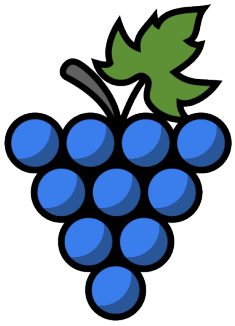
Scalys

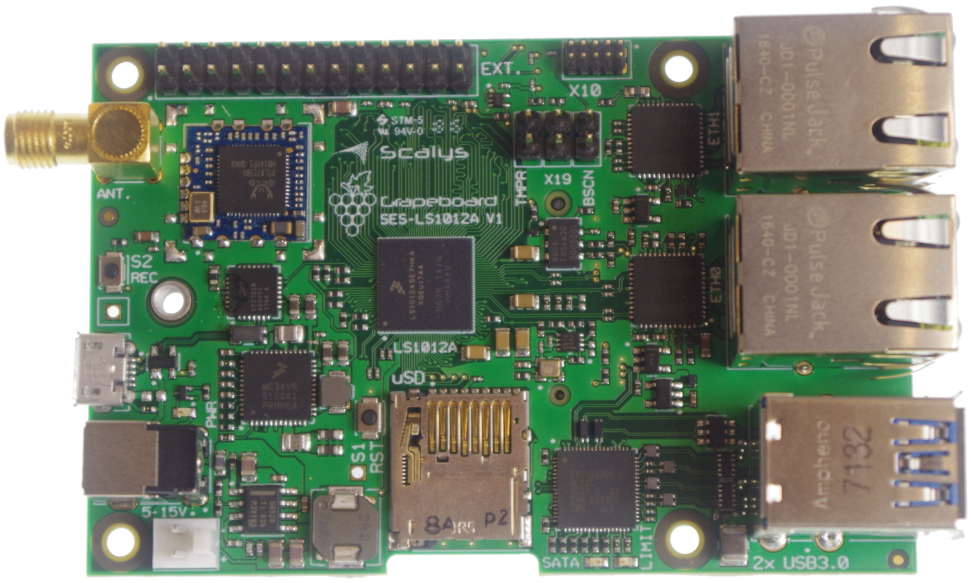
Grapeboard BSP user guide

Description

This technical reference document serves as a guide on how to build and use the BSP for the Scalys grapeboard.



partner of NXP



Introduction

This document serves as a technical reference document on how to build the Board Support Package (BSP) for the Scalys Grapeboard™. This document is meant to supplement the available documentation already provided by NXP[1, 2].

The Grapeboard™ features a QorIQ LS1012A processor[3] that contains a single Arm Cortex-A53 core. NXP provides two flavours of BSP s for this specific processor. The first being the Yocto based Linux Software Development Kit (SDK) for QorIQ processors and the second flavour is the more recently introduced Layerscape Software Development Kit (LSDK). The QorIQ Linux SDK is based upon the Yocto Project [4], whilst the LSDK is based on a more disaggregated flex-building environment [5].
This document assumes the reader has a basic knowledge of platform software and deployment, and it will not go into too much detail of the actions performed to build the BSPs. This document currently only describes building the BSP through the LSDK method. Contact Scalys for more information on the Yocto based SDK. Additionally, NXP intends to update the LSDK regularly, however this document only focusses on a specific release version. Contact Scalys when updates are desired.

Rev	Date	Description
1.0	18th May, 2018	First release

Contents

List of Tables

1

Preparation

This document assumes the system used to build the BSP is a newly installed OS.

1.1 Install the dependencies using the package manager

A list of additional packages used throughout this user guide are as follows:

curl
git
gawk
wget
tftpd
tftp
isc-dhcp-server
nfs-kernel-server
gcc-aarch64-linux-gnu
device-tree-compiler
libncurses5-dev
libncursesw5-dev
docker.io

You can install them through your package manager using the following command:

```
sudo apt-get install curl git gawk wget tftpd tftp isc-dhcp-server nfs-kernel-server \
gcc-aarch64-linux-gnu docker.io device-tree-compiler libncursesw5-dev libncurses5-dev
```

1.1.1 LSDK

The LSDK build has been verified to work on Kubuntu 16.04.

