Oxalis

Getting Started

Document:

Document:	Getting Started	Content of the document: First steps to power up, the board, to run uboot, build kernel and boot linux .
Version:	001	
Creator:	ANT/MSB	
Date:	01.12.2018	
Release Status:	Release	Pages: 10

Table of Contents

List of Figures

List of Tables

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

Change index:

Rev.:	Date:	Name:	Adaptation:
001	01.12.2018	ANT/MSB	Kick off Document

1. Scope of Document and Board Overview

After reading this document you will be able to power up the board, run the bootloader, build the Linux image and to run Linux from SD-Card or USB-Memory-Stick.

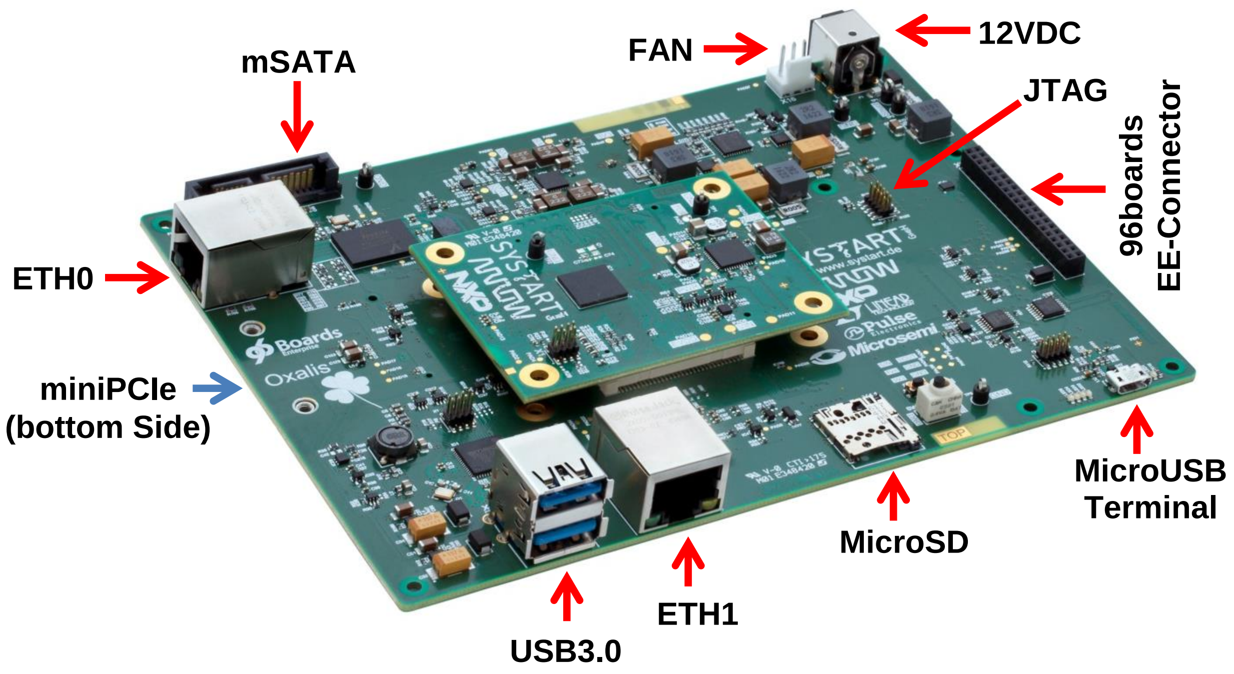


Figure 1: Oxalis with SoM

Oxalis is a modular development board compliant with the 96Boards Enterprise Edition (EE) Platform specification. Developed by SYSTART, the platform consists of a carrier board and a System on Module (SoM) based on the NXP Layerscape LS1012A processor, with a single ARM Cortex-A53 core running up to 800 MHz and 1GB of 16bit DDR3L memory. Implemented Interfaces:

- 2 Ethernet
- 2 USB 3.0
- 1 PCI-Express
- 1 SD-Card
- 1 SATA
- 4 User LEDs
- JTAG LS1012
- Expansion Connector
    - UART
    - SPI
    - I2C
    - GPIO, Reset, Power button

NOTE

SYSTART provides an expansion board, Oxalis IO Breakout Board One (BB1) with all the interfaces listed under the expansion connector section above. This expansion connector can be used for small electronic applications and Oxalis interface verifications.

