

**BSP User Guide**

**For**

**Rx3 Platforms**

Document Reference: BSP User Guide for Rx3 Platforms
Document Issue: 0.2

Contents

[1. Introduction 3](#_Toc516660927)

[2. Environment Setup 3](#_Toc516660928)

[2.1 Embedded Linux Components 3](#_Toc516660929)

[2.2 Installation of the Embedded Linux build components 3](#_Toc516660930)

[2.3 Development Machine Setup 5](#_Toc516660931)

[3. Building with Binary Root Filesystem Distributions 6](#_Toc516660932)

[3.1 Creating the SD Card Image 6](#_Toc516660933)

[3.2 Writing the SD Card Image 6](#_Toc516660934)

[4 Building with Buildroot 7](#_Toc516660935)

[4.1 Buildroot introduction 7](#_Toc516660936)

[4.2.1 Quickboot demo, with MPlayer support 7](#_Toc516660937)

[4.2.2 QT5 and BlueZ 5 7](#_Toc516660938)

[4.3 Buildroot outputs 7](#_Toc516660939)

[5. Building with Yocto 8](#_Toc516660940)

[5.1 Install Yocto BSP Sources 8](#_Toc516660941)

[5.2 Install BCT Yocto Layer 8](#_Toc516660942)

[5.3 Add BCT Layer to Yocto Configuration 8](#_Toc516660943)

[5.4 Configure Build for Target Platform 9](#_Toc516660944)

[6. Custom Build 10](#_Toc516660945)

[6.1 Kernel Build 10](#_Toc516660946)

[6.2 Uboot Build 11](#_Toc516660947)

[6.3 Generating SD card image 11](#_Toc516660948)

# 1. Introduction

The content of this document provides information required to start building Linux operating systems for the Rx3 platforms (currently RE3 and RM3) using the Rx3 BSP. It covers:

* The tools and components required for building a Linux operating system
* How to install the build components
* How to compile the U-Boot boot loaders stand alone
* How to compile the Linux Kernel stand alone
* How to setup a root filesystem using Freescale (NXP) demo image
* How to setup a root filesystem using Yocto build system
* How to setup a root filesystem using Ubuntu 14.04 Core
* How to build a root filesystem including using buildroot

# 2. Environment Setup

## 2.1 Embedded Linux Components

The components involved in a typical Embedded Linux system targeting the ARM architecture are:

1. Bootloader (Typically uboot)
2. Linux Kernel
3. Root filesystem.

U-boot 2015.04 was ported to provide the bootloader functionality for the Rx3 platforms.

Linux kernel 4.1.15 was ported to be compatible with the Rx3 platforms.

Pre-built Freescale (NXP) and Ubuntu Core demo images are provided for quick start development. Instructions are also provided for building images from source using either Yocto or buildroot frameworks. Custom builds are also possible.

The Rx3 software components above have all been tested to compile using an Ubuntu 16.04 LTS development machine. The following sections detail the necessary steps for setting up the environment on the development machine; they assume the development machine (either physical or virtual) has already been created and has Ubuntu 16.04 installed. The development machine must have at least 50GB of free disk space.

## 2.2 Installation of the Embedded Linux build components

Create a directory called “embedded” in the root of the filesystem and give it unrestricted access. Issue the following commands to achieve this:

cd /

sudo mkdir embedded
**sudo chmod 777 embedded**

**cd embedded**

Copy the latest Rx3 BSP into the /embedded directory. The BSP is can be downloaded as follows:

sudo wget [http://dl.bluechiptechnology.com/dl/rx3/software/Rx3BSPv4.1.15.tar.bz2](http://dl.bluechiptechnology.com/dl/tm1/software/tm1linuxv102.tar.bz2)

sudo wget [http://dl.bluechiptechnology.com/dl/rx3/software/Rx3BSPv4.1.15.tar.bz2.md5](http://dl.bluechiptechnology.com/dl/tm1/software/tm1linuxv102.tar.bz2.md5)

Check that the integrity of the download is ok by issuing the following command:

md5sum -c Rx3BSPv4.1.15.tar.bz2.md5

Extract the tar ball by issuing the command:

sudo tar xvjf Rx3BSPv4.1.15.tar.bz2

Once extracted the build components will be laid out in the following structure on the development machine. The first directory (“embedded”) is the folder created in the root of the filesystem.

|  |  |  |
| --- | --- | --- |
| Directory | Directory | **Description** |
| /embedded/toolchains | gcc-linaro-4.9.4-2017.01-x86\_64\_arm-linux-gnueabihf | Prebuilt cross compiling tool chain based on GCC 4.9, for building ARMhf software.  |
| /embedded/projects/rx3 | linux-rx3 | 4.1.15 kernel source code with configuration for BCT Rx3 platforms |
| sdcard | Contains scripts and file system overlays for creating bootable disk images using root file systems contained in rootfsimages directory. Images can be copied onto SD card to boot the target Rx3 platform. |
| u-boot-rx3 | Source code for the U-boot boot loader including configuration for BCT Rx3 platforms |
| rootfsimages | Directory containing prebuilt root filesystems for Rx3 platforms including:fsl-image - official demo image from Freescale(NXP) for imx6 based platforms.Ubuntu Core 14.04 - minimal Ubuntu root file system aimed at embedded devices. Can be extended with apt-get to install extra packages and features. |
| x | buildroot-2017.11.2. | Buildroot 2017.11 release with configurations for building rootfs images, linux kernel, and linux device tree blobs for Rx3 platforms. |
|  | yocto | Yocto FSL BSP release with configurations for building rootfs images, linux kernel, and linux device tree blobs for Rx3 platforms. |
| /embedded/projects/genimage |  | Contains the source code and build scripts for the genimage application. |

