User Manual

LINUXBUILD User's Manual

Linux build environment for LEON systems

Document: LINBUILD
September 2021, Version 2.2.0b

Cobham Gaisler AB
Kungsgatan 12 | SE-41119 Gothenburg | Sweden
+46 31 7758650 | www.caes.com/Gaisler

Table of Contents

1. LINUXBUILD

1.1. Introduction

This document describes how the LINUXBUILD utility is used to build one or more of the components listed below. The LINUXBUILD utility consists of a graphical configuration utility, configuration files and a set of small Makefile-scripts for building and configuring some of the Linux tools that Cobham Gaisler provides support for.

The LINUXBUILD utility provides is a quick way of getting started with Linux development for the LEON architecture. It ties different components together to build a complete Linux environment. Each component is designed to be used separately from each other or together from LINUXBUILD. One can see LINUXBUILD as an example utility that provides a quick way of getting started with Linux development using the different tools and components. Currently the following components are supported in LINUXBUILD.

- GNU Toolchain - GCC, BINUTILS, GLIBC
- Linux Kernel + LEON Linux patches
- LEON Linux RAM loader (**mklinuximg**)
- Buildroot + LEON patches
- MKPROM2

Settings for standard LEON Linux configurations are available within the LINUXBUILD package and custom configurations can also be created by the user. Predefined configurations can be found in the `gaisler/configs` directory.

1.2. Requirements

- Linux host machine
- SPARC/LEON Linux Toolchain
- MKPROM2 - for creating PROM/FLASH images
- wget
- git
- Internet access

Buildroot requires a number of tools such as bison, flex, msgfmt, makeinfo, etc. please see respective tool's home-page for requirements.

1.3. Download Location

Table 1.1. Download locations

Tool	Download location
LINUXBUILD	http://gaisler.com/anonftp/linux/linux-2.6/linuxbuild/linuxbuild-x.y.z.tar.bz2 [http://gaisler.com/anonftp/linux/linux-2.6/linuxbuild/]
Linux toolchain	http://gaisler.com/anonftp/linux/linux-2.6/toolchains/ [http://gaisler.com/anonftp/linux/linux-2.6/toolchains/]
MKPROM2	http://gaisler.com/anonftp/mkprom2/linux/ [http://gaisler.com/anonftp/mkprom/linux]

1.4. Using the LINUXBUILD GUI

LINUXBUILD uses the KConfig configuration system originally from the Linux kernel. This is what is what is mentioned as the “GUI” in this manual. Support has been added to execute commands from within the configuration interface.

The GUI can be started in a number of different forms. Using xconfig is the recommended configuration interface, the interface that is used in further examples and descriptions and the sub-interface that can be launched to configure Linux and Buildroot.

- **make xconfig** — Qt based GUI (Qt-3/4 libs required)

Provided as is:

- **make gconfig** — GTK based GUI
- **make menuconfig** — ncurses based terminal interface

The Buildroot and Linux configuration can be started as xconfig from within the LINUXBUILD config session. They can also be started using make xconfig in the `linux` and `dist/buildroot` directory respectively, or from the top level as **make linux-xconfig** and **make buildroot-xconfig** respectively. The same goes for gconfig and menuconfig when selecting that as configuration tool.

This manual describes how to use the GUI to set up and use LINUXBUILD. However, most of the actions that can be taken from within the GUI can be executed via makefiles such as `Makefile`, `linux/Makefile`, `dist/buildroot/Makefile` and `boot/mklinuximg/Makefile`.

1.5. Mini Quickstart Guide

Quick guide from zero to default configuration linux RAM image:

1. Download and install Linux toolchain in /opt
2. Download and unpack LINUXBUILD
3. change directory into newly unpacked directory
4. Run make xconfig
5. Click “Install Linux”
6. Double click “Execute Linux installation of latest stable leon-linux”
7. Click yes on popup and wait for installation to finish in separate window
8. Check that installation did not end with error and press return to close window which restarts xconfig
9. Scroll down to the end and click “Step3: build”
10. Double click “Execute make build”
11. Click yes on popup and wait for build to finish in separate window
12. Check that build did not end with error and press return to close window which restarts xconfig
13. Close xconfig

If all went well the following files can be found in `output/images:`

- `image.ram` — RAM image (for loading and running or for feeding to bootloader)
- `image` — virtual address image with symbols (for debug)