2. Scope of Delivery

The following components are included in the delivery.

- Oxalis board
- LS1012A SoM mounted onto the Oxalis
- A plug fitting the 12V power jack on the board, in case you need to fashion a power supply cable.
- Leaflet with information sources.

3. Required Equipment

- 12V / 3A DC power supply
- Micro-USB cable
- USB Stick or MicroSD-Card
- A Windows, Linux or Mac computer with your favorite terminal software

4. Resources and Information

For the sake of completeness, all sources of information are listed below. The required sources for working through this document are marked in bold.

https://www.shop.systart.de/embedded

- **Obtaining Oxalis**
- Obtaining expansion boards
- Obtaining accessories

https://www.ebs-systart.com/oxalis

- **Binaries**
  - Download Binary: **kernel_stable.itb**
- Datasheets
- Schematics, Assembly Drawings
- SDK Build Environment and Sources for beginners
- Links to other resources

https://github.com/ebs-systart/oxalis

- Developer Information
- Source Codes
- Examples

https://www.96boards.org/products/ee/

- Developer Community and Forum
- Other 96boards Edition compliant resources

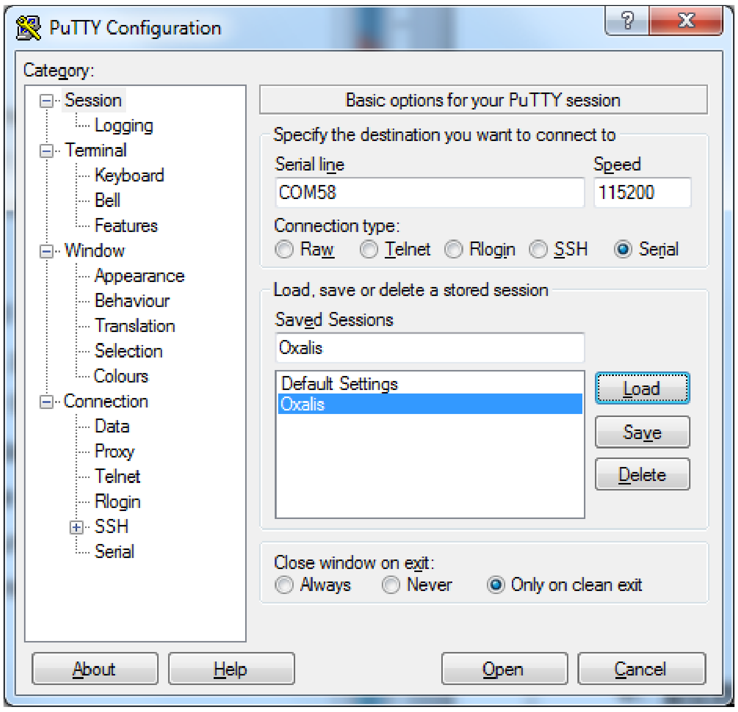
https://www.nxp.com/

- **NXP-QoriQ Linux SDK v2.0 source**
  - For the download you have to register with NXP.

5. Running the Bootloader U-Boot

In this chapter you will learn how to power up the board and use U-Boot.

- Connect your computer to the Micro-USB port (X1) of the Oxalis via the Micro-USB cable. Your system should recognize the Oxalis as multiple devices, one of them called "MBED".
- Start a terminal software on your computer (e.g. putty or minicom) with following settings
  - Baud rate 115200
  - 8-N-1 (8 data bits - no parity bit - one stop bit)



Basic options for your PuTTY session

Specify the destination you want to connect to

Serial line: COM58
Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Load, save or delete a stored session

Saved Sessions: Oxalis

Default Settings
Oxalis

Close window on exit: Always Never [✓ Only on clean exit]

Figure 2: Putty

- Connect the Oxalis to the 12 Volt Power supply and switch it on
- Watch the LED next to the MicroUSB Connector X1

- During a data transmission, this LED flashes.