## 2.3 Development Machine Setup

The following packages are known to be required on an Ubuntu 16.04 development machine to successfully build the components:

sudo apt-get update

sudo apt-get install build-essential

sudo apt-get install u-boot-tools

sudo apt-get install flex

sudo apt-get install bison

sudo apt-get install lzop

sudo apt-get install ncurses-dev

sudo apt-get install gcc-multilib

sudo apt-get install gawk chrpath texinfo

sudo apt-get install genext2fs

sudo apt-get install libconfuse-common

sudo apt-get install libconfuse-dev

sudo apt-get install dh-autoreconf

sudo apt-get install git

The tool genimage must be compiled when the BSP is first installed as this is subsequently used by the build scripts to create SD card images. To build genimage do the following:

cd /embedded/projects/genimage
./autogen.sh
./configure ./CFLAGS='-g -O0' --prefix=/usr
make

# 3. Building with Binary Root Filesystem Distributions

The directory /embedded/projects/rx3/rootfsimages contains a number of pre-built root file systems that may be used to quickly create a bootable image. The following root file systems are currently available:

* Official Freescale demo image (fsl-rootfs); this is based on Yocto Krogoth release providing GUI and QT5 X11 support. Also supports iMX6 hardware accelerated graphics.
* Ubuntu Core 14.04 release (ubuntu-core-rootfs); this is a basic command line release that can be subsequently configured through the use of apt-get. Note that this file system does not contain support for iMX6 hardware accelerated graphics.

## 3.1 Creating the SD Card Image

The following commands will create the required disk image:

cd /embedded/projects/rx3/sdcard

./create-rx3-image <fsl-rootfs|ubuntu-core-rootfs> <re3|rm3>

For example:

./create-rx3-image fsl-rootfs rm3

will create an SD card image suitable for booting the Freescale demo root filesystem on an RM3 platform called fsl-rootfs.rm3.sdcard .

## 3.2 Writing the SD Card Image

Plug an SD card reader into the development machine and insert a suitable uSD card; the card should be at least 2GB. Find out what name the kernel has given the card by issuing the following command:

dmesg | grep "sd\*" | grep "removable"

Assuming the kernel allocated the device "sdf" the following commands will write the image to the uSd card (note that you will require root privileges to use dd):

cd /embedded/projects/rx3/sdcard

sudo dd if=fsl-rootfs.rm3.sdcard of=/dev/sdf

Once the above command has completed the uSD card will be ready to insert into the target platform and allow the target platform to boot.

# 4 Building with Buildroot

## 4.1 Buildroot introduction

Buildroot is a build system that aids the process of building the various components of an Embedded Linux system in a single environment. We think Buildroot is easy to get to grips with, and provides a reasonable amount of package support.

Buildroot 2017.11 is provided in the Linux BSP for Rx3. It contains two sample configurations which build the Linux kernel, and a root filesystem.

## 4.2.1 Quickboot demo, with MPlayer support

This configuration is designed to be small and demonstrate a quick ~3 second boot time. MPlayer is included in the configuration, and is configured to automatically play videos found in the root of a USB flash drive during boot. AVI, and MP4 video formats are supported, and video resolutions must match the target LCD screen resolution.

To build the quickboot configuration issue the following commands.

cd /embedded/projects/rx3/buildroot-2017.11.2

make rx3\_mplayerquickbootdemo\_defconfig

make -j4

## 4.2.2 QT5 and BlueZ 5

This configuration will build a root filesystem containing QT5 libraries, QT5 sample applications, and BlueZ libraries To build this configuration issue the following commands.

cd /embedded/projects/rx3/buildroot-2017.11

make rx3\_qt5\_defconfig

make -j4

## 4.3 Buildroot outputs

After completion of the build an SD card image is left in

/embedded/projects/rx3/buildroot-2017.11.2/output/images/sdcard.img

 This image can be written to a uSD card using the instructions given in section 3.2.

# 5. Building with Yocto

Yocto is the build system supported by NXP for iMX6 platforms. Bluechip Technology (BCT) have created a specific Yocto layer to support the building of a basic image for Rx3 platforms; this section will provide the necessary instructions for creating a Yocto build using the BCT layer.

