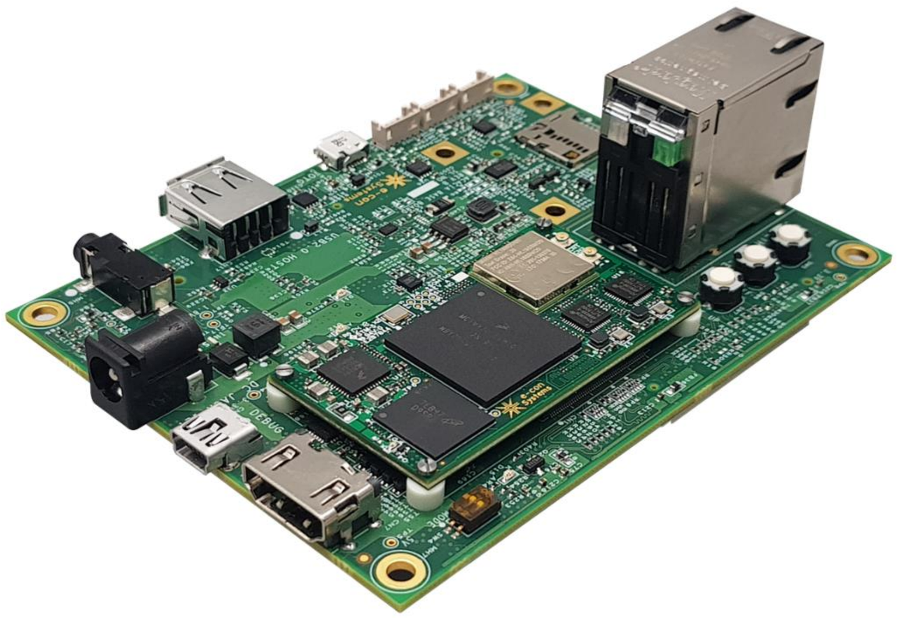
Acacia

Linux Build Guide



Version 1.0

e-con Systems

5/7/2018

Disclaimer

e-con Systems reserves the right to edit/modify this document without any prior intimation of whatsoever.

Contents

Introduction

This document guides you to build bootloader, kernel and rootfs for eSOMiMX7 Acacia board. The build procedure is minimized such that you can complete the whole build using 3 to 4 commands. You need to set up required packages for build. Make sure to follow each and every step in this document, skipping a step creates problem during build.

The following table lists the packages and versions of the various sources.

Table 1: Packages and Version

Source	Package	Version
Poky	Poky-morty	2.2
Toolchain	arm-poky-linux-gnueabi	GCC 6.2.0
Bootloader	U-Boot	2017.03
Kernel	Linux	V4.9.11

Prerequisites

You need to have the basic knowledge of Linux commands, Linux BSP, Git repository and Yocto Poky project.

The build system and package requirement for starting the build process is explained in the following sections.

Build System Requirements

The build system requirements are as follows:

- Minimum PC Requirements: 2.0 GHz CPU, 4GB RAM, 120GB Free Storage Space
- OS Requirement: Ubuntu 14.04 or 16.04

Packages to Install

To successfully build the Yocto project in a Linux host machine, you need to install the required packages and utilities.

To install the required packages and utilities, follow these steps:

1. Run the following command to install essential utilities for Yocto project host package.

```
$ sudo apt-get install gawk wget git-core diffstat
unzip texinfo gcc-multilib build-essential chrpath
socat libsdl1.2-dev
```

2. Run the following command to install additional host packages for Ubuntu 14.04/16.04.

```
$ sudo apt-get install libsdl1.2-dev xterm sed cvs
subversion coreutils texi2html docbook-utils python-
pysqlite2 help2man make gcc g++ desktop-file-utils
libgl1-mesa-dev libglu1-mesa-dev mercurial autoconf
automake groff curl lzop asciidoc xclip
```

3. Run the following command to install u-boot-tools package on Ubuntu 14.04.

```
$ sudo apt-get install u-boot-tools
```

4. Run the following commands to install Repo tool.

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-
repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=~/bin:$PATH
```

Local Git Configuration

This configuration can be skipped if you are already using Git tool actively in your development system.

To configure name and e-mail variable for Git, you need to run the following commands with your name and e-mail ID.

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
```

Run the following command to verify the saved configuration.

```
$ git config –list
```

You can view the entered user name and email ID.