1.2 Git setup

```
# Setup username and email for git if not already done
git config --global user.name "Your Name"
git config --global user.email "user@example.com"
```

2

Building images

2.1 Building with LSDK

The following stepwise guide primarily follows the steps given in the LSDK documentation. More in-depth information regarding the LSDK flexbuild environment can be found in the NXP manual[2] and the *REAMDE.md* file found in step 4 below. For compatibility reasons we will choose to use an Ubuntu 16.04 Docker container in the flex-build environment, which allows for the use of Linux distros other than Ubuntu 16.04 on the host machine.

In the this section we will build our LSDK Root File System (RFS) which includes the Linux kernel and device tree. Note that only a subset of the default LSDK commands is supported for the Grapeboard™.

1. First begin with creating a working directory in your home directory:

```
mkdir -p ~/grapeboard/ && cd ~/grapeboard/
```

2. Set-up Docker on the host machine. If not already installed, then do so from your package manager. Add your user to the docker group.

```
sudo addgroup --system docker
sudo usermod -aG docker <accountname>
sudo gpasswd -a <accountname> docker
sudo service docker restart
```

   Make sure you logout once to apply the changes to your user.

3. Manually download the flexbuild_lsdk1712.tgz tar-ball[6] from the NXP webpage:

   Source: Layerscape Software Development Kit (rev 17.12) *44.3 KB*
   flexbuild_lsdk1712.tgz
   MD5 Signature *a3dd27757377c53bb1dca599cb3eaf21*

4. And set-up the flex-build environment and open the directory.

```
tar xvzf flexbuild_lsdk1712.tgz -C /home/<user>/grapeboard && cd flexbuild
```

5. Patch the flexbuild environment to support the Grapeboard™. Ensure you apply the latest patch to the associated flexbuild version.

```
curl https://git.scalys.com/lsdk/lsdk-scalys-bsp/plain/grapeboard_support_lsdk-1712.patch |\
    patch -p1
```

6. Enter the docker environment.

```
source setup.env
flex-builder docker
source setup.env
```

7. Build generic RFS for arm64 with additional packages. Modify the *'additional_packages_list_full_grapeboard'* file to add/remove packages. It can be found in *'packages/apt-packages/'*.

```
flex-builder -i mkrfs -a arm64 -m grapeboard -B additional_packages_list_full_grapeboard
```

8. Now we have to build the Linux kernel. With the following command we use the default configuration:

```
flex-builder -c linux -a arm64 -m grapeboard
```

   Alternatively, we can append the *'-B menuconfig'* argument to modify the default configuration or use *'-B fragment:"defconfig <custom.config>"'* to specify custom configuration files.

9. Download the pre-compiled components tar ball for the ARM64 target[7]:

   Source: LSDK (rev 17.12) prebuilt components for ARM64 target *135.0 MB*
   components_arm64.tgz
   MD5 Signature *e22f92f6311e7678bcf5f95b9c4b4c4f*

   Provided you accept the license[7] you can also directly download it using the following command:

```
wget http://www.nxp.com/lgfiles/sdk/lsdk1712/components_arm64.tgz
```

10. Insert the arm64 components and modules

```
tar xvzf components_arm64.tgz -C build/images
flex-builder -i merge-component -a arm64 -m grapeboard
```

11. Optionally we can compress the RFS

```
flex-builder -i compressrfs -a arm64 -m grapeboard
```

12. Leave the docker environment

```
exit
```

Now that we have a RFS prepared, we have the option to go directly to section 5.1 in order to prepare the SD card for the Grapeboard™ or continue with the following sections to build and modify U-boot and the Linux kernel.