- On your terminal program you see the Oxalis going through the boot process

```
U-Boot 2016.092.0+ga06b209 (Jun 29 2017 - 16:33:07 +0200)

SoC:  LS1012AE Rev1.0 (0x87040010)
Clock Configuration:
       CPU0(A53):800  MHz
       Bus:      250  MHz  DDR:      1000 MT/s
Reset Configuration Word (RCW):
       00000000: 08000008 00000000 00000000 00000000
       00000010: 35080000 c000000c 40000000 00001800
       00000020: 00000000 00000000 00000000 00014551
       00000030: 00000000 18c2a120 00000096 00000000
I2C:   ready
DRAM:  1022 MiB
SEC0: RNG instantiated
SEC Firmware: Bad firmware image (not a FIT image)
PSCI: PSCI does not exist.
Did not wake secondary cores
Using SERDES1 Protocol: 13576 (0x3508)
MMC:   FSL_SDHC: 0, FSL_SDHC: 1
SF: Detected S25FS512S with page size 256 Bytes, erase size 256 KiB, total 64 MiB
```

.
.
.

```
pfe_hw_init: done
pfe_firmware_init
pfe_load_elf: no of sections: 13
pfe_firmware_init: class firmware loaded
pfe_load_elf: no of sections: 10
pfe_firmware_init: tmu firmware loaded
ls1012a_configure_serdes 0
PCIe0: pcie@3400000 Root Complex: no link
pfe_eth0 [PRIME], pfe_eth1
Hit any key to stop autoboot:  0
=>
```

- Hit one key to stop autoboot.

Congratulations! You are running U-Boot now.

6. Building an image from the NXP QoriQ Reference SDK

The following instructions assume you are running Linux and are well versed in using a terminal. On any other operating system we recommend installing VirtualBox from https://www.virtualbox.org or any other virtualization system that lets you run a Linux distribution.

NOTE 1

You can also skip this whole chapter if you download the prebuilt **kernel_stable itb** image.

NOTE 2

Quoting the NXP SDK Documentation: "Yocto Project supports typical Linux distributions: Ubuntu, Fedora, CentOS, Debian, OpenSUSE, etc. More Linux distributions are continually being verified. This SDK has been verified on following Linux distributions: Ubuntu 14.04, CentOS-7.1.1503, Debian 8.2, Fedora 22 and OpenSUSE 13.2" You can use other distros, notably a more current LTS Ubuntu, but then you get Warnings to ignore and patches to apply. It will not work out of the box.

6.1 Install and update the NXP Reference SDK

The SDK is only available for download with an NXP.com account, which is free. (Detailed descriptions are on nxp.com).

- Download, unpack and install the NXP-QoriQ Linux SDK v2.0 source.
- Download and install the NXP-QoriQ Linux SDK v2.0 17.03 Upgrade.
- Change into your SDK directory using the linux console.
- Run . `./fsl-setup-env -m ls1012frdm`
- Accept the EULA, you should now have been transferred into a directory called `build_ls1012afrdm`.
- Verify your installation state by running `bitbake fsl-image-core` (This will take a long time, depending on your system)

If bitbake runs through without major errors, your installation of the Yocto SDK has been successful.

NOTE

The image that resulted from this test-build carries no Oxalis specific customizations but might be bootable on your board (not supported).

6.2 Pull in the source code for Oxalis

Yocto is essentially a complete source tree of Linux and all its tools that can be changed and configured using a layering approach. EBS-SYSTART provides a BSP in the shape of a Yocto layer.

NOTE

If you want to understand what is happening in detail, refer to the Yocto documentation.

6.3 Get the BSP Layer

Execute the following:

```
$  cd <sdk-install-dir>/

$  mkdir systart

$  cd systart

$  git clone https://github.com/ebs-systart/meta-systart-oxalis

$  source meta-systart-oxalis/scripts/oxalis-setup-env
```

You should now have been switched into a new build directory

```
$  sdk-install-dir>/build_oxalisls1012a
```

(this is called the `<build-dir>` in the following sections).