GitLab Configuration

The source code for eSOMiMX7 Acacia is maintained in Global GitLab. You need to configure the SSH key in GitLab to access source code from your development system.

Note: You cannot push or pull project code through SSH until you add the SSH key to your GitLab profile.

Adding the SSH key involves the following process:

1. Generating the SSH key in development system.
2. Adding the generated key in GitLab.

Generating SSH Key in Development System

To generate the SSH key in development system, follow these steps:

1. Run the following command with your e-mail id to generate a new SSH key pair.

   ```
   $ ssh-keygen -t rsa -C "your.email@example.com" -b
   4096
   ```

2. Enter a file path to save your SSH key pair.

   Note: If you do not already have a SSH key pair, use the suggested path by pressing enter.

3. Enter a password to secure your SSH key pair.

   It is a best practice to use a password for an SSH key pair, but it is optional, so you can skip creating a password.

4. Run the following xclip command to copy the public SSH key to clipboard.

   ```
   $ xclip -sel clip < ~/.ssh/id_rsa.pub
   ```

You need to paste the copied SSH key in Key box as described in the Adding SSH Key in GitLab section.

Adding SSH Key in GitLab

To add the SSH key in GitLab, follow these steps:

1. Navigate to https://gitlab.com/profile and login with your credentials.
2. On the **User Settings** tab, click **SSH Keys**.
3. Paste the SSH Key which is already copied in clipboard in **Key** box.
4. Enter a title for the Key in **Title** box.
5. Click **Add Key** to add the key.

Note: Make sure that the SSH key settings is properly configured using the following SSH command in your development system.

```
$ ssh -T git@gitlab.com
```

You will get a welcome message from GitLab with your Name stored in GitLab account.

Build Basics

Building an image from the source package involves the following steps:

1. Downloading Source from GitLab using Repo Tool
2. Configuring for Build
3. Compiling the Source Package

You need to understand some basics of Yocto build before proceeding further, click the following links to get some basic insights about Yocto build.

https://www.yoctoproject.org/docs/2.2.3/yocto-project-qs/yocto-project-qs.html

https://www.yoctoproject.org/docs/2.2/bitbake-user-manual/bitbake-user-manual.html

Downloading Source from GitLab using Repo Tool

The source code for the eSOMiMX7 Acacia is maintained in global GitLab (http://www.gitlab.com) with private access, you need to get the access credential from e-con Systems to access the source code. You can write to techsupport@e-consystems.com to get the source code access. It is recommended to use your official e-mail address to get access to the source code.

Configuring for Build

After downloading the source code, you need to configure environment for the build and you need to select the distribution and machine corresponding to your target board. The details of the different distribution and machine configuration are listed in the following tables.

Table 2: MACHINE Configurations

MACHINE Name	Machine Description
esomimx7d-2gb	Machine name for eSOMiMX7 Dual SOM with 2GB RAM.
esomimx7d-1gb	Machine name for eSOMiMX7 Dual SOM with 1GB RAM.
esomimx7s-2gb	Machine name for eSOMiMX7 Solo SOM with 2GB RAM.
esomimx7s-1gb	Machine name for eSOMiMX7 Solo SOM with 1GB RAM.
esomimx7d-2gb-epdc	Machine name for eSOMiMX7 Dual SOM with 2GB RAM and supports EPDC display.
esomimx7d-1gb-epdc	Machine name for eSOMiMX7 Dual SOM with 1GB RAM and supports EPDC display.