2.2 Standalone Linux kernel build

Modifying the Linux kernel to your requirements is recommended to do within the flexbuild. Refer to the official NXP guide[2] to get more details on how to accomplish this. In this section we will show how to build a Linux kernel in a stand-alone fashion.

Start by cloning the correct Linux kernel repository (with the appropriate branch) from the Scalys website. For this example we take the kernel used in the LSDK with the Grapeboard™ patches, so make sure you also use the associated RFS for it to work.

```
git clone http://git.scalys.com/lsdk/linux -b scalys-lsdk-1712 && cd linux/
```

Configure the kernel:

```
CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 make defconfig lsdk.config grapeboard_wireless.config
```

Optionally, we can now configure and modify the kernel manually with:

```
CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 make menuconfig
```

Build the kernel:

```
CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 make -j 8
```

Convert the kernel image to a uImage format:

```
CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 mkimage -A arm64 -O linux -T kernel -C gzip \
-a 0x80080000 -e 0x80080000 -n Linux -d arch/arm64/boot/Image.gz uImage
```

Install the generated files in the RFS , e.g. on the SD card:

```
sudo cp uImage <path_to_rfs>/boot/uImage
sudo cp arch/arm64/boot/dts/freescale/grapeboard.dtb <path_to_rfs>/boot/grapeboard.dtb
sudo CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 make INSTALL_MOD_PATH=<path_to_rfs> modules_install
```

2.3 Building U-boot

Although the flexbuild build system does support the command to build a U-boot image, for the Grapeboard™ it has been chosen to focus on building and customizing U-boot outside of the flexbuild environment.

Download the Scalys U-boot repository:

```
git clone http://git.scalys.com/lsdk/u-boot -b scalys-lsdk-1803 && cd u-boot
```

We now can choose between two default options to configure U-boot.

- PCIe support on the M.2. connector:

```
ARCH=aarch64 CROSS_COMPILE=aarch64-linux-gnu- make grapeboard_pcie_qspi_defconfig
```

- SATA support on the M.2. connector:

```
ARCH=aarch64 CROSS_COMPILE=aarch64-linux-gnu- make grapeboard_sata_qspi_defconfig
```

Optionally, we can now configure U-boot with additional options. The Grapeboard™ specific options may be found under *"ARM architecture > Grapeboard configuration options > ..."*.

```
ARCH=aarch64 CROSS_COMPILE=aarch64-linux-gnu- make menuconfig
```

Build U-boot image:

```
ARCH=aarch64 CROSS_COMPILE=aarch64-linux-gnu- make -j 8
```

Unlike for the existing NXP LS1012A development boards, the Grapeboard™ U-boot and Pre-Boot Loader (PBL) binaries are combined together in order to reduce the total memory footprint. This combined binary is automatically generated with the U-boot '*make*' command, which results in a file named: '*u-boot-with-pbl.bin*'. Through the '*make menuconfig*' U-boot compiler command you have the option to select a different PBL binary file or skip it entirely.

2.4 Secure boot

Contact Scalys for more information.

3

Configuration necessary for a TFTP boot

These steps are optional and only necessary when the target is connected directly to the computer used to build the BSP.

3.1 TFTP server installation and configuration

Install the TFTP server (if not already done):

```
sudo apt-get install xinetd tftpd tftp
```

Create/edit the "/etc/xinetd.d/tftp" file and add the following entry:

```
service tftp
{
   protocol    = udp
   port        = 69
   socket_type = dgram
   wait        = yes
   user        = nobody
   server      = /usr/sbin/in.tftpd
   server_args = /tftpboot
   disable     = no
}
```

Create a folder to serve the TFTP data:
Warning: TFTP has no security so be aware this folder is NOT SECURE!

```
sudo mkdir /tftpboot
sudo chmod -R 777 /tftpboot
sudo chown -R nobody /tftpboot
sudo chmod g+s /tftpboot
```