1.6. References

RD0	MKPROM2 User's Manual http://www.gaisler.com/doc/mkprom2.pdf
RD1	MKLINUXIMG User's Manual http://www.gaisler.com/anonftp/linux/linux-2.6/doc/mklinuximg-2.x.y.pdf [http://www.gaisler.com/anonftp/linux/linux-2.6/doc/]
RD2	Prebuilt Toolchain http://www.gaisler.com/anonftp/linux/linux-2.6/toolchains/
RD3	LEON Linux User's Manual http://www.gaisler.com/anonftp/linux/linux-2.6/doc/linux-x.y-a.b.pdf [http://www.gaisler.com/anonftp/linux/linux-2.6/doc/]
RD4	The Buildroot user manual http://buildroot.org/downloads/manual/manual.html

2. Installing

After downloading the LINUXBUILD package it is extracted using **tar -xf**.

```
$ tar -xf linuxbuild-x.y.z.tar.bz2
$ cd linuxbuild-x.y.z
```

2.1. Toolchain

Before configuring and using LINUXBUILD the SPARC/LEON Linux toolchain must be installed, unless Buildroot is used to build a toolchain.

Download a toolchain from the Cobham Gaisler web server at http://gaisler.com/anonftp/linux/linux-2.6/toolchains/. Make sure to use a toolchain that does not have newer kernel headers than the kernel that you will be using. For example a sparc-gaisler-linux4.9-x.y toolchain should not be used with main kernel versions lower than 4.9. The x.y part is the sub-version for the toolchain itself. For the specific kernel stable version 4.9.x that is used for kernel headers see the changelog or documentation of the toolchain.

The toolchain can be installed anywhere, but the default installation directory (that various default configurations is set up to use) in in `/opt`. For example for a sparc-gaisler-linux4.9 toolchain the resulting directory is `/opt/sparc-gaisler-linux4.9`. If the toolchain is installed elsewhere, both the Linuxbuild and Buildroot toolchain configurations should be updated accordingly.

2.2. Linux kernel

A Linux kernel package then needs to be installed the first time in the GUI. Best is to used **make xconfig** or **make gconfig**. To install an all default Linux kernel **make -C linux install** can be used.

By default, Linux is installed by cloning the official Linux GIT repository and applying patches therein. If the option "Use tar archive instead of GIT when possible" is selected, a tar archive from kernel.org is downloaded and used as the patch-base.

By default, the latest stable LEON Linux kernel distribution from Cobham Gaisler is automatically downloaded when installing Linux. To install a different LEON Linux kernel distribution, download a LEON Linux kernel package from the Cobham Gaisler website and place it in the `linux/reference` directory, before starting the GUI. Then choose "Install specific leon-linux tar-file".

When the installation is done using the GUI, the installation is started by executing the install command. When double-clicking on "Execute Linux installation" a dialog will pop up asking whether to execute the installation in a new xterm or in the parent shell. Choosing "yes" will open a new window, choosing "no" will execute in the same terminal as the GUI was started and choosing "cancel" will cancel the install.

When installing, upgrading and and permanently loading and saving a configurations, the configurator is restarted automatically so that the configuration change is shown in the configurator properly.

2.3. Buildroot distribution

The lastest LEON Buildroot release at the time of the LINUXBUILD release is preinstalled. To upgrade to a later LEON Buildroot release, see Chapter 5.

In Step 1 in the GUI, one can choose to use no distribution, e.g. to instead provide Linux with a ready made file system image through the Linux configuration. This option can for example be useful when using custom scripts to generate a root file system, of the kernel uses a file system over NFS or located in FLASH/PROM.

2.4. Mklinuximg Linux RAM loader

The latest mklinuximg at the time of the release of LINUXBUILD is preinstalled. This package can be upgraded in Step 1 in the GUI.

2.5. MKPROM2

The mkprom utility is used to generate FLASH/PROM bootable images. It is downloaded from Cobham Gaisler website [http://www.gaisler.com/index.php/downloads/compilers] and installed manually. This is a separate pack-

age and can be used with other LEON tools and Operating Systems than LINUXBUILD. The MKPROM2 utility is found through the PATH environment variable or a custom absolute path set by the configuration.

3. Configuring

After the selected components have been installed or upgraded, LINUXBUILD and each selected component should be configured. This can be done using **make xconfig**, **make gconfig**, or **make menuconfig**. Executable actions works best via **make xconfig** or **make gconfig**.

Note that since each component (LINUXBUILD, Linux, Buildroot) is configured separately it is sometimes needed to set the same configuration option in multiple locations. For example, both LINUXBUILD and Buildroot have separate toolchain configurations.

3.1. Predefined configurations

Prepared LEON configurations can be found in `gaisler/configs`. They can be loaded by LINUXBUILD under "Predefined configurations" in the GUI. In addition, there the current configuration can be saved back into a predefined configuration or saved as a new predefined configuration.

To load a predefined configuration, select an available predefined configuration and double click "Load the selected configuration". The GUI will restart after loading the configuration. Note that loading a configuration overwrites the current configuration of all components.