**Note that Yocto build system prevents builds being performed by root users so the following must be performed by a non-root user.**

## 5.1 Install Yocto BSP Sources

For GIT purposes ensure you have a user name and email address set up; below is an example of how to enter this information (use your own details):

git config --global user.email "engineer@bluechiptechnology.co.uk"
git config --global user.name "BCT Engineer"

To get the BSP you need to have repo installed and use it as follows:
Install the repo utility:

$: mkdir ~/bin

$: curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo

$: chmod a+x ~/bin/repo

Download the Yocto BSP source:

$: PATH=${PATH}:~/bin

$: cd /embedded/projects/rx3/yocto

$: mkdir fsl-community-bsp

$: cd fsl-community-bsp

$: repo init -u https://github.com/Freescale/fsl-community-bsp-platform -b krogoth

$: repo sync

## 5.2 Install BCT Yocto Layer

Add the following entries to .repo/manifest.xml

**<remote fetch="git://github.com/bluechiptechnology" name="bct"/>
<project remote="bct" revision="krogoth" name="meta-bct-rx3" path="sources/meta-bct-rx3"/>**

Now refresh the Yocto source (this will download and install the new layer) using this command:

$: repo sync

## 5.3 Add BCT Layer to Yocto Configuration

Initialise the Yocto configuration using the command (note that you will be asked to read and accept the FSL license before you can continue. If you don't accept the license you will not be able to build the image):

$: source setup-environment build

This creates, and places you in, the working build directory fsl-community-bsp/build
Add the following entry into the file conf/bblayers.conf :

**${BSPDIR}/sources/meta-bct-rx3 \**

## 5.4 Configure Build for Target Platform

For RM3 add the following entry to conf/local.conf :

**MACHINE = "bct-rm3"**

For RE3 add the following entry to fsl-community-bsp/build/conf/local.conf :

**MACHINE = "bct-re3"**

To start a build type the following :

$: bitbake core-image-sato

After successful completion of the build the output image can be found at:

/embedded/projects/rx3/yocto/fsl-community-bsp/build/tmp/deploy/images/bct-re3/core-image-sato-bct-re3.sdcard.gz

This file can be uncompressed and written to a uSD card as detailed in section 3.2.

# 6. Custom Build

It is possible to build the kernel and uboot binaries outside of the build frameworks detailed in the preceding sections. When building is done this way the user must provide a suitable root filesystem. The root file system must be installed in /embedded/projects/rx3/sdcard/rootfs directory.

Before compiling various stand alone Linux components we must set some environment variables. This is to ensure the configuration tools build for the correct architecture and can find the cross compiling tool chain. To make this task simpler a script file is provided to configure the environment for the Rx3 build. Issue the following commands to run the script:

cd /embedded/

source buildenv-4.9

## 6.1 Kernel Build

To compile the kernel we must enter the root of the kernel source tree, make some configuration changes and use make to start the compile. Issue the following commands.

cd /embedded/projects/rx3/linux-rx3

./build.sh

After completing, the compile process will leave a Linux kernel (zImage) at, “./arch/arm/boot/zImage”.

The Rx3 kernel implements the device tree model for configuring a hardware platform. The Rx3 kernel includes two configurations for the various Rx3 platforms:

|  |  |
| --- | --- |
| **Device tree definition** | **Description** |
| bctre3.dtb | RE3 platform |
| bctrm3.dtb | RM3 platform |

build.sh is an example of a script file that simplifies the process of building Rx3 Linux components. Please study these files for an understanding of their purpose.

If changes are required to the kernel configuration the command “make menuconfig” can be used to present a menu based configuration utility for the Linux kernel. If any changes are made using the menuconfig tool, the “./build.sh” command must be re-issued.

Once the kernel has been compiled, the kernel modules must be copied to the root filesystem. Issuing the following command performs this task.

./installmodulesrootfs.sh

You will notice that this particular script installs the kernel modules in the /embedded/projects/rx3/sdcard/rootfs directory.

## 6.2 Uboot Build

U-Boot 2015.04 has been ported to work with Rx3 platforms. Its purpose is to initialise the hardware, and boot a Linux operating system.

The Rx3 platforms are factory initialised with a working bootloader installed in the SPI NOR flash; this bootloader is sufficient to boot the Linux kernel (built as per section 6.1). Therefore it is only necessary to build uboot if customisation of its operation is required. The customised uboot can then be written to the SD card (see below) and the Rx3 will then use this as the bootloader. Note that the bootloader in SPI NOR is not overwritten by this process so the customised uboot must always be resident on the SD card if this is the version to be used.

To build U-Boot for RE3 issue the following commands.

 cd /embedded/projects/rx3/u-boot-rx3

./buildre3.sh

To build U-Boot for RM3 issue the following commands.

 cd /embedded/projects/rx3/u-boot-rx3

./buildrm3.sh

In each case the compiled boot loader is located in /embedded/projects/rx3/sdcard/boot directory.

## 6.3 Generating SD card image

Once the kernel and uboot (if required) have been built the uSD card image can be created as follows:

cd /embedded/projects/rx3/sdcard

./mkrootfs <platform>

where <platform> is one of re3 or rm3. The output image is written to <platform>.sdcard.img which can then be copied to uSD card as per section 3.2.