Restart the xinetd service:

```
sudo /etc/init.d/xinetd restart
```

3.2 DHCP Server installation and configuration

Install the DHCP server (if not already done):

```
sudo apt-get install isc-dhcp-server
```

edit the '/etc/network/interfaces' file, where 'eth1' is the chosen interface to the board:

```
# Make sure the network ranges match your host system!
auto <interface>
allow-hotplug <interface>
iface <interface> inet static
address 192.168.1.1
netmask 255.255.255.0
```

and edit the '/etc/dhcp/dhcpd.conf' file:

```
default-lease-time 600;
max-lease-time 7200;

# Optionally, we can assign static addresses for the targets
host <target_hostname> {
   hardware ethernet 00:11:22:33:44:55;
   fixed-address 192.168.1.180;
}

subnet 192.168.1.0 netmask 255.255.255.0 {
   range 192.168.1.150 192.168.1.200;
   option routers 192.168.1.254;
   option domain-name-servers 192.168.1.1, 192.168.1.2;
}
```

Specify the interface in the '/etc/default/isc-dhcp-server' file:

```
INTERFACES="<interface>"
```

Restart the DHCP service.

```
sudo service isc-dhcp-server restart
```

4

Grapeboard flash configuration

4.1 Primary flash partitioning

The primary NOR flash has been partitioned into the following functional parts:

u-boot
Partition containing the PBL (Reset Configuration Word (RCW) + Pre-Boot Instructions (PBI)) and U-boot bootloader

env
Environment settings as used in U-boot

pfe
Primary firmware for the PFE under U-boot

ppa
Primary Protected Application (PPA) firmware.

u-boot_hdr
U-boot header for secure boot.

ppa_hdr
PPA header for secure boot.

UBI/rootfs
Remainder is available for user to place small a RFS. By default its prepared for UBIFS.

The partitioning may be modified to fit a specific application, as long as the requirement is met for the PBL binary to start at address 0x0. Altering these partitions requires updates to the offset(s) defined within the PBI and/or U-boot.

Table 4.1: Primary NOR flash Partitioning

No.	Name	Size	Offset
0	u-boot	0x00200000	0x00000000
1	env	0x00040000	0x00200000
2	pfe	0x00040000	0x00240000
3	ppa	0x00100000	0x00280000
4	u-boot_hdr	0x00040000	0x00380000
5	ppa_hdr	0x00040000	0x003c0000
6	UBI	0x03c00000	0x00400000

```
mtdparts: mtdparts=1550000.quadspi:2M@0x0(u-boot),256k(env),256k(pfe),1M(ppa),256k(u-boot_hdr),256k(
    ppa_hdr),-(UBI)
```

4.2 Rescue flash partitioning

The rescue flash is read-only by default and contains a U-boot binary to recover the Grapeboard's™ primary flash. It also holds unique board configuration data, such as MAC addresses and manufacturing data. Contact Scalys for more information regarding the rescue flash memory.

5

Installing and updating images

5.1 Flashing/updating the RFS on the SD card

By default the Grapeboard™ will attempt to boot the Linux kernel from an SD card. Without modification it requires the first partition to be formatted into a *ext4* filesystem. You can accomplish this with, for instance, the *fdisk+mkfs.ext4* command line tools or the *gparted* tool.

Mount the SD card and go to your flexbuild folder and enter the following commands:

```
cd build/images/ubuntu_xenial_arm64_rootfs.d/

sudo rsync -avxHAWX --progress ./ </mount/location/of/sdcard/>
#This will take several minutes depending on your system and sdcard.

sync
```

Wait for complete synchronisation before unmounting the SD card. Alternatively, if we have a compressed RFS image we can also simply extract the file onto the empty prepared SD card.

