and a description of the default board behavior is detailed.

2.1 Electrostatic warning

OLinuXino is shipped in a protective anti-static package. The board must not be exposed to high electrostatic potentials. A grounding strap or similar protective device should be worn when handling the board. Avoid touching the component pins or any other metallic element.

2.3 Requirements

In order to set up the OLinuXino optimally, the following items are required:

- 6V to 16V source of power with 1A maximum amperage.
- SJTAG interface programmer
- USB keyboard
- Monitor with composite interface or Personal Computer + USB-SERIAL-CABLE
- SD card with Linux image

Note that the board arrives without SD card or Linux image. You can purchase a card with Linux separately. It is recommended that the user has basic Linux experience.

Some of the suggested items can be purchased by Olimex, for instance:

iMX233-OLinuXino-SD - SD card with the Linux image

USB-SERIAL-CABLE-F - USB serial console cable female (check “6.1.1 UART Debug” for info how to connect it to the board)

SY0612E - power supply adapter 12V/0.5A for iMX233-OLinuXino-Maxi

2.4 Powering the board

The board is powered either via the PWR jack or via a battery. It should be supplied from a 6V to 16V source with maximum current of 1A from the power jack.

All measures below are taken at 10V.

If measuring the current consumption it should be around 0.06A before initializing all the peripherals. The consumption raises to 0.12A without LAN and USB hosts initialized and Linux running. The consumption goes up 0.15A with the Linux running when LAN and both USB hosts initialized.

If you have a standard USB flash drive attached to a USB host, Linux and LAN running the typical consumption is around 0.20A.

When powered by the typical 3.7V battery the LAN and USB-hosts will be powered-off if you use the external 3.3V DC-DC (they will stil work if using the internal in the processor DC-DC). The consumption from the battery when Linux is running is around 0.75A.

IMPORTANT! We discovered a situation which might leave some of the SD cards (iMX233-OLinuXino-SD) in unrecoverable state when powering OLinuXino-MICRO. The problem might occur if two specific conditions are met simultaneously:

1)Plugged iMX233-OLinuXino-SD micro SD card with holographic sticker on its back side (some of the cards we have distributed are from a brand that places holographic sticker on their backs, the other half lack such a sticker)
2)Plugged USB-SERIAL-CABLE-F at the moment when powering the board

If you happen to have received SD card with holographic sticker on its back side and you use it with OLinuXino-MICRO and you plug USB-SERIAL-CABLE and then you power the board there is a chance of malfunction of the SD card.

There are two possible workarounds to protect the SD card. The first one is simpler and the second one requires some soldering experience.

Workaround 1: First insert the iMX233-OLinuXino-SD card and then power the board (and if powering the board from a battery also press the PWR button). Wait 4-5 seconds and then connect the USB-SERIAL-CABLE-F. After the initial power-up it is safe to use the reset button.

Workaround 2: You will need a Shottky diode. The Shottky should be soldered on the USB-SERIAL-CABLE-F TX line/wire (RED cable) with anode towards the board.

When you power the board by battery you have to press the PWR_BUT to start the board.
If you start Linux and it is already running no matter which powering method you use (PWR_JACK or BAT) pressing the PWR_BUT will put the Linux in power-save mode.

When you power the board by battery you have to press the PWR_BUT to start the board.
If you start Linux and it is already running no matter which powering method you use (PWR_JACK or BAT) pressing the PWR_BUT will put the Linux in power-save mode.

For the European customers we sell a power supply adapter 12V/0.5A - SY0612E.

2.5 Prebuilt software

Note that the boards arrive without Linux or SD card. The Linux image can be purchased separately on a SD card or you can built and adjust it yourself.

When we program the boards we change the default position of the following HW_OCOTP_ROM0 fuses of the processor:

SD_MBR_BOOT(3) - Blown
SD_POWER_GATE_GPIO(21:20) – 10-PWM3

For burning the fuse position we use the BitBurner software. This operation is discussed in details before. Proceed with great caution when burning fuses since it is irreversible operation.