To save the current configuration back to a predefined configuration, select an available predefined configuration and double click "Save the current configuration back...". To save the current configuration to a new configuration, configure a name for the new configuration (LB_SELECTED_SAVE_CONFIG) and double click "Save the current configuration into a new...". The GUI will restart after saving the configuration.

3.2. Toolchain configuration

Here, a choice can be made to use an external toolchain in PATH, an toolchain with a given path (default with a default path) or using a Buildroot toolchain. The choice here should in general correlate to the Buildroot toolchain configuration. Note that this top level toolchain configuration will set the toolchain used for the Linux kernel and MKLINUXIMG, but will not affect the toolchain configuration of Buildroot. Buildroot has its own toolchain configuration.

3.3. Buildroot configuration

Buildroot is used to build user-space applications, and optionally tool chains. The LINUXBUILD GUI can launch a separate configuration session for Buildroot. Alternatively configuration can be started with e.g. **make buildroot-xconfig** from the top level directory or **make xconfig** from the `dist/buildroot` directory. The same goes for gconfig and menuconfig when selecting that as configuration tool.

3.3.1. Using the official Buildroot user manual

For general documentation on Buildroot see the Buildroot user manual [RD3].

Note that in the LINUXBUILD environment the `output` Buildroot build directory directory mentioned in the Buildroot user manual is set up as `dist/buildroot/build-br` directory and other directories mentioned are inside the `dist/buildroot/buildroot-src` directory.

3.3.2. Buildroot toolchain configuration

Note that the Buildroot toolchain configuration is separate from the LINUXBUILD toolchain configuration (even though the default settings for the two are the same). By default, an external named toolchain with a default path is used. Buildroot can be configured to build a toolchain. To use that for building the Linux kernel LINUXBUILD needs to be configured to use it.

3.3.3. Adding custom files to the file system

There are many ways to add custom files to the file system documented in the Buildroot user manual [RD4], in the “Customizing the generated target filesystem” section. See Section 3.3.1 for details on directories.

The quick and dirty way is to simply put files in the Buildroot build directory `dist/buildroot/build-br/target` (after an initial build) that are then copied to the final file system building the image. However, such

additions disappears on a “make clean”. Refer to the Buildroot user manual [RD4] for more permanent additions to the file system that is intact on complete rebuilds.

3.3.4. Network Configuration

Adding network settings for the network interfaces can be done by editing the `/etc/network/interfaces` file (see Section 3.3.3 on custom files). For example, setting eth0 in DHCP and eth1 to a static IP address is done by editing the interfaces file as follows:

```
# Configure Loopback
auto lo
iface lo inet loopback

# Do DHCP for ETH0
auto eth0
iface eth0 inet dhcp

# Static IP for ETH1
auto eth1
iface eth1 inet static
    address 192.168.1.207
    network 192.168.1.0
    netmask 255.255.255.0
    broadcast 192.168.1.255
    gateway 192.168.1.1
```

3.4. Linux configuration

Here some Linux kernel related options outside of the kernel configuration can be set up. The number of threads used for kernel compilation for Linux kernel compilation can be set. Whether LINUXBUILD should set up file system source for Linux or not can also be set up(using Buildroot as distribution, y is the way to go).

In addition, the LINUXBUILD GUI can launch a separate configuration session for Linux. Alternatively configuration can be started with e.g. **make linux-xconfig** from the top level directory or **make xconfig** from the `linux` directory. The same goes for gconfig and menuconfig when selecting that as configuration tool.

LEON specific Linux kernel configuration options are documented separately in the LEON Linux User's Manual [RD3].

3.5. LEON Linux Loader configuration

This is where one can configure how mklinuximg should be called. See mklinuximg documentation for details [RD1].

This section links the Linux kernel to main memory and sets up the LEON Linux RAM loader responsible for setting up a low-level environment required for the SPARC Linux kernel.

3.6. MKPROM2 configuration

This is where one can configure if and how MKPROM2 should be used. If selected, the path needs to be specified, unless mkprom2 is in PATH. Hardware specific parameters needs to be set to get a functional PROM image. See MKPROM2 documentation for details [RD0].

The resulting PROM image can be loaded to FLASH using the GRMON flash commands **flash unlock**, **flash erase** and **flash load**. The PROM image can also be loaded into TSIM or GRSIM and executed.

4. Building

After installing Linux, configuring LINUXBUILD and all the selected components, the build process is started by typing **make build** (or simply **make**) in a shell, or by double clicking the build command inside the GUI under Step 3.

Individual build steps can be performed with **make buildroot**, **make linux**, **make mklinuximg**, **make mkprom**. This can be convenient when used with care.