5.2 Updating partitions on the primary flash

The default U-boot image on the Grapeboard™ has a set of environment variables to update the partition data in the primary flash memory from external sources. The default supported sources are:

tftp
Refer to section 3.1 to setup the TFTP server.

mmc
By default the variables expect the first partition on the SD card to be formatted with an *ext4* filesystem.

usb
By default the variables expect the first partition on the USB memory stick to be formatted with a *fat32* filesystem.

If the updated files are put in a subdirectory on the source then you have to ensure that the '*update_files_path*' U-boot environment variable is matched correctly. By default this variable is set to '.'. It can be modified using the 'editenv' command in U-boot.
Updating the *u-boot* target partition requires the file named '*u-boot-with-pbl.bin*' to be available on the chosen file source. Similarly, the *ppa* target partition requires the file named '*ppa.itb*', and the *pfe* target partition requires the file named '*pfe_fw_sbl.itb*'[8].

Each of these variables can be executed using the following command:

```
run update_<source>_<target_partition>_qspi_nor
```

These variables can be modified to change the partition index or filesystem type using the U-boot *editenv <variable>* command.

5.3 Recovering from the rescue flash

The primary flash memory data may become corrupted during usage. The Grapeboard™ therefore includes a back-up U-boot image on a read-only rescue flash memory, that may be used to write correct data onto the primary flash memory. To boot from this rescue flash the following two steps should be performed:

1. Connect the Grapeboard™ to your host PC and open the terminal application.
2. Press and hold switch *'S2'* on the Grapeboard™.
3. Power-up (or reset with switch *'S1'*) the Grapeboard™.
4. Release switch *'S2'* once U-boot prints the message: *'Please release the rescue mode button (S2) to enter the recovery mode'*

After performing these steps the user should be able to program the primary flash memory using the commands described in the previous section 5.2.

6

Modifying the Reset Configuration Word

The LS1012A processor requires a valid Reset Configuration Word RCW in order to be successfully initialized at boot time. The current revisions of the processor merely support reading the RCW from two sources, namely an external QSPI flash device and an internal hardcoded source. The Grapeboard™ by default selects the QSPI device as the RCW source. Refer to XXXXXXXX to learn how to use the hardcoded source.

The RCW is a set of 64 bytes, which define how the LS1012A and its interfaces are configured. The PBL reads these 64 bytes along with any appended Pre-Boot Instructions PBI from the chosen source.

The Grapeboard™ by default comes with an RCW configuration with the following main features:

- SATA configuration on the M.2. port.
- I2C and DSPI configured on the 26-pins header.

When the two default RCW options are not sufficient we can modify it using the following steps.

1. Download and install CodeWarrior Development Studio for QorIQ LS series - ARM V8 ISA (modifying the PBL binary using the included PBL configuration tool does not require a paid license).
2. Create a QCVS project for the LS1012A[1] with a default PBL component and make sure to set its RCW source to 'Quad SPI(QSPI)' in the properties tab.
3. Import one of the existing PBL binaries from the u-boot repository, namely: 'grapeboard/u-boot/board/scalys/grapeboard/PBL_0x##_0x0#_800_250_1000.bin'.
4. Modify any desired settings (at your own risk!).
5. Generate the new file.
6. Copy the generated 'PBL_swapped_CRC_not_swapped.bin' file to u-boot.
7. Overwrite the old binary or use the u-boot 'make menuconfig' command to select your custom file.
8. Refer to section 2.3 in order to build a new U-boot binary with the updated PBL file.

[1] At the time of writing, an LS1012A rev. 2 project had to be used because the PBL tool for a rev. 1.0 project forces bit 188 of the RCW to 0, which should actually be forced to 1 in order to boot the LS1012A rev.1 successfully. **Make sure you don't modify any bits that aren't officially supported by your LS1012A revision!**

7

Frequently asked questions

7.1 What is the default login of the LSDK/QorIQ SDK?

For the LSDK built RFS the default login is *root* with the password *root*. For the QorIQ SDK RFS the default login is simply *root*. It is recommended to change the password after your initial login!