Table 3: DISTRO Configurations

DISTRO Name	Description
esomimx7-fb	Supports Qt5 on a frame buffer backend

esomimx7-wayland	Supports Qt5 Weston Wayland image
esomimx7-xwayland	Supports X11 image with Qt5 support
esomimx7-x11	Supports Xwayland image with Qt5 support

Compiling the Source Package

Compiling the source package is performed using bitbake command. Bitbake is a build engine that follows recipes in a specific format in order to perform sets of tasks. Bitbake is a core component of the Yocto project. You will get further details on compiling the source in later sections.

Building Source Code

Acacia BSP build starts with downloading required source and then build it using bitbake tool.

To download, configure, and build, follow these steps:

1. Run the following commands to create a new directory for eSOMiMX7 build.

   ```
   $ mkdir esomimx7-release-bsp
   $ cd esomimx7-release-bsp/
   ```

2. Run the export command to export the release version you want to build. You will get the release version details in release notes.

   For example,

   ```
   $ export RELEASE_VERSION="esomimx7-L4.9.11_1.0-rc1"
   ```

3. Run the repo init and repo sync commands to download the source code.

   ```
   $ repo init -u git@gitlab.com:esomimx7-linux-
   release/esomimx7_linux_release_manifest.git -b
   $RELEASE_VERSION
   $ repo sync
   ```

Note:

- Repo is a tool built on top of Git that makes it easier to manage projects that contain multiple repositories, which do not need to be on the same server. Repo complements very well the layered nature of the Yocto Project, making it easier for users to add their own layers to the BSP.
- This repo sync will take some time depending on the internet speed. When this process is completed, the source code is checked out into the directory esomimx7-release-bsp or sources. If errors occur during repo initialization, try deleting the repo directory and try running the repo initialization command again.

The esomimx7-setup-release.sh script is also downloaded in esomimx7-release-bsp directory along with source code, after repo sync.

You can use esomimx7-setup-release.sh script to configure for particular MACHINE and DISTRO, the syntax for using this script is shown below.

```
$ DISTRO=<distro name> MACHINE=<machine name> source
esomimx7-setup-release.sh -b <build dir>
```

The sample configuration for your reference are as follows:

For eSOMIMX7 Dual with 1 GB RAM

```
$ DISTRO=esomimx7-x11  MACHINE=esomimx7d-1gb source
esomimx7-setup-release.sh -b out_dual-1gb
```

For eSOMIMX7 Solo with 1 GB RAM

```
$ DISTRO=esomimx7-x11  MACHINE=esomimx7s-1gb source
esomimx7-setup-release.sh -b out_solo-1gb
```

Refer to the *Configuring for Build*

section, to know about the DISTRO and MACHINE variables in eSOMiMX7. This is a one-time step for creating and configuring build directory. For consecutive build setup of same build directory, you must use setup environment script which will be explained as separate note.

After completing the environment configuration, you can start the build using bitbake with image name as parameter.

Run the following sample command to build entire yocto image for Acacia.

```
$ bitbake esomimx7-image-gui
```

When bitbake is completed successfully, you will get all the required binaries to flash and boot. The details about the generated binaries and its location are provided in the next section of this document.

Note:

The procedure mentioned above is only for a fresh build, on consecutive builds you must not do all these steps. You need to run the following command to setup the environment and start the build using bitbake command.

```
$ source setup-environment <build-dir>
```

When a new terminal window is opened, or the machine is rebooted after build directory is setup then the setup environment script must be used to setup the environment. You must not use esomimx7-setup-release.sh after setting up the build directory.

Supported Images

The following table lists the images supported in eSOMiMX7 Acacia build environment.

Table 4: Images supported in eSOMiMX7 Acacia Build Environment

Images supported	Description of image
esomimx7-image-gui	To build an eSOMiMX7 image with GUI featured packages.
esomimx7-image-console	To build an eSOMiMX7 console-only image that fully supports the target device hardware.
esomimx7-image-mfgtool-initramfs	To build firmware for MfgTool. This can be used only for MfgTools temporary firmware as it is very minimal image.