Resulting images and file system images are found in the `output/` directory. A RAM image, that can e.g. be loaded into memory and run via GRMON or in TSIM, is created as `output/images/image.ram`. Corresponding symbols can be found in `output/images/image`. If MKPROM is configured and used, a PROM image is created from the RAM image. That PROM image is created as `output/images/image.prom`.

After major configuration changes or removal of any previously included Buildroot packages, rebuilding from scratch might be required. To do that either type **make clean** in a shell, or double click the clean command inside the GUI under Step 3. The Linux kernel is very good at knowing when things need to be rebuilt, but Buildroot requires more decisions from the user when it is time to rebuild from scratch.

If the build process fails it may be due to that a required tool is missing. Build logs for different stages can be found in the respective build directories (e.g. linux/build-linux/build.log or dist/buildroot/build-br/build.log) or in the output directory (e.g. output/images/mklinuximg.log or output/images/mkprom.log).

5. Upgrading

5.1. Upgrading to a newer LINUXBUILD release

There is no direct way to upgrade LINUXBUILD itself, but the `linux/linux-src` and `dist/buildroot/buildroot-src` trees can be moved from the old LINUXBUILD directory to the new, also the current configuration can be saved from the xconfig GUI and loaded into the new LINUXBUILD installation or manually by copying the corresponding .config files.

5.2. Upgrading to a newer LEON Linux release

The easiest way to upgrade to a newer LEON Linux release is to do a new LINUXBUILD install, install Linux from there and replace `linux/linux-src` in the currently installed LINUXBUILD with that of the new one. New default configurations can be found in the `linux/leon-linux/config` directory of the new LINUXBUILD install.

5.3. Following upstream stable Linux

See the LEON Linux User's Manual [RD3] for details on how to follow the upstream stable branch that the LEON Linux release is based upon.

5.4. Upgrading to a newer LEON Buildroot release

The easiest way to upgrade to a newer LEON Buildroot release is to download the latest LEON Buildroot as a separate release and do a fresh install using

```
make install
```

and then and then replace `dist/buildroot/buildroot-src` in the LINUXBUILD directory with the `buildroot-git` directory produced by the separate install (or soft link to it). New default configurations can be found in the `config` directory of the new LEON Buildroot install. Apart from upgrading to the latest LEON Buildroot release, this makes the `dist/buildroot/buildroot-src` source directory be a full git repository with the LEON Buildroot patches put on top of an upstream release.

Buildroot is not robust robust in tracking dependencies for things that has once been downloaded, built, installed, etc. so it is advised to do

```
make clean
```

before building after changing the underlying Buildroot sources.

5.5. Following upstream stable Buildroot

First, follow the above flow to upgrade the Buildroot source directory to a full git repository. The rest of the actions is done from the `dist/buildroot/buildroot-src`. If following these steps without using LINUXBUILD, do it from the `buildroot-git` directory of the LEON Buildroot release install.

Our LEON Buildroot releases are set up as a relatively small number of patches on top of upstream Buildroot releases. The idea is that is should be easy for the user to fetch the latest updates from the upstream stable branch and rebase the LEON branch on top of the latest corresponding stable branch, or even to an new Buildroot release series if so desired. Thus, one can get the latest upstream fixes without necessarily having to wait for a new LEON Buildroot release from us.

The upstream Buildroot stable branches can be found from the `origin` remote already set up. If one e.g. is on the branch `leon-2021.02-1.0` the corresponding upstream branch is `2021.02.x` (with a literal `x`). Official upstream stable releases, to continue the 2021.02 example, are tagged on the form `2021.02.`*`n`* where *`n`* is replaced by a number.

To pull in the latest stable patches, assuming the LEON patches from the LEON Buildroot release are in a branch leon-2021.02-1.0, do:

```
git fetch origin
git rebase origin/2021.02.x leon-2021.02-1.0
```

Rebasing upon a release tag or a new Buildroot release series is done in a similar fashion.

Buildroot is not always robust in tracking dependencies for things that has once been downloaded, built, installed, etc. so it is advised to do

```
make clean
```

before building after changing the underlying Buildroot sources.

5.6. Upgrading MKLINUXIMG

The MKLINUXIMG component can be upgraded when Cobham Gaisler release a new updated package. That upgrade process can be started from the KConfig GUI (make xconfig).

6. Support

For support contact the Cobham Gaisler support team at support@gaisler.com.

When contacting support, please identify yourself in full, including company affiliation and site name and address. Please identify exactly what product that is used, specifying if it is an IP core (with full name of the library distribution archive file), component, software version, compiler version, operating system version, debug tool version, simulator tool version, board version, etc.

The support service is only for paying customers with a support contract.

Cobham Gaisler AB
Kungsgatan 12
411 19 Gothenburg
Sweden
www.caes.com/Gaisler
sales@gaisler.com
T: +46 31 7758650
F: +46 31 421407