7.2 How to connect to the Grapeboard?

7.2.1 Serial port connection

Connect the Grapeboard™ to your host PC with the micro usb port. Verify that a serial device has been successfully added. Note that for Windows 10 based hosts you have to manually install the drivers from: https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers. Using your favourite terminal program, e.g. PuTTY for Windows or Minicom/Kermit for Linux, you have to open the connection with the serial device with the following settings: *115200 8N1 with no flow control*.

7.2.2 SSH connection

There are several ways to achieve this, but for simplicity we will chose a setup without a DHCP server in the network. Note that by default this connection does not allow access to the U-boot bootloader.

Using the serial connection described subsection 7.2.1 you should configure the ethernet interface on your Grapeboard™ with a static IP in the address range of your dedicated network port on your host PC. To do this we can add the following lines in the *'/etc/network/interfaces'* file:

```
auto eth0
allow-hotplug eth0
iface eth0 inet static
address 192.168.1.100
netmask 255.255.255.0
```

Now configure the network interfaces of your host PC with the 192.168.1.0 address and connect it directly to the Grapeboard™. Verify that it works by pinging the Grapeboard's™ address. Note that it may takes several seconds for the Grapeboard™ to boot completely before the SSH service become available. Once successfully connected we can open the SSH connection using PuTTY (under Windows) or the SSH command under Linux with the chosen Grapeboard's™ address.

7.3 The ethernet interfaces of the PFE are not working?

The PFE on the LS1012A processor requires firmware to operate. For U-boot it requires the *'pfe_fw_sbl.itb'* image (see section 5.2 on how to program it) and under Linux the *'ppfe_class_ls1012a.elf'* and *'ppfe_tmu_ls1012a.elf'* files in the *'/lib/firmware/'* directory. These files can be retrieved from the LSDK github page[6]. **Note that it is required to execute the command *'pfe stop'* in U-boot**

before booting a Linux kernel. This has already been included for the default boot command(s) of the Grapeboard™.

7.4 Bluetooth can't find any devices?

The default bluetooth kernel driver doesn't initialize the RF-component of the WiFi/BT module properly. You first need to initialize it through the WiFi interface, e.g.:

```
ifconfig <wlxXXXXXXXXXXXX> up
```

A

References

[1] *SDK QorIQ online manual*. [Online]. Available: http://cache.nxp.com/files/infocenter/QORIQSDK_INFOCTR.html

[2] *LSDK online manual*. [Online]. Available: https://freescale.sdlproducts.com/LiveContent/web/pub.xql?c=t&action=home&pub=QorIQ_LSDK&lang=en-US

[3] *QorIQ© Layerscape 1012A Low Power Communication Processor*. [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/qoriq-layerscape-arm-processors/qoriq-layerscape-1012a-low-power-communication-processor:LS1012A

[4] *Yocto project*. [Online]. Available: https://www.yoctoproject.org/

[5] *NXP supporting information on LSDK in comparison to QorIQ Linux SDK*, 2018. [Online]. Available: https://www.nxp.com/docs/en/supporting-information/DN-LSDK-Introduction.pdf

[6] *Layerscape Software Development Kit download link (requires registration*. [Online]. Available: https://www.nxp.com/support/developer-resources/run-time-software/linux-software-and-development-tools/layerscape-software-development-kit:LAYERSCAPE-SDK?tab=Design_Tools_Tab

[7] *Layerscape Software Development Kit github component page*. [Online]. Available: https://lsdk.github.io/components.html

[8] *Layerscape Software Development Kit github page*. [Online]. Available: https://lsdk.github.io/

B

List of Acronyms

BSP Board Support Package
LSDK Layerscape Software Development Kit
PBI Pre-Boot Instructions
PBL Pre-Boot Loader
RFS Root File System
RCW Reset Configuration Word
SDK Software Development Kit
PFE Packet Forwarding Engine
PPA Primary Protected Application