Build Output

On successful build of all the images and packages, deployed images and binaries are present in the specified path.

The following table lists the path of the deployed images and binaries.

Table 5: Deployed Images and Binaries Output Path

Machine	Build Output Path
esomimx7d-1gb	<build_dir>/tmp/deploy/images/esomimx7d-1gb
esomimx7s-1gb	<build_dir>/tmp/deploy/images/esomimx7s-1gb
esomimx7d-2gb	<build_dir>/tmp/deploy/images/esomimx7d-2gb
esomimx7s-2gb	<build_dir>/tmp/deploy/images/esomimx7s-2gb
esomimx7d-2gb-epdc	<build_dir>/tmp/deploy/images/esomimx7d-2gb-epdc
esomimx7d-1gb-epdc	<build_dir>/tmp/deploy/images/esomimx7d-1gb-epdc

The output directory will be having lot of binaries. The following table lists the information about binaries which you require.

Table 6: Description of Binaries

Image Built	Output Files	Description
Any Image	u-boot.imx	Bootloader image with IVT configuration.
	zImage	Linux Kernel Image.
	zImage-imx7x-acacia.dtb	Device tree file for Acacia board.
	zImage-imx7x-acacia-m4.dtb	Device tree file when M4 core is enabled in Acacia.
	initramfs.cpio.gz.uboot	Init Ramfs Image.
esomimx7-image-gui	esomimx7-image-gui-esomimx7x-1gb.sdcard	SDImage for esomimx7-image-gui image.
	esomimx7-image-gui-esomimx7x-1gb.tar.bz2	RootFS tar ball for esomimx7-image-gui image.
esomimx7-image-console	esomimx7-image-console-esomimx7x-1gb.sdcard	SDImage for esomimx7-image-console image.
	esomimx7-image-console-esomimx7x-1gb.tar.bz2	RootFS tar ball for esomimx7-image-console image.

Where, x is s for Solo and d for Dual.

Refer to the *e-con_Acacia_Linux_Yocto_Prebuilt_Binaries_UserManual.pdf*, for more details about deploying images into SD, eMMC, or NAND.

Build Directory Details

The basic information about the directories present inside the build folder are explained in this section.

You need to know the exact content of the build directory since the build directory is created using scripts.

The following table lists the contents of build directory and its descriptions.

Table 7: Description of Build Directory

Directory	Description
Conf	Directory contains configuration files such as local.conf and bblayers.conf.
Downloads	Directory contains downloaded upstream source tarballs.
sstate-cache	Directory contains the shared state cache. Bitbake has the capability to accelerate builds based on previously built output. This is done using shared state files which can be thought of as cache objects and this option determines where those files are placed.
Tmp	The OpenEmbedded build system creates and uses this directory for all the build system's output. The TMPDIR variable points to this directory.

Additional Bitbake Commands

This section is required only for developers who needs to customize the build for the custom board. This sections also guides you to build individual packages using Bitbake command.

The following sections describes the different options of bitbake command.

Installing Toolchain

Yocto is a Linux distribution creator and intended to be an image builder, a rootfs creator. So, Yocto itself must not be used to develop a new package. Although, Yocto can help creating the environment for development such as meta-toolchain or Eclipse ADT.

Run the following command to build the tool chain image from the build environment.

```
$  bitbake meta-toolchain
```

The toolchain will be installed on your host machine which can be used to build any source code for the target system.

The following table list the toolchain installation path according to the OS or System support.

Table 8: Toolchain Installation Path for OS

OS	Toolchain Installation Path
64-bit	<BUILD_DIR>/tmp/sysroots/x86_64-linux/usr/bin/arm-poky-linux-gnueabi/
32-bit	<BUILD_DIR>/tmp/sysroots/i686-linux/usr/bin/arm-poky-linux-gnueabi/

Building Kernel

You can build kernel package separately using the bitbake command. The kernel source is downloaded in **esomimx7-release-bsp/source/esomimx7-kernel/** path during Repo command.