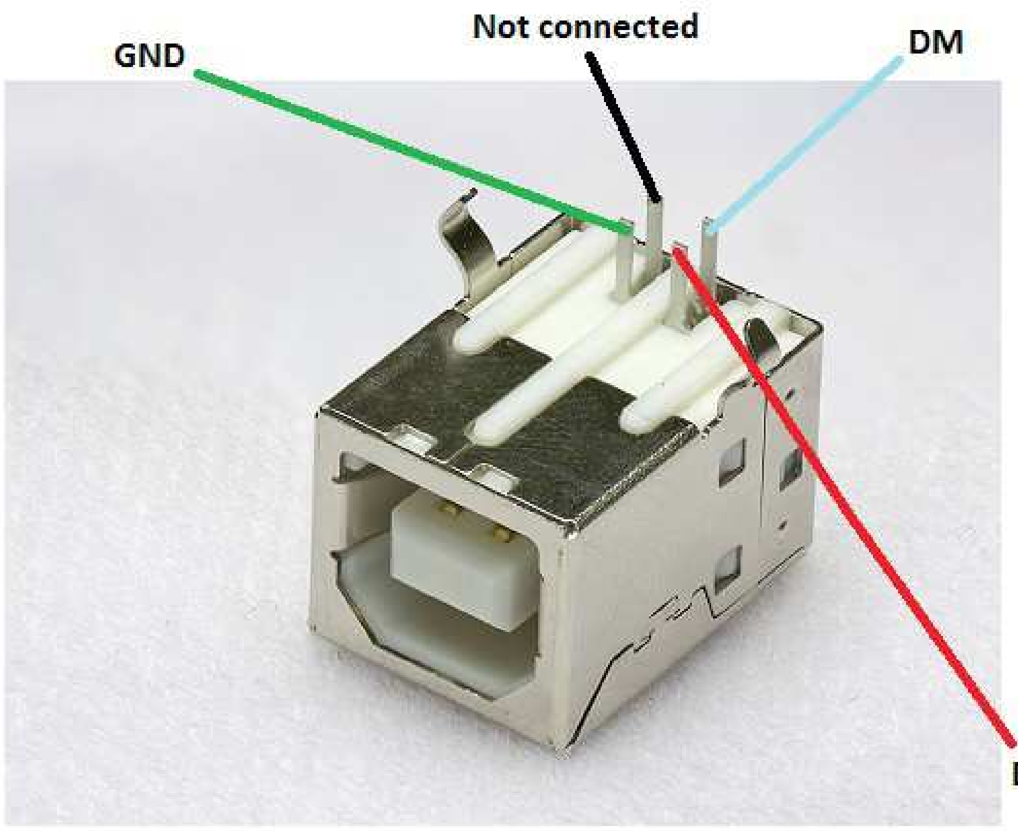
The first batches of the board and the SD-card used the Debian Linux image. After that we switched over to ArchLinux for the ease of the package manager. Instructions how to build the ArchLinux can be found at the gitHub address of OLinuXino.

2.6 Using BitBurner

IMPORTANT! MODIFYING THE FUSES IS IRREVERSIBLE PROCESS! BURNING THE WRONG FUSES MIGHT DAMAGE OLINUXINO IRREVERSIBLY! BURNING WRONG FUSES MIGHT CAUSE BOOT PROBLEMS!

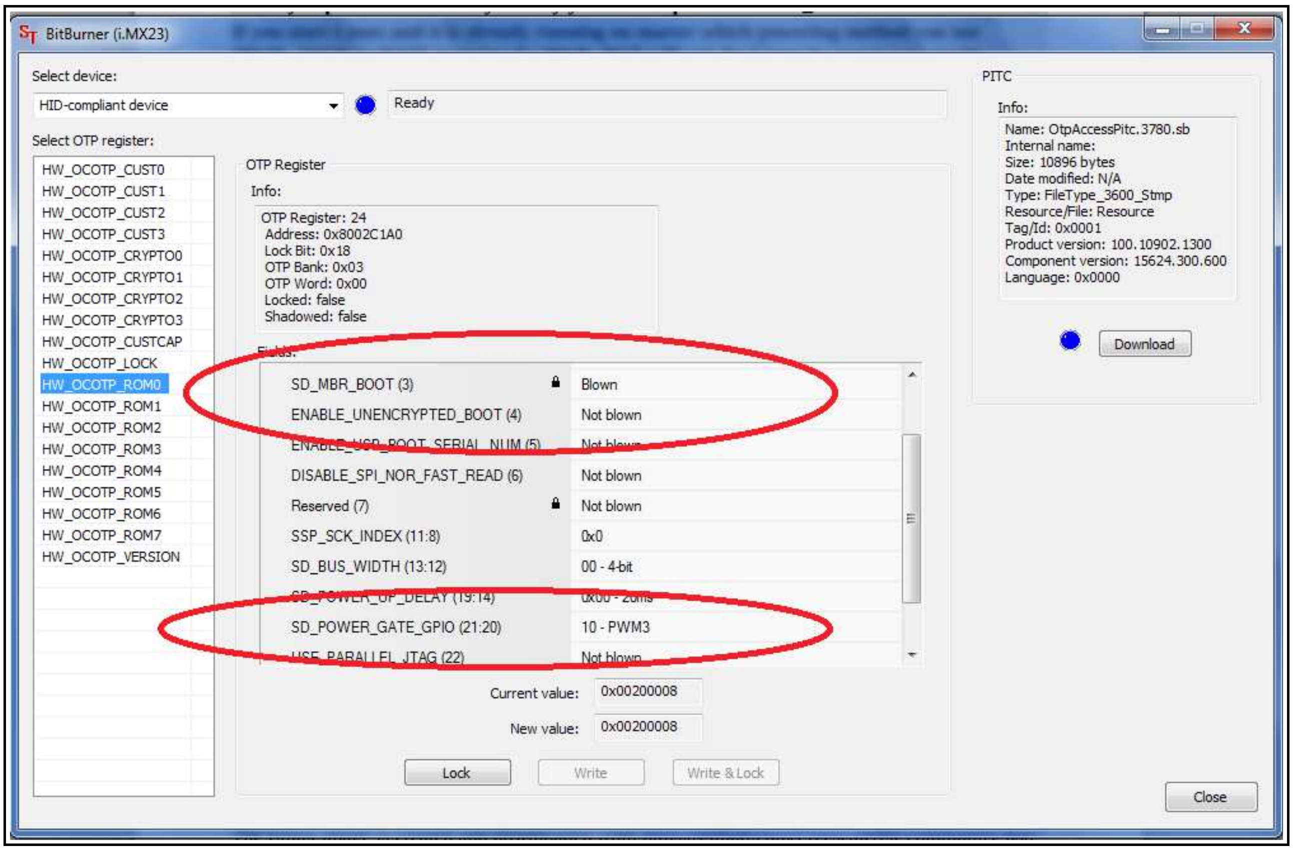
BURN FUSES AT OWN RISK!