From here `bitbake` should be able to build these images:

- `oxalis-image-kernelitb`: Kernel, U-Boot and minimal Linux environment (BusyBox) ramdisk in U-Boot loadable format
- `oxalis-image-core`: RootFS image with core components
- `oxalis-image-full`: RootFS with full content of the yocto distribution, including GUI (very large, extreme build times, not supported).

NOTE

The `bitbake` process should take less time than before if you selected the core or kernel image.

The best starting point is `oxalis-image-kernelitb` because of its compactness, its simplicity and the fact that our deployment guide assumes you have built it. Often you might want to build a complete distribution on a root filesystem on a larger storage medium than the 64MB onboard flash, then the `-core` and `-full` rootfs images are what you need.

After the build you can find the images in:

- `<sdk-install-dir>/<build-dir>/tmp/deploy/images/ls1012aoxalis/`

7. Deploying the kernelitb ramdisk image to Oxalis

This chapter assumes you have successfully completed the above chapter **5 Running the Bootloader** and can see the U-Boot console on Oxalis. It also assumes that you have successfully completed the chapter **6 Building an image from the NXP QoriQ Reference SDK** or you have just downloaded the prebuilt kernel_stable.itb image.

To boot Linux you have to execute three steps:

- Prepare storage medium
- Load the Linux image into Oxalis's RAM
- Transfer it into onboard flash memory for automatic booting

Or

- Just boot Linux on-the-fly

The onboard flash memory is 64MB in size. You should keep your Kernel and Ramdisk image below 55MB in size for unproblematic operation. (The first 5 MB of onboard flash are reserved for bootloader and initialization data.)

7.1 Prepare Storage medium

Use your PC to prepare the storage medium as described below:

- Format the storage medium (SD-Card or USB-Memory-Stick) with fat32.
- Copy and rename the downloaded `kernel_stable.itb` image to `kernel.itb` and place it in the root folder of the storage medium.

- Ensure that the image has been fully written. Use `Safe remove` on Windows and the `sync` command on the linux console.

Now you have a bootable Linux storage medium.

7.2 Load the Linux image into Oxalis's RAM

- Power down your Oxalis
- Insert the medium in the appropriate slot
- Power the Oxalis up again
- Watch the bootloader and stop autoboot loader
- Load the Kernel ramdisk into RAM:
    - SDard: `fatload mmc 0 0x96000000 kernel.itb`
    - USB Stick: `usb start; fatload usb 0 0x96000000 kernel.itb`

7.3 Transfer Linux image from RAM to Flash

If the loading was successful, you can now boot directly with following commands:

```
pfe stop; bootm 0x96000000
```

Now linux boots the minimal ramdisk environment and the login prompt will appear. Login with **root** and press enter. No password is required.

NOTE

Why the `pfe stop` command? The packet forwarding engine (PFE) is part of the hardware accelerated Ethernet subsystem of the LS1012A. It needs to be shut down before Kernel boot, so the Kernel code can reinitialize PFE for itself.

8. Configure Autoboot from QSPI

If your ramdisk size is <50MB you can set up autobooting on the Oxalis, without any external storage devices necessary. This is possible by writing the ramdisk to Oxalis's onboard flash memory and configure U-Boot to automatically load the ramdisk from there and boot it.

Press reset button and repeat the instructions in chapter **7.2 Load the Linux image into Oxalis's RAM**.

- Flash it to QSPI using `sf write 0x500000 0x96000000 $filesize`

Set boot command environment variable:

```
$   setenv bootcmd "run oxalissettings; run ramargs; pfe stop;
    sf probe; sf read 0x96000000 0x500000 $filesize; bootm
    0x96000000"s

$   saveenv
```

NOTE

The `saveenv` command writes the small bootcmd script into U-Boot's config area in onboard flash.

Press the reset button and watch Oxalis autobooting.

9. Congratulations

You are now running a full Linux Kernel with a minimal Linux environment called BusyBox on your Oxalis! From here it is possible to add your own applications, either by loading them via the network or by adding your own yocto layer (complex).

The official BusyBox page is here:
https://busybox.net

Wikipedia has a very nice and concise explanation of what BusyBox actually is:
https://en.wikipedia.org/wiki/BusyBox

Document_Version_V001
Created MSB/23.10.2018; Released STL/23.10.2018