Kernel recipe is prepared in such a way that bitbake builds the kernel from a particular branch and the branch format is given below.

<RELEASE_VERSION>_local

where, RELEASE_VERSION is defined during repo initialization step.

For example,

esomimx7-L4.9.11_1.0-rc1_local

Note: You can modify the source code and make sure to locally commit the changes in the same branch mentioned above.

Run the following commands to perform the clean build of kernel image.

```
$ bitbake -c cleansstate linux-esomimx7
$ bitbake linux-esomimx7
```

The bitbake command builds and installs driver modules as well, you can get the details about driver module file.

The following table lists the build output path of kernel for each machine configuration.

Table 9: Build Output Path of Kernel

Machine	Kernel Image Path	Driver Module File
esomimx7d-1gb	tmp/work/esomimx7d-1gb-poky-linux-gnueabi/linux-esomimx7/4.9.11-r0/linux-4.9.11/arch/arm/boot/zImage	4.9.11-r0-esomimx7d-1gb-XXXXXX.tgz
esomimx7s-1gb	tmp/work/esomimx7s-1gb-poky-linux-gnueabi/linux-esomimx7/4.9.11-r0/linux-4.9.11/arch/arm/boot/zImage	4.9.11-r0-esomimx7s-1gb-XXXXXX.tgz
esomimx7d-2gb	tmp/work/esomimx7d-2gb-poky-linux-gnueabi/linux-esomimx7/4.9.11-r0/linux-4.9.11/arch/arm/boot/zImage	4.9.11-r0-esomimx7d-2gb-XXXXXX.tgz
esomimx7s-2gb	tmp/work/esomimx7s-2gb-poky-linux-gnueabi/linux-esomimx7/4.9.11-r0/linux-4.9.11/arch/arm/boot/zImage	4.9.11-r0-esomimx7s-2gb-XXXXXX.tgz
esomimx7d-2gb-epdc	tmp/work/esomimx7d-2gb-epdc-poky-linux-gnueabi/linux-esomimx7/4.9.11-r0/linux-4.9.11/arch/arm/boot/zImage	4.9.11-r0-esomimx7d-2gb-epdc-XXXXXX.tgz
esomimx7d-2gb-epdc	tmp/work/esomimx7d-1gb-epdc-poky-linux-gnueabi/linux-esomimx7/4.9.11-r0/linux-4.9.11/arch/arm/boot/zImage	4.9.11-r0-esomimx7d-1gb-epdc-XXXXXX.tgz

Where, XXXXXX is the date and time of creation.

Building U-Boot

The U-Boot source is downloaded during repo command in the **esomimx7-release-bsp/source/esomimx7-uboot**/ path. The U-Boot recipe is prepared in such a way that bitbake builds the U-Boot from a particular branch and the branch format is given below.

<RELEASE_VERSION>_local

Where, RELEASE_VERSION is defined during repo initialization step.

For example,

esomimx7-L4.9.11_1.0-rc1_local

Note: You can modify the U-Boot source code and make sure to locally commit the changes in the same branch mentioned above.

Run the following commands to perform the clean build of kernel image.

```
$ bitbake -c cleansstate u-boot-esomimx7
$ bitbake u-boot-esomimx7
```

The following table lists the build output path of U-Boot for each machine configuration.

Table 10: Build Output Path of U-Boot

Machine	U-boot Image Path
esomimx7d-1gb	tmp/deploy/images/esomimx7d-1gb/u-boot.imx
esomimx7s-1gb	tmp/deploy/images/esomimx7s-1gb/u-boot.imx
esomimx7d-2gb	tmp/deploy/images/esomimx7d-2gb/u-boot.imx
esomimx7s-2gb	tmp/deploy/images/esomimx7s-2gb/u-boot.imx
esomimx7d-2gb-epdc	tmp/deploy/images/esomimx7d-2gb-epdc/u-boot.imx
esomimx7d-1gb-epdc	tmp/deploy/images/esomimx7d-1gb-epdc/u-boot.imx

Building Individual Packages

Bitbake provides option to build packages in rootfs individually.