The bit burning is done via the USB of the computer connected to the OLINUXINO board and the BitBurner software. To be able to burn the fuses you will need to make a custom cable that connects a USB with the 3 pin holes found at the bottom of the board named “GND”, “DP”, “DM” (check the picture below for a better view how the three wires must be connected.



After soldering the three wires you can place a marker on each of them or use colored wires to be able to distinguish them. You can also use some gel to keep them tight on the USB connector. On the opposite side of the cables you might place 50mil (1.27mm) male connector following the order of the signals. Please also restrain from using wires longer than 20 cm since that might make the connection unreliable.

Download BitBurner from https://www.olimex.com/dev/OLINUXINO/iMX233-OLINUXINO/BitBurner.v1.0.4.6.zip. Extract it and start the .exe. If you connect everything you should see and choose HID-compilant device from the "Select device" drop-down menu.


BitBurner (i.MX23)

Select device: HID-compliant device ● Ready

Select OTP register:

- HW_OCOTP_CUST0
- HW_OCOTP_CUST1
- HW_OCOTP_CUST2
- HW_OCOTP_CUST3
- HW_OCOTP_CRYPTO0
- HW_OCOTP_CRYPTO1
- HW_OCOTP_CRYPTO2
- HW_OCOTP_CRYPTO3
- HW_OCOTP_CUSTCAP
- HW_OCOTP_LOCK
- HW_OCOTP_ROM0
- HW_OCOTP_ROM1
- HW_OCOTP_ROM2
- HW_OCOTP_ROM3
- HW_OCOTP_ROM4
- HW_OCOTP_ROM5
- HW_OCOTP_ROM6
- HW_OCOTP_ROM7
- HW_OCOTP_VERSION

OTP Register

Info:

OTP Register: 24
Address: 0x8002C1A0
Lock Bit: 0x18
OTP Bank: 0x03
OTP Word: 0x00
Locked: false
Shadowed: false

Fields:

SD_MBR_BOOT (3)	🔒	Blown
ENABLE_UNENCRYPTED_BOOT (4)		Not blown
ENABLE_USB_BOOT_SERIAL_NUM (5)		Not blown
DISABLE_SPI_NOR_FAST_READ (6)		Not blown
Reserved (7)	🔒	Not blown
SSP_SCK_INDEX (11:8)		0x0
SD_BUS_WIDTH (13:12)		00 - 4-bit
SD_POWER_UP_DELAY (19:14)		0x00 - 20ms
SD_POWER_GATE_GPIO (21:20)		10 - PWM3
USE_PARALLEL_JTAG (22)		Not blown

Current value: 0x00200008

New value: 0x00200008

PITC

Info:

Name: OtpAccessPitc.3780.sb
Internal name:
Size: 10896 bytes
Date modified: N/A
Type: FileType_3600_Stmp
Resource/File: Resource
Tag/Id: 0x0001
Product version: 100.10902.1300
Component version: 15624.300.600
Language: 0x0000

2.7 Building the Linux image

Note that building the Linux image from scratch is a time-consuming task. Even with powerful machine and fast internet connection it might take few hours compiling. Some Linux distributions might lack the tools required to compile/build/execute scripts/download from repository – how to get those is not discussed below.

The Linux image is created and downloaded from https://github.com/Freescale/fsl-community-bsp-platform. For the test here we used Debian 6.0 with GNOME visual libraries. The steps we did:

1) From the terminal created folder “bin” in home folder:

```
user@dist$: mkdir bin
user@dist$: cd bin
```

Add bin directory to PATH in order to do the next steps easier. Else navigate to the right folders.

2) Installed `repo` utility needed for the bitbake file fetching from the repository:

```
user@dist$: curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo
```

```
user@dist$: chmod a+x ~/bin/repo
```

3) Created directory for the project and download the BSP source from the git repository:

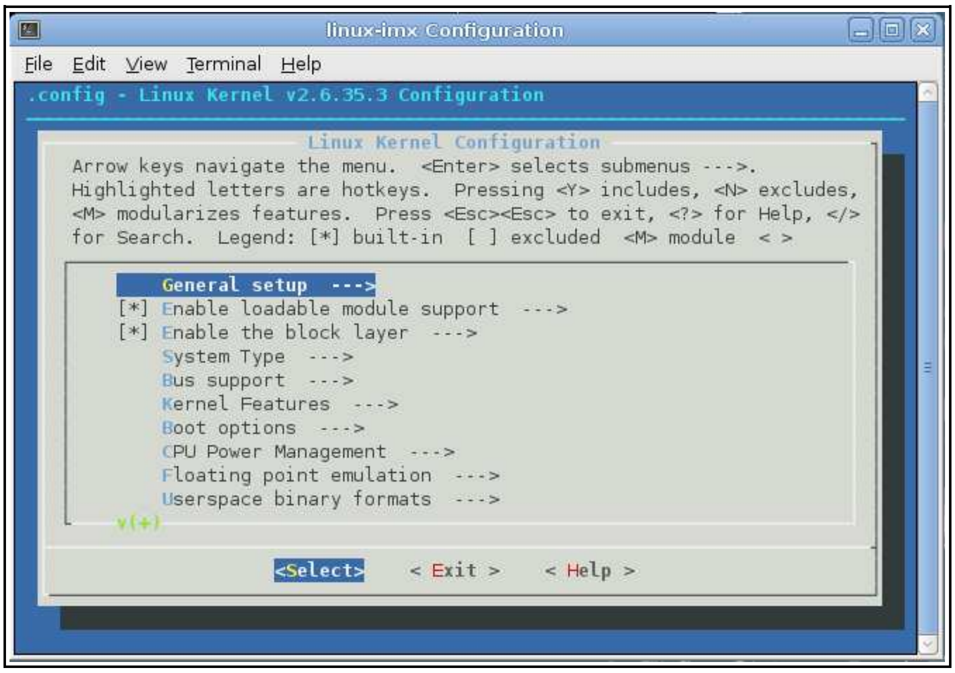
```
user@dist$: mkdir fsl-community-bsp
user@dist$: cd fsl-community-bsp

~/fsl-community-bsp$: repo init -u https://github.com/Freescale/fsl-community-bsp-platform -b
denzil
~/fsl-community-bsp$: repo sync
```

4) You can change the settings for the build if you want at fsl-community-bsp/build/conf/local.conf. I changed the "machine" name to "imx233-olinuxino-maxi".
For Linux kernel configurations and settings you can do (of course you can use also the default settings):

```
~/fsl-community-bsp$:. ./setup-environment build
~/fsl-community-bsp/build$:bitbake linux-imx -c menuconfig
```

Check the image below:



```
.config - Linux Kernel v2.6.35.3 Configuration
                    Linux Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus --->.
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes,
  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </>
  for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >

        General setup  --->
    [*] Enable loadable module support  --->
    [*] Enable the block layer  --->
        System Type  --->
        Bus support  --->
        Kernel Features  --->
        Boot options  --->
        CPU Power Management  --->
        Floating point emulation  --->
        Userspace binary formats  --->
    v(+)

              <Select>    < Exit >    < Help >
```

5) Now to start building the image:

```
~fsl-community-bsp$:. ./setup-environment build

~fsl-community-bsp/build $: bitbake core-image-minimal
```

Note: on different Linux distributions you might have different tools installed and you will probably need to install dependencies needed for the compile/build scripts. Here are some (but not all) of the mandatory ones: G++; diffstat; texi2html; chrpath; gawk; texinfo; some git client.

To ensure you have the latest version supported with all the updates visit https://github.com/OLIMEX/OLINUXINO and https://github.com/Freescale/fsl-community-bsp-platform.

2.8 How to blink the LED

In this sub-chapter you will find a way to achieve the most basic task in electronics – the "Hello World" of electronics - blinking the LED.

First we set the pin responsible for the LED as an output and we can set its value manually to high or low position – make it blink manually. The LED mounted on the board uses GPIO65. You can use external diode instead of the one mounted - you have to look at the table "The Linux implementation of pins" in the hardware section to get the correct linux name for the GPIO pin.

```
echo out > /sys/class/gpio/gpio65/direction
echo 1 > /sys/class/gpio/gpio65/value
```

If you want to set the blink off you should change the value on the second line to:

```
echo 0 > /sys/class/gpio/gpio65/value
```

To show the info for all GPIOs:

```
ls /sys/class/gpio
```

To make it turn on – turn off automatically (e.g. blink) we use the text redactor VI to write the Linux script:

```
echo out > /sys/class/gpio/gpio65/direction
while true
do
echo 1 > /sys/class/gpio/gpio65/value
sleep 1
echo 0 > /sys/class/gpio/gpio65/value
sleep 1
done
```

We save it as as "gpio" and we make it executable with

```
chmod +x gpio
```

then we execut the script with:

```
./gpio
```

The LED should start blinking with 0.5Hz.

2.9 How setup the I2C, SPI, UART

There are number of examples with our extension module board to achieve those connections on the UEXT. The examples might be used as an example for I2C, SPI or UART communication. You can find them at our GitHub page:
https://github.com/OLIMEX/OLINUXINO/tree/master/SOFTWARE/iMX233

NOTE that to use software I2C you have to set the SMD jumpers to the proper side. By default the board uses hardware I2C.

2.10 How to setup Arch-Linux distribution

This is a step-by-step guide for building the Archlinux image for Olimex OLinuXino-MAXI. The second and the third part of the document explain how to setup the internet connection on the board and how to use the GCC compiler to compile „HelloWorld“ example on the OLinuXino.

Two important preparations:

1)You have to download the latest software package from the GitHub of iMX233-OLinuXino, down in the code you will meet “xxxxxxx” which you will have to replace with the correct locations in the package.

2)When trying I2C examples I2C jumpers has to be configured for hardware I2C mode.

I. How to create SD-card with Archlinux image

1. Download the image files from: xxxxxxx . Put the files in desired folder, for example **/home/User/Archlinux.**

2. Insert the card-reader into the Linux host machine.

IMPORTANT NOTE - this example is given with "sdb1 and sdb2" devices but it could enumerate differently on your host, so confirm what it enumerates as before running fdisk!

3. Unmount the card-reader:
& sudo umounth /dev/sdb1

4. Format the SD-card using fdisk:

```
& sudo fdisk /dev/sdb
```

5. With the help of the menu create two partitions. The steps are the following:

5.1. Type "**p**" to view existing partitions on the SD-card
5.2. Type "**d**" to delete all partitions
5.3. Type "**n**" to create new one
5.4. Set "**1**" as partition number
5.5. Type "**p**" to select primary partition
5.6. Press **Enter** to select default beginning sector
5.7. Type "**+32M**" to create 32MB partition
5.8. Type "**t**" to change partition type
5.9. Type "**53**"
5.10. Type "**n**" to create another partition
5.11. Set "**2**" as partition number
5.12. Type "**p**" to set partition as primary
5.13. Press **Enter** to set default size of the partition
5.14. Type "**w**" to write partitions to the SD-card

6. Create the second partition with ext3 file system:

```
& sudo mkfs.ext3 /dev/sdb2
```

7. Mount the second partition:

```
& sudo mount /dev/sdb2 /mnt/mmc
```

If you don't have mountpoint you should create one. Just type:

```
sudo mkdir /mnt/dir
```

8. Login as root for the next operations:

```
& sudo su
```

IMPORTANT NOTE: It is necessary to log-in as root, not as super-user!

9. Extract the downloaded tarball into the second partitions:

```
& tar -xzf /home/User/Archlinux/xxxxx.tar.gz
```

10. The next step is to to write the bootloader and kernel image to the first partition.

```
& dd if=/Archlinux/xxxxx.img of=/dev/sdb1 ibs=512 seek=4 conv=sync,notrunc
```

11. Unmount the SD-card, it should be ready for use.

```
& cd
& umount /dev/sdb2
```

12. Download the new kernel patch from: xxxxxxxxxxxxxxxxxxxxx

13. Put the file in USB flash drive. Insert the flash drive into OLinuXino board.

```
& mount /dev/sdc1 /mnt/usb
& cp -fv /home/User/Downloads/<file> /mnt/usb
& umount /mnt/usb
```

Again this is example. You should correct this commands with your paths and corresponding devices.

IMPORTANT NOTE: The following commands are entered into OLinuXino terminal.

14. Mount the flash drive

```
& mount /dev/sdb1 /mnt/usb
```

15. Run **pacman** to update the kernel:

```
& pacman -S /mnt/usb/kernel26-olinuxino-2.6.35.3-6-arm.pkg.tar.xz
```

16. Follow the instructions. Just type "y" when promted.

17. Reboot

IMPORTANT NOTE: If you are connected to internet, you can use the following command:

```
& pacman -U http://xxxxxxx.tar.xz.
```

Using this you can skip the part with the USB flash drive. Using the same command you could download different packages:

```
& pacman -Sy <package name>
```

II. Set up the internet

This is a working example, but you might wish to read some additional information, if this doesn't work for you.

With every reboot the device is assigned with different MAC address. To change this use the commands:

```
& ip link set dev usb0 down
& ip link set dev usb0 address aa:bb:cc:dd:ee:ff
& ip link set dev usb0 up
```

To set an IP address enter:

```
& ifconfig usb0 192.168.0.249
```

or whatever address you want.
After that add default gateway:
route add default gw 192.168.0.1 usb0

Finally you should add DNS-server. You should modify **/etc/resolv.conf** with **vi**, or other program. For example:

```
& vi /etc/resolv.conf
```

Press **d** several times to remove all lines. Press **I** to enter insert mode. Type:

```
& nameserver 192.168.0.1
```

(Again this is example. You should add you DNS-server.)
After you finish entering this press **Esc**. This will set the program in command mode. In command mode press:

```
& :wq
```

This should do the work. To check connection use **ping**. If it doesn't check your configuration again and router settings. You can use **ifconfig** to see OLinuXino settings.

III. Using GCC to make Hello_World

In the home folder there are some examples

```
& cd /home/examples
& ls
```

And you should see all files and folders. You can create new folder (if you want) with the command:

```
& mkdir <some dir>
```

To see our HelloWorld program go to **/HelloWorld** folder

```
& cd /home/examples/HelloWorld
& ls
```

In this folder are two files:

```
- hello
- hello_world.c
```

The C file is the sourse code. Actually this is text file with .c extention. Type **vi** or **cat** hello_world.c to view the source. You could modify the source whatever you want. After the source is ready to compile the file use:

```
& gcc -o hello_world.c hello
```

For more complicated compiling type:

```
& man gcc
```

To execute the program:

```
& ./hello <some name>
```

The result of the execution should be:

Hello <some name>.

In the other directory /home/i2c are three examples for hardware I2C. You could compile them using the same command:

```
& gcc -o <file>.c <file>
```

After execution you could see the I2C working using osciloscope connected to the corresponding GPIOs.

You can see that the LED1 on the board is blinking. This is a backgrond shell process. Again in the /home you can see a third folder led_blink. In this one there is a file led_blink. Type:

```
& cat led_blink
```

You can see that this is shell script. To execute this script every time the linux is loaded the path should be added to the file rc.local.

```
& vi /etc/rc.local
```

Add the path to the script file:

```
/home/examples/led_blink/led_blink &
```

The "**&**" means that the script will be running in background.

2.11 How to use a custom Wi-Fi dongle based on RealTek RTL8188CUS and RTL8192CU under ARCH Linux

This procedure was done on the Olinuxino-Maxi. It was connected with a wired ethernet during the procedure. Once the WiFi is functional, the wired ethernet can be removed. The microSD card can then be moved to another Olinuxino board such as the Micro.

Log into the Olinuxino board.

upgrade the system:

```
pacman -Syu
```

Then install development tools

```
pacman -S base-devel
```

Delete empty directories:

```
rmdir /usr/lib/modules/2.6.35-6-ARCH+/build
rmdir /usr/lib/modules/2.6.35-6-ARCH+/source
```

Install kernel source:

```
pacman -S kernel26-headers-olinuxino
```

Get the Realtek drivers from the Realtek web site:
http://www.realtek.com.tw/downloads/downloadsView.aspx?Langid=1&PFid=48&Level=5&Conn=4&ProdID=277&DownTypeID=3&GetDown=false&Downloads=true

I downloaded RTL8188CUS (the one showing up on lsusb)
If you choose RTL8192CU you get the same file

The file comes as a zip file:
RTL819xCU _USB_linux_v3.4.3_4369.20120622.zip

Assume you downloaded it to your workstation.
Unzip the file.
Inside there is a folder called driver with a file inside called:
rtl8188C_8192C_usb_linux_v3.4.3_4369.20120622.tar.gz

Copy that file to the Olinuxino target:

```
scp rtl8188C_8192C_usb_linux_v3.4.3_4369.20120622.tar.gz root@ip_addr_of_Olinuxino:/root
```

log into Olinuxino

```
cd /root
mkdir driver
mv rtl8188C_8192C_usb_linux_v3.4.3_4369.20120622.tar.gz driver
cd driver
tar xzf rtl8188C_8192C_usb_linux_v3.4.3_4369.20120622.tar.gz
cd rtl8188C_8192C_usb_linux_v3.4.3_4369.20120622
vi Makefile
```

Change the line:
CONFIG_PLATFORM_I386_PC = y
to
CONFIG_PLATFORM_I386_PC = n

Below it add the line:
CONFIG_PLATFORM_ARM_iMX233 = y

find the block:

```
ifeq ($(CONFIG_PLATFORM_ARM_PXA2XX), y)
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN
ARCH := arm
CROSS_COMPILE := arm-none-linux-gnueabi-
KVER  := 2.6.34.1
KSRC ?= /usr/src/linux-2.6.34.1
endif
-------------------------------------------------------
```

Below it adds the block:

```
-------------------------------------------------------
ifeq ($(CONFIG_PLATFORM_ARM_iMX233), y)
EXTRA_CFLAGS += -DCONFIG_LITTLE_ENDIAN
ARCH := arm
CROSS_COMPILE := /usr/bin/
KVER  := KVER  := 2.6.35-6-ARCH+
KSRC ?= /usr/src/linux-2.6.35-6-ARCH+
MODDESTDIR := /usr/lib/modules/2.6.35-6-ARCH+/kernel/drivers/net/wireless/rtl818x
endif
-------------------------------------------------------
```

Save and exit:

```
:wq
```

```
make clean
make
rmmod 8192cu.ko  (not really needed)
insmod 8192cu.ko
make install
```

Verify that the driver has been installed:

```
[root@alarm wireless]# ls -l /sys/class/net
total 0
drwxr-xr-x 4 root root 0 Aug 14 16:55 lo
drwxr-xr-x 4 root root 0 Aug 14 16:55 usb0
drwxr-xr-x 5 root root 0 Aug 14 16:43 wlan0
```

wlan0 is the new driver

```
reboot
```

Make sure you can see both usb0 (ethernet) and wlan0 on all the following:

```
ifconfig
ls -l /sys/class/net
iwconfig
```

you may need to bring the interface up manually:

```
ip link set wlan0 up
```

scan for access points:

```
iwlist wlan0 scan
cp /etc/wpa_supplicant/wpa_supplicant.conf /etc/wpa_supplicant/wpa_supplicant.conf.original
wpa_passphrase myssid "my_secret_passkey" > /etc/wpa_supplicant/wpa_supplicant.conf
```

Associate with access point according to the encryption type:

Encryption	Command
No Encryption	iwconfig wlan0 essid "linksys"
WEP w/ Hex Key	iwconfig wlan0 essid "linksys" key "0241baf34c"
WEP w/ ASCII passphrase	iwconfig wlan0 essid "linksys" key "s:pass1"
WPA	wpa_supplicant -B -Dwext -i wlan0 -c /etc/wpa_supplicant/wpa_supplicant.conf

Verify association:

```
iwconfig wlan0
```

Assign IP address:

```
  dhcpcd wlan0                 (dhcp)
or                         (static)
  ip addr add 192.168.0.2/24 dev wlan0
  ip route add default via 192.168.0.1
```

To set up automatically at boot:

```
pacman -S netcfg
pacman -S wpa_actiond
pacman -S ifplugd

cp /etc/network.d/examples/wireless-wpa /etc/network.d/my_wifi_network
cp /etc/network.d/examples/ethernet-dhcp /etc/network.d/my_eth_network
```

```
  vi /etc/network.d/my_wifi_network
```

configure all parameters

```
  vi /etc/network.d/my_eth_network
```

configure all parameters (make sure you change eth0 to usb0)

You can have multiple profiles (e.g. for roaming, etc.)

To manually connect a profile:

```
  netcfg my_wifi_network
```

To manually disconnect a profile:

```
  netcfg down my_wifi_network
```

There are 3 daemon options for configuring networks at boot time.

1. network -- the original setting. don't use that since we're using netcfg
2. net-profiles

or

3. net-auto-wireless and net-auto-wired

This procedure is using the last option.

```
vi /etc/rc.conf
```

Look for DAEMONS=(..)
delete network and add:
  net-auto-wireless and net-auto-wired

```
vi /etc/conf.d/netcfg
```

Change to: NETWORK=(@my_wifi_network @my_eth_network)
Make sure you have the correct names for WIRED_INTERFACE and WIRELESS_INTERFACE (change eth0 to usb0)

Also see:
https://wiki.archlinux.org/index.php/Beginners'_Guide#Setup_wireless_network
https://wiki.archlinux.org/index.php/Wireless_Setup#Getting_an_IP_address
https://wiki.archlinux.org/index.php/Netcfg

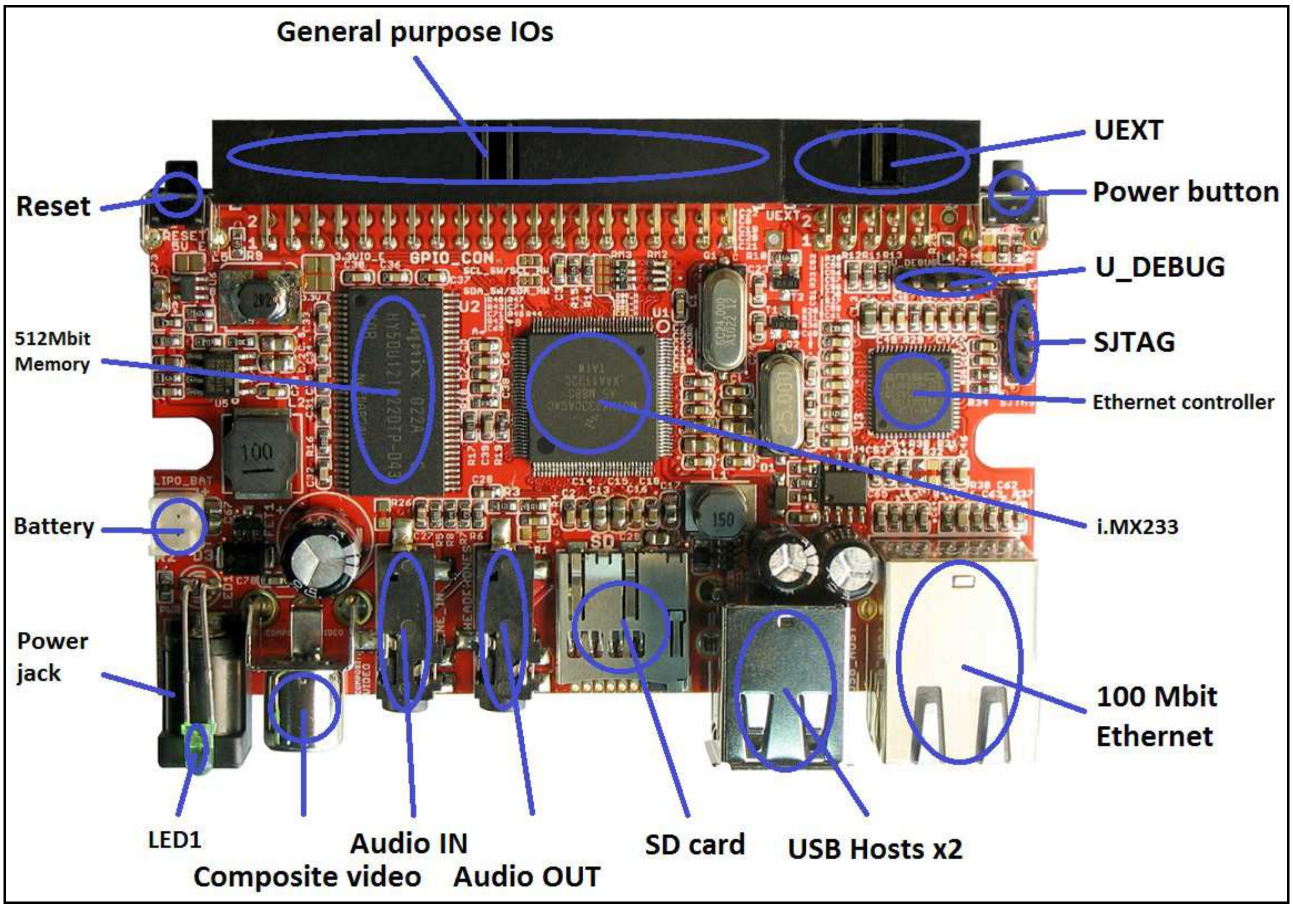
Other commands:
ip link show wlan0

CHAPTER 3: OLINUXINO BOARD DESCRIPTION

3. Introduction to the chapter

Here you get acquainted with the main parts of the board. Note the names used on the board might differ from the names used below to describe them. For the actual names check the OLinuXino board itself.

3.1 Layout (top view)



CHAPTER 4: THE iMX233 MICROCONTROLLER

4. Introduction to the chapter

In this chapter is located the information about the heart of OLinuXino – its microcontroller. The information is a modified version of the datasheet provided by its manufacturers.

4.1 The microcontroller

- ARM926 CPU Running at 454 MHz
- Integrated ARM926EJ-S CP
- 16-Kbyte data cache and 16-Kbyte instruction cache
  — One-wire JTAG interface
  — Resistor-less boot mode selection using integrated OTP values
- 32Kbytes of Integrated Low-Power On-Chip RAM
- 64 Kbytes of Integrated Mask-Programmable On-Chip ROM
- 1 Kbit of On-Chip One-Time-Programmable (OCOTP) ROM
- Universal Serial Bus (USB) High-Speed (Up to 480 Mb/s), Full-Speed (Up to 12 Mb/s)
  — Full-speed/high-speed USB device and host functions
  — Fully integrated full-speed/high-speed Physical Layer Protocol (PHY)
  — Mass storage host-capable (uncertified by USB-IF)
- Power Management Unit
  — Single inductor DC-DC switched converter with multi-channel output supporting Li-Ion batteries.
  — Features multi-channel outputs for VDDIO (3.3 V), VDDD (1.2 V), VDDA (1.8 V), VDDM (2.5V) and regulated 4.2V source.
  — Direct power from 5-V source (USB, wall power, or other source), with programmable current limits for load and battery charge circuits.
  — Silicon speed and temperature sensors enable adaptive power management over temperature and silicon process.
- Audio Codec
  — Stereo headphone DAC with 99 dB SNR
  — Stereo ADC with 85 dB SNR
  — Stereo headphone amplifier with short-circuit protection and direct drive to eliminate bulky capacitors
  — Amplifiers are designed for click/pop free operation.
  — Two stereo line inputs
  — Microphone input