Run the following command to lists all packages that can be built in the current environment.

```
$  bitbake -s
```

Based on the package list, you can build a particular package using following command.

```
bitbake <PACKAGE_NAME>
```

For example,

```
$  bitbake bluez5
```

The following table lists the bitbake commands which will be helpful for your development purpose.

Table 11: Description of Bitbake Commands

Command	Description
bitbake <image>	Bake an image.
bitbake <image> -k	Bake an image and continue building even errors are found in the tasks execution.
bitbake <image> -D	Bake an image in debug mode. You can specify this more than once to increase the debug level.
bitbake <image> -v	Bake an image in verbose mode to output more log message data to the terminal.

bitbake <package> -c <task>	Execute a particular package's task.
	Default Tasks names:
	fetch, unpack, patch, configure, compile, install, package, package_write and build.
	Example: To (force) compiling kernel, use following commands.
	$ bitbake linux-esomimx7 -f -c compile
	$ bitbake linux-esomimx7
	Check if certain package is present on current Yocto Setup.

bitbake -s | grep <pkg>	Check if certain package is present on current Yocto Setup.
bitbake-layers show-recipes "*-image-*"	Show possible images to bake. Without "*-images-*", it shows ALL recipes.
bitbake -g <image> && cat pn-depends.dot | grep -v -e '-native' | grep -v digraph | grep -v -e '-image' | awk '{print $1}' | sort | uniq	Show packages of image.
bitbake -g <pkg> && cat pn-depends.dot | grep -v -e '-native' | grep -v digraph | grep -v -e '-image' | awk '{print $1}' | sort | uniq	Show package's dependencies.
bitbake -c cleansstate	Cleans all the states except downloads.
bitbake -c cleanall	Cleans everything along with downloads.

Troubleshooting

In this section, you can view the list of commonly occurring issues and their troubleshooting steps.

What can I do when source download or fetch fails during build?

This issue happens due to network failure. You can retry the image build again to resolve this issue.

For example, run the following command to build the image.

```
$ bitbake esomimx7-image-gui
```

What can I do when repo init command failed with *Please make sure you have the correct access rights and the repository exists* error message?

This issue happens due to incorrect or missing SSH key. Make sure you have configured SSH key in your GitLab account correctly.

FAQ

1. **How do I get BSP source code access?**

   You can refer to *Downloading Source from GitLab using Repo Tool*

   section. Please write to techsupport@e-consystems.com, to get the source code access.

2. **What is the build PC requirement and Ubuntu version supported for building the source?**

   You can refer to *Build System Requirements* section, to know about build PC requirement and supported Ubuntu versions for building the source. It is recommended to use Ubuntu 14.04 or Ubuntu 16.04

3. **Is it possible to modify the kernel source and build the kernel alone?**

   Yes, it is possible to build individual components. Please refer to the *Building Kernel*

   section, to build the kernel.

What’s Next?

You can refer to *Acacia Linux Yocto Prebuilt Binaries User Manual*, to get information about updating the binaries build.

Glossary

DTB	Device Tree Blob
eMMC	Embedded MultiMedia Controller
GUI:	**G**raphical User Interface
OTG	On-The-Go
RAM	Random Access Memory
SOM	System on Module

Support

Contact Us

If you need any support on Acacia product, please contact us using the Live Chat option available on our website - https://www.e-consystems.com/

Creating a Ticket

If you need to create a ticket for any type of issue, please visit the ticketing page on our website - https://www.e-consystems.com/create-ticket.asp

RMA

To know about our Return Material Authorization (RMA) policy, please visit the RMA Policy page on our website - https://www.e-consystems.com/RMA-Policy.asp

General Product Warranty Terms

To know about our General Product Warranty Terms, please visit the General Warranty Terms page on our website - https://www.e-consystems.com/warranty.asp

Revision History

Rev	Date	Description	Author
1.0	07 May 2018	Initial Draft	